Applications to Computational Molecular Biology

Giuseppe Lancia

1 Introduction

Computational (Molecular) Biology is a relatively young science, which has experienced a tremendous growth in the last decade. The seeds for the birth of Computational Biology were sowed in the end of the Seventies, when computers became cheaper, easily available and simpler to use, so that some biology labs decided to adopt them, mainly for storing and managing genomic data. The use of computers allowed the quick completion of projects that before would have taken years. With a snowball effect, these projects generated larger and larger amounts of data whose management required more and more powerful computers. Recently, the advent of the Internet has allowed all the laboratories to share their data, and make them available worldwide through some new genomic data banks (such as GenBank [12], EMBL [79], PDB [13]). Without any doubt, today a computer is a necessary instrument for doing research in molecular biology, and is invariably present, together with microscopes and other more classical instruments, in any biotechnology lab.

Currently, there is not a full agreement on what "Computational Biology" means. Some researchers use a very loose definition such as "Computer Science applied to problems arising in Molecular Biology". This definition encompasses at least two major types of problems: (i) Problems of storage, organization and distribution of large amounts of genomic data; (ii) Problems of interpretation and analysis of genomic data. We would rather call problems of the first type *Bioinformatics* problems, and reserve the term *Computational Biology* for the study of problems of the second type. Furthermore, although Computer Science plays a key role in the solution of such problems, other disciplines such as Discrete Mathematics, Statistics and Optimization, are as important. Hence, the following definition of Computational Biology will be adopted in this survey: "*Study of mathematical and computational problems of modeling biological processes in the cell, removing experimental errors from genomic data, interpreting the data and providing theories about their biological relations*".

The study of a typical problem in Computational Biology starts by a representation of the biological data in terms of mathematical objects, such as strings, graphs, permutations, etc. The biological relations of the data are mapped into mathematical relations, and the original question is expressed either as a feasibility or an optimization problem $\mathcal{P}$. From this point on, depending on the nature of $\mathcal{P}$, standard techniques can be adopted for its solution. A first step is usually the study of the computational complexity of the problem. This may possibly lead to a polynomial algorithm, a result that has been obtained, in particular, with Dynamic Programming for several problems in Computational Biology. This line of attack is conducted primarily by computer scientists. However, most times the problems turn out to be NP-hard optimization problems. Their exact solution becomes then work for scientists with a background in Combinatorial Optimization, and, particularly, Operations Research (OR). Standard OR techniques, such as Integer Linear or Quadratic Programming, solved by Branch-and-Cut, Branch-and-Price, or Lagrangian Relaxation, have been increasingly adopted for Computational Biology problems in the last years. Correspondingly, the recent years have witnessed an increasing number of scientists with OR background approaching the field of Computational Biology.

In this chapter, we intend to review some of the main results in Computational Biology that were obtained with the use of Mathematical Programming techniques. The problems will come from different areas, touching virtually every aspect of modern Computational Biology. However, many relevant results that do not employ Mathematical Programming will not be addressed in this survey. For a complete coverage of the many beautiful and important results that have been achieved in Computational Biology, the reader is referred to the classic textbooks by the pioneers of this field, i.e., Michael Waterman [82], Dan Gusfield [38] and Pavel Pevzner [72].

Perhaps the main contribution of Computational Biology so far lies in the key role it played in the completion of the Human Genome Project, which culminated with the 2001 announcement of the completion of the sequencing of the human genome [45, 80]. This result was obtained thanks to experimental techniques, such as *Shotgun Sequencing*, which posed challenging computational problems. Sophisticated algorithms for these problems (which are mostly variants of the famous *Shortest Superstring Problem*, see Garey and Johnson [31] problem [SR9]) were developed mainly by Myers et al. [67, 68, 84].

The material in this chapter is organized in sections. In each section, we describe one or more problems of the same nature, and the Mathematical Programming approaches that were proposed in the literature for their solution. Some of these problems are solved via a reduction to very well known optimization problems, such as the TSP and Set Covering. For these problems, the core of the analysis will focus on the modeling and the reduction, while the solving algorithm can be assumed to be a standard, state-of-the-art, code for the target optimization problem (e.g., CONCORDE [4] for the TSP). For the remaining problems, *ad hoc* algorithms were developed for their solution. Many of these algorithms are based on Integer Programming formulations with an exponential number of inequalities (or variables), that are solved by Branch-and-Cut (respectively, Branch-and-Price).

The chapter is organized as follows:

- **Alignment Problems.** In Section 3 we review several problems related with the alignment of genomic objects, such as DNA or RNA sequences and secondary or tertiary structures. We distinguish three types of alignment:

i) Sequence vs Sequence Alignment. Section 3.1 is devoted to the problem of aligning genomic sequences. First, we discuss the classical multiple alignment problem, studied by Reinert et al. [75] and Kececioglu et al. [50], who developed a Branch-and-Price approach for its solution. Then, we describe a heuristic approach for the same problem, by Fischetti et al. [28]. The approach generalizes an approximation algorithm by Gusfield [37] and uses a reduction to the Minimum Routing Cost Tree problem, solved by Branch-and-Price. Closely related to the alignment problem, are the so called "center" problems, in which one tries to determine a sequence as close or as far as possible from a set of input sequences. For this problem, we describe an approximation algorithm based on LP relaxation and Randomized Rounding, by Ben Dor et al. [11]. We also discuss some works that, developing similar ideas, obtain Polynomial Time Approximation Schemes (PTAS) for the consensus and the farthest-sequence problems (Li et al. [61], Ma [64], Lanctot et al. [54]).

ii) Sequence vs Structure Alignment. In Section 3.2 we describe the problem of aligning two RNA sequences, when for one of them the secondary structure is already known. For this problem, we describe a Branch-and-Price approach by Lenhof et al. [60] and Kececioglu et al. [50].

iii) Structure vs Structure Alignment. In Section 3.3 we consider the problem of comparing two protein tertiary structures. For this problem, we review an Integer Linear Programming approach, proposed by Lancia et al. [53], and another approach, based on a Quadratic Programming formulation and Lagrangian Relaxation, given by Caprara and Lancia [20].

- **Single Nucleotide Polymorphisms.** In Section 4 we describe some combinatorial problems related with human diversities (polymorphisms) at the genomic level. The *Haplotyping* problem consists in determining the values of a set of polymorphic sites in a genome. First, we discuss the single individual haplotyping problem, for which some Integer Programming techniques were employed by Lancia et al. [52] and Lippert et al. [63]. Then, we review the population version of the problem, as studied by Gusfield [39, 40]. The approach employs a reduction to a graph problem, which is then solved by Integer Programming.

- **Genome Rearrangements.** In Section 5, we describe a successful Branch-and-Price approach, by Caprara et al. [21, 22] for computing the evolutionary distance between two genomes evolved from a common ancestor.

- **Mapping problems and the TSP.** In Section 6 we review the *Physical Mapping* problem and its connections with the TSP problem and the Consecutive One property for 0-1 matrices. We first describe an Integer Programming approach for mapping a set of probes on a set of clones, proposed by Alizadeh et al. [3]. We then turn to the problem of mapping radiation hybrids, studied by Ben Dor et al. [9, 10] and by Agarwala et al [2]. These papers reduced the problem to a standard TSP, for which they used the best available software, based on Branch-and-Cut. Finally, we mention a Branch-and-Cut approach for the physical mapping of probes coming from the chromosomes ends, by Christof et al. [26].

- **Applications of Set Covering.** In Section 7 we mention a few problems of Computational Biology whose solution algorithm consisted mainly in a reduction to the Set Cover problem. In particular, we review the PCR primer design problem (Pearson et al. [71] and Nicodeme and Steyaert [70]) and the analysis of microarray expression data (Halldorsson et al. [41, 14]).

The elementary concepts of molecular biology needed to follow the material are introduced in Section 2 and in the context of the following sections. The web provides a wonderful source of information for further reading. In particular, we suggest visiting the sites of National Institute of Health (`www.nih.gov`) and the European Molecular Biology Laboratory (`www.embl.de`).

2 Elementary Molecular Biology Concepts

One of the major problems encountered by researchers coming from mathematical fields and approaching computational biology, is the lack of vocabulary and of understanding of the biological phenomena underlying the mathematical models of the problems. This section is intended to provide the reader with some, very basic, preliminary notions, and many additional biology concepts will be given in the context of the specific problems described in the following sections. A nice exposition of molecular biology at an introductory level can be found in the Human Genome Project Primer [24]. Alternatively, Fitch [29] has written an introductory paper aimed specifically at mathematicians and computer scientists. For a comprehensive and deeper analysis of the topic, the reader is referred to some of the standard textbooks in molecular biology, such as the one by Watson, Gilman, Witkowski and Zoller [83].

2.1 The DNA

A complete description of each living organism is contained in its *genome*. This can be thought of as a "program", in a very special language, describing the set of instructions to be followed by the organism in order to grow and fully develop to its final form. The language used by nature to encode life is represented by the *DNA*.

The *deoxyribonucleic acid* (DNA) is present in each of the organism's cells, and consists of two linear sequences (*strands*) of tightly coiled threads of *nucleotides*. Each nucleotide is a molecule composed by one sugar, one phosphate and one nitrogen-containing chemical, called a *base*. Four different bases are present in the DNA, namely *Adenine* (A), *Thymine* (T), *Cytosine* (C) and *Guanine* (G). The particular order of the bases is called the *DNA sequence* and

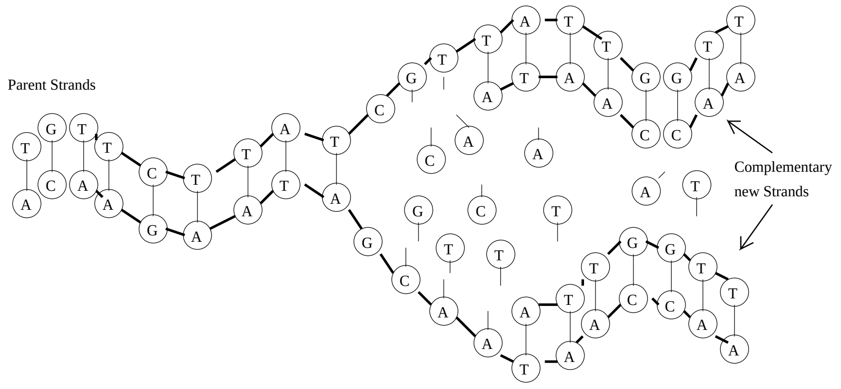


Figure 1: Schematic DNA replication.

varies among different organisms. The sequence specifies the precise genetic instructions to create a particular form of life with its own unique traits.

The two DNA strands are inter-twisted in a typical helix form, and held together by bonds between the bases on each strand, forming the so called *base pairs*. A complementarity law relates one strand to its opposite, the only admitted pairings being Adenine with Thymine (A $\leftrightarrow$ T) and Cytosine with Guanine (C $\leftrightarrow$ G). Thanks to this complementarity, the DNA has the ability to replicate itself. Each time a cell divides into two new cells, its full genome is duplicated: First, the DNA molecule unwinds and the bonds between the base pairs break (figure 2.1). Then, each strand directs the synthesis of a complementary new strand, by matching up free nucleotides floating in the cell with their complementary bases on each of the separated strands. The complementarity rules should ensure that the new genome is an exact copy of the old one. However, although very reliable, this process is not completely error-free and there is the possibility that some bases are lost, duplicated or simply changed. Variations to the original content of a DNA sequence are called *mutations* and can affect the resulting organism or its offspring. In many cases mutations are deadly. In some other cases, they can be completely harmless, or lead, in the long run, to the evolution of a species.

In humans the DNA in each cell contains about $3 \cdot 10^9$ base pairs. Human cells are *diploid*, i.e. their DNA is organized in *pairs* of *chromosomes*. The two chromosomes that form a pair are called *homologous*. One copy of each pair is inherited from the father and the other copy from the mother. The cells of some lower organisms, such as fungi and single-celled algae, may have only a single set of chromosomes. Cells of this type are called *haploid*.

2.2 Genes and proteins

The information describing an organism is encoded in its genome by means of a universal code, called the *genetic code*. This code is used to describe how to build the *proteins*, which provide the structural components of cells and tissues, as well as the *enzymes* needed for essential biochemical reactions.

Not all of the DNA sequence contains coding information (as a matter of fact, only a small fraction does). The information-coding DNA regions are organized into *genes*, where each gene is responsible for the coding of a different protein. Regions of the DNA sequences containing genes are called *exons*, since they are "expressed" by the mechanism which builds the proteins. Conversely, the *introns* are non-coding regions whose functions are still obscure.

The gene is the unit of heredity, which, together with many other such units, is transmitted from parents to offspring. Each gene, acting either alone or with other genes, is responsible for one or more characteristics of the resulting organism. Each characteristic can take many distinct values, called *alleles.* For a simplistic example, the alleles for the eye-color could be {black, blue, green, hazel, brown}.

Genes occur on the chromosomes, at specific positions called their *loci.* In diploid organisms, for each gene on one chromosome, there is a corresponding similar gene on the homologous chromosome. Thanks to this redundancy, sometimes an organism may survive even when one copy of a gene is defective, provided the other is not. A pair of corresponding genes can consist of the same allele on both chromosomes, or of different alleles. In the first case we say the organism is *homozygous* for the gene, while in the second it is *heterozygous.* The human genome is estimated to comprise between $30,000$ and $50,000$ genes, whose size ranges from a thousand to hundreds of thousands of base pairs.

A *protein* is a large molecule, consisting of one or more linear chains of *amino acids*, folded into a characteristic 3-dimensional shape called the protein's *native state.* The linear order of the amino acids is determined by the sequence of nucleotides in the gene coding for the protein. There exist 20 amino acids, each of which is identified by some triplets of DNA letters. The DNA triplets are also called *codons* and the correspondence of codons with amino acids is the *genetic code.* Since there are $4^3 = 64$ possible triplets of nucleotides, most amino acids are coded by more than one triplet (called *synonyms* for that amino acid), whereas a few are identified by a unique codon. For example, the amino acid Proline (Pro) is represented by the four triplets CCT, CCC, CCA and CCG. The code redundancy reduces the probability that random mutations of a single nucleotide can cause harmful effects. For instance, any mutation in the third base of Proline would result in a codon still correctly identifying the amino acid.

In order to build a protein, the DNA from a gene is first copied onto the *messenger RNA* (mRNA), which will serve as a template for the protein synthesis. This process is termed *transcription.* Only exons are copied, while the introns are spliced out from the molecule. Then the mRNA moves out of the cell's nucleus, and some cellular components named *ribosomes* start to read its triplets sequentially, identify the corresponding amino acids and link them so as to form the protein. The process of translation of genes into proteins, requires the presence of signals to identify the beginning and the end of each information-coding region. To this purpose, there are codons that specifically determine where a gene begins and where it terminates.

2.3 Some experiments in molecular biology

We now describe some common experiments performed in molecular biology labs, which are relevant to the problems described in this survey.

- **Cloning in-vivo and Polymerase Chain Reaction.**

  Since most experiments cannot be conducted on a single copy of a DNA region, the preliminary step for further analysis consists in making a large quantity of copies (*clones*) of the DNA to be studied. This process is referred to as *cloning* or *DNA amplification*. There are two basic cloning procedures, namely *cloning in-vivo* and *Polymerase Chain Reaction* (PCR). Cloning in-vivo, consists in inserting the given DNA fragments into a living organism, such as bacteria and yeast cells. After the insertion, when the host cells are are naturally duplicated, the foreign DNA gets duplicated as well. Finally, the clones can be extracted from the new cells.

  PCR is an automatic technique by which a given DNA sequence can be amplified hundreds of millions of times within a few hours. In order to amplify a double-stranded DNA sequence by PCR, one needs two short DNA fragments (*primers*) from the ends of the region of interest. The primers are put in a mixture containing the region of interest, free DNA bases and a specialized polymerase enzyme. The mixture is heated and the two strands separate. Later, the mixture is cooled and the primers bind to their complementary sequences on the separated strands. Finally, the polymerase enzyme synthesizes new complementary strands by extending the primers. By repeating this cycle of heating and cooling, an exponential number of copies of the target DNA sequence can be readily obtained.

- **Gel Electrophoresis.** Gel electrophoresis is an experiment by which DNA fragments can be separated according to their size, which can then be estimated with high precision. A large amount of DNA fragments are put in an agarose gel, to which an electric field is applied. Under the field, the fragments migrate in the gel, moving with a speed inversely proportional to their size. After some time, one can compare the position of the fragments in the gel to the position of a sample molecule of known size, and derive their size by some simple computation.

- **(Fragment) Sequencing.** *Sequencing* is the process of reading out the ordered list of bases from a DNA fragment. Due to technological limitations, it is impossible to sequence fragments longer than a few hundred base pairs. One technique for sequencing is as follows. Given a fragment which has been amplified, say, by PCR, the copies are cut randomly at all positions. This way one obtains a set of nested fragments, differing in length by exactly one nucleotide. The specific base at the end of each successive fragment is then detected, after the fragments have been separated by gel electrophoresis. The machine to sequence DNA is called a *sequencer*. To each output base, the sequencer attaches a value (*confidence* *level*) which represents the probability that the base has been determined correctly.

- **(Genome) Shotgun Sequencing.**

  In order to sequence a long DNA molecule (e.g., a whole genome), this must first be amplified into many copies, and then be broken, at random, into several fragments, of about 1,000 nucleotides each, which are individually fed to a sequencer. The cloning phase is necessary so that the fragments can have nonempty overlap. From the overlap of two fragments one may infer a longer fragment, and so on, until the original DNA sequence has been reconstructed. This is, in essence, the principle of *Shotgun Sequencing*, in which the fragments are *assembled* back into the original sequence by using sophisticated algorithms and powerful computers.

  The *assembly* (i.e. overlap and merge) phase is complicated by the fact that in a genome there exist many regions with identical content (*repeats*) scattered all around and due to replicating events during evolution. The repeats may be confused by the assembler to be all copies of a same, unique, region. To partly overcome the problems of repeats, some fragments used in shotgun sequencing may have some extra information attached. In particular, some fragments can be obtained by a process that generates pairs (called *mate pairs*) of fragments instead of individual ones. Each pair is guaranteed to come from the same copy of a chromosome and to have a given distance between its elements. A pair of mates can help since, even if one of them comes from a repeat region, there is a good chance that the other does not.

3 Alignment Problems

Oftentimes, in Computational Biology, one must compare objects which consist of a set of elements arranged in a linearly ordered structure. A typical example is the genomic sequence, which, as we saw, is equivalent to a string. Another example is the protein, when this is regarded as a linear chain of amino acids (and hence with a *first* and a *last* amino acid).

Aligning two (or more) such objects consists in determining subsets of corresponding elements in each. The correspondence must be order-preserving, i.e., if the i-th element of object 1 corresponds to the k-th element of object 2, no element following i in object 1 can correspond to an element preceding k in object 2.

The situation can be described graphically as follows. Given two objects, where the first has n elements, numbered $1, \ldots, n$ and the second has m elements, numbered $1, \ldots, m$, we consider the complete bipartite graph

$$W_{nm} := ([n], [m], L) \tag{1}$$

where $L = [n] \times [m]$ and $[k] := \{1, \ldots, k\}$ $\forall k \in Z^+$.

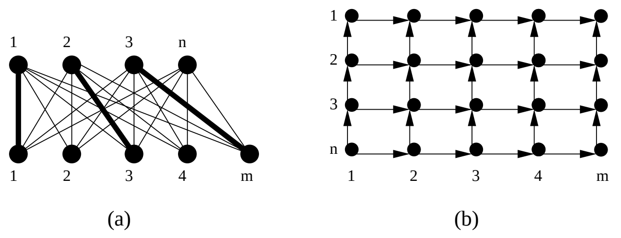


Figure 2: (a) A noncrossing matching (alignment). (b) The directed grid.

We hereafter call a pair (i, j) with $i \in [n]$ and $j \in [m]$ a *line* (since it *aligns* i with j), and we denote it by $[i, j]$. L is the set of all lines. Two lines $[i, j]$ and $[i', j']$, with $[i, j] \neq [i', j']$, are said to *cross* if either $i' \geq i$ and $j' \leq j$ or $i' \leq i$ and $j' \geq j$ (by definition, a line does not cross itself). Graphically, crossing lines correspond to lines that intersect in a single point. A matching is a subset of lines no two of which share an endpoint. An alignment is identified by a *noncrossing matching*, i.e., a matching for which no two lines cross, in W_{nm}. Figure 2(a) shows (in bold) an alignment of two objects.

In the following Sections 3.1, 3.2 and 3.3, we will describe Mathematical Programming approaches for three alignment problems, i.e., Sequences vs. Sequences (Reinert et al. [75], Kececioglu et al. [50]), Sequences vs. Structures (Lenhof et al. [60], Kececioglu et al. [50]), and Structures vs. Structures (Lancia et al. [53], Caprara and Lancia [20]) respectively. All Integer Programming formulations for these problems contain binary variables x_l to select which lines $l \in L$ define the alignment, and constraints to insure that the selected lines are noncrossing. Therefore, it is appropriate to discuss here this part of the models, since the results will apply to all the following formulations. We will consider here the case of two objects only. This case can be generalized to more objects, as mentioned in Section 3.1.

Let X_{nm} be the set of incidence vectors of all noncrossing matchings in W_{nm}. A noncrossing matching in W_{nm} corresponds to a stable set in a new graph, G_L (the *Line Conflict Graph*), defined as follows. Each line $l \in L$ is a vertex of G_L, and two vertices l and h are connected by an edge if the lines l and h cross.

The graph G_L has been studied in Lancia et al. [53], where the following theorem is proved:

Theorem 1 *([53]) The graph G_L is perfect.*

A complete characterization of $\text{conv}(X_{nm})$ in terms of linear inequalities, is given –besides non-negativity– by the following *clique inequalities*:

$$\sum_{l \in Q} x_l \leq 1 \qquad \forall Q \in \text{clique}(L). \tag{2}$$

where $\text{clique}(L)$ denotes the set of all cliques in G_L. Given weights x_l^* for each line l, the separation problem for the clique inequalities (2) consists in finding a clique Q in G_L with $x^*(Q) > 1$. From Theorem 1, we know that this problem can be solved in polynomial time by finding the maximum x^*-weight clique in G_L. Instead of resorting to slow and complex algorithms for perfect graphs, the separation problem can be solved directly by Dynamic Programming, as described in Lenhof et al. [60] and, independently and with a different construction, in Lancia et al. [53]. Both algorithms have complexity $O(nm)$. We describe here the construction of [60].

Consider L as the vertex set of a directed grid, in which vertex $[1, n]$ is put in the lower left corner and vertex $[m, 1]$ in the upper right corner (see Figure 2(b)).

At each internal node $l = [i, j]$, there are two incoming arcs, one directed from the node below, i.e. $[i, j + 1]$, and one directed from the node on the left, i.e. $[i - 1, j]$. Each such arc has associated a length equal to x^*_l.

Theorem 2 *([60, 53]) The nodes on a directed path P from $[1, n]$ to $[m, 1]$ in the grid correspond to a clique Q, of maximal cardinality, in G_L and vice versa.*

Then the most violated clique inequality can be found by taking the longest $[1, n]$-$[m, 1]$ path in the grid. There is a violated clique inequality if and only if the length of the path (plus x^*_{1n}) is greater than 1.

Closely related to the problem of finding a largest-weight clique in G_L is the problem of finding a largest-weight stable set in G_L (i.e., a maximum weighted noncrossing matching in W_{nm}). Also for this problem, there is a Dynamic Programming solution, of complexity $O(nm)$. The basic idea is to compute the matching recursively. Let $V(a, b)$ be the value of the maximum w-weight noncrossing matching of nodes $\{1, \ldots, a\}$ with nodes $\{1, \ldots, b\}$, where each line $[i, j]$ has a weight w_{ij}. Then, in the optimal solution, either a is matched with b, and $V(a, b) = w_{ab} + V(a-1, b-1)$, or one of a and b is unmatched, in which case $V(a, b) = \max\{V(a - 1, b), V(a, b - 1)\}$. With a look-up table, $V(n, m)$ can be computed in time $O(nm)$. Incidentally, this algorithm coincides with the basic algorithm for computing the *edit distance* of two strings (a problem described in the next section), rediscovered independently by many authors, among which Smith and Waterman [78]. In [20], Caprara and Lancia solve the Lagrangian Relaxation of a model for aligning two protein structures, by reducing it to finding the largest weighted noncrossing matching in a suitable graph. This is described in Section 3.3.

3.1 Sequence vs Sequence Alignments

Comparing genomic sequences drawn from individuals of the same or different species is one of the fundamental problems in molecular biology. In fact, such comparisons are needed to identify highly conserved (and therefore presumably functionally relevant) DNA regions, spot fatal mutations, suggest evolutionary relationships, and help in correcting sequencing errors.

A genomic sequence can be represented as a string over an alphabet Σ consisting of either the 4 nucleotide letters or the 20 letters identifying the 20 amino acids. Aligning a set of sequences (i.e., computing a *Multiple Sequence Alignment*) consists in arranging them in a matrix having each sequence in a row. This is obtained by possibly inserting gaps (represented by the '`-`' character) in each sequence so that they all result of the same length. The following is a simple example of an alignment of the sequences `ATTCCGAC`, `TTCCCTG` and `ATCCTC`. The example highlights that the pattern `TCC` is common to all sequences.

```
A  T  T  C  C  G  A  -  C
-  T  T  C  C  C  -  T  G
A  -  T  C  C  -  -  T  C
```

By looking at a column of the alignment, we can reconstruct the events that have happened at the corresponding position in the given sequences. For instance, the letter `G` in sequence 1, has been *mutated* into a `C` in sequence 2, and *deleted* in sequence 3.

The multiple sequence alignment problem has been formalized as an optimization problem. The most popular objective function for multiple alignment generalizes ideas from optimally aligning two sequences. This problem, called *pairwise alignment*, is formulated as follows: Given symmetric costs (or, alternatively, profits) $\gamma(a, b)$ for replacing a letter a with a letter b and costs $\gamma(a, -)$ for deleting or inserting a letter a, find a minimum-cost (respectively, maximum-profit) set of symbol operations that turn a sequence S' into a sequence S''. For genomic sequences, the costs $\gamma(\cdot, \cdot)$ are usually specified by some widely used *substitution matrices* (e.g., PAM and BLOSUM), which score the likelihood of specific letter mutations, deletions and insertions. The pairwise alignment problem can be solved by Dynamic Programming in time and space $O(l^2)$, where l is the maximum length of the two sequences, by Smith and Waterman's algorithm [78]. The value of an optimal solution is called the *edit distance* of S' and S'' and is denoted by $d(S', S'')$.

Formally, an *alignment* $\mathcal{A}$ of two or more sequences is a bidimensional array having the (gapped) sequences as rows. The value $d_{\mathcal{A}}(S', S'')$ of an alignment of two sequences S' and S'' is obtained by adding up the costs for the pairs of characters in corresponding positions, and it is easy to see that $d(S', S'') = \min_{\mathcal{A}} d_{\mathcal{A}}(S', S'')$. This objective is generalized, for k sequences $\{S_1, \ldots, S_k\}$, by the *Sum–of–Pairs* (SP) score, in which the cost of an alignment is obtained by adding up the costs of the symbols matched up at the same positions, over all the pairs of sequences,

$$SP(\mathcal{A}) := \sum_{1 \le i < j \le k} d_{\mathcal{A}}(S_i, S_j) = \sum_{1 \le i < j \le k} \sum_{l=1}^{|\mathcal{A}|} \gamma(\mathcal{A}[i][l], \mathcal{A}[j][l]) \qquad (3)$$

where $|\mathcal{A}|$ denotes the length of the alignment, i.e., the number of its columns.

Finding the optimal SP alignment was shown to be NP-hard by Wang and Jiang [81]. A straightforward generalization of the Dynamic Programming from 2 to k sequences, of length l, leads to an exponential-time algorithm of complexity $O(2^k l^k)$. In typical real-life instances, while k can be possibly small (e.g., around 10), l is in the order of several hundreds, and the Dynamic Programming approach turns out to be infeasible for all but tiny problems.

As an alternative approach to the ineffective Dynamic Programming algorithm, Kececioglu et al. [50] (see also Reinert et al. [75] for a preliminary version) proposed an approach based on Mathematical Programming to optimize an objective function called the Maximum Weight Trace (MWT) Problem. The MWT generalizes the SP objective as well as many other alignment problems in the literature. The problem was introduced in 1993 by Kececioglu [49], who described a combinatorial Branch-and-Bound algorithm for its solution.

3.1.1 Trace and Mathematical Programming approach

The *trace* is a graph theoretic generalization of the noncrossing matching to the case of the alignment of more than two objects. Suppose we want to align k objects, of $n_1, n_2, \ldots, n_k$ elements. We define a complete k-partite graph $W_{n_1,\ldots,n_k} = (V, L)$, where $V = \bigcup_{i=1}^{k} V_i$, the set $V_i = \{v_{i1}, \ldots, v_{in_i}\}$ denotes the nodes of level i (corresponding to the elements of the i-th object), and $L = \bigcup_{1 \le i < j \le k} V_i \times V_j$. Each edge $[i, j] \in L$ is still called a line. Given an alignment $\mathcal{A}$ of the objects, we say that the alignment *realizes* a line $[i, j]$ if i and j are put in the same column of $\mathcal{A}$. A trace T is a set of lines such that there exists an alignment $\mathcal{A}$ that realizes all the lines in T. The Maximum Weight Trace (MWT) Problem is the following: given weights $w_{[i,j]}$ for each line $[i, j]$, find a trace T of maximum total weight.

Let A be the set of directed arcs pointing from each element to the elements that follow it in an object (i.e., arcs of type (v_{is}, v_{it}) for $i = 1, \ldots, k$, $1 \le s < t \le n_i$). Then a trace T is characterized as follows:

Proposition 3 *([75]) T is a trace if and only if there is no directed mixed cycle in the mixed graph $(V, T \cup A)$.*

A directed mixed cycle is a cycle containing *both* arcs and edges, in which the arcs (elements of A) are traversed according to their orientation, while the undirected edges (lines of T) can be traversed in either direction.

Using binary variables x_l for the lines in L, the MWT problem can be modeled as follows [50]:

$$\max \sum_{l \in L} w_l x_l \tag{4}$$

subject to

$$\sum_{l \in C \cap L} x_l \le |C \cap L| - 1 \qquad \forall \text{ directed mixed cycles } C \text{ in } (V, L \cup A) \tag{5}$$

$$x_l \in \{0, 1\} \qquad \forall l \in L. \tag{6}$$

The polytope $\mathcal{P}$ defined by (5) and $0 \le x_l \le 1,\ l \in L$, is studied in [50], and some classes of facet-defining inequalities are described. Among them, are the clique inequalities (2), relative to each subgraph $W_{n_i,n_j} := (V_i, V_j, V_i \times V_j)$ of

$W_{n_1,\ldots,n_k}$, and the following subset of (5), called *mixed-cycle* inequalities. Given a directed mixed cycle C, a line $l = [i, j]$ is a *chord* of C if $C_1 \cup \{l\}$ and $\{l\} \cup C_2$ are directed mixed cycles, where C_1 and C_2 are obtained by splitting C at i and j.

Lemma 1 *([50]) Let C be a directed mixed cycle of $(V, L \cup A)$. Then the inequality*

$$x(C \cap L) \leq |C \cap L| - 1$$

is facet-defining for $\mathcal{P}$ if and only if C has no chord.

The mixed-cycle inequalities can be separated in polynomial time, via the computation of at most $O(\sum_{i=1}^{k} n_i)$ shortest paths in $(V, L \cup A)$, [50]. A Branch-and-Cut algorithm based on the mixed-cycle inequalities and the clique inequalities (as well as other cuts, separated heuristically) allowed Kececioglu et al. [50] to compute, for the first time, an optimal alignment for a set of 15 proteins of about 300 amino acids each from the SWISSPROT database, whereas the limit for the combinatorial version of Kececioglu's algorithm for this problem was 6 sequences of size 200 [49]. We remark the the optimal alignment of 6 sequences of size 200 is already out of reach for Dynamic Programming-based algorithms. We must also point out, however, that the length of the sequences is still a major limitation to the applicability of the Integer Programming solution, and the method is not suitable for sequences longer than a few hundred letters.

3.1.2 Heuristics for SP alignment

Due to the complexity of the alignment problem, many heuristic algorithms have been developed over time, some of which provide an approximation-guarantee on the quality of the solution found. In this section we describe one such heuristic procedure, which uses ideas from Network Design and Integer Programming techniques. This heuristic is well suited for long sequences.

A popular approach for many heuristics is the so called "progressive" alignment, in which the solution is incrementally built by considering the sequences one at a time. The most effective progressive alignment methods proceed by first finding a tree whose node set spans the input sequences, and then by using the tree as a guide for aligning the sequences iteratively. One such algorithm is due to Gusfield [37], who suggested to use a *star* (i.e., a tree in which at most one node -the *center*- is not a leaf), as follows. Let $\mathcal{S} = \{S_1, \ldots, S_k\}$ be the set of input sequences. Fix a node (sequence) S_c as the center, and compute $k - 1$ pairwise alignments $\mathcal{A}_i$, one for each each S_i aligned with S_c. These alignments can then be merged into a single alignment $\mathcal{A}(c)$, by putting in the same column two letters if they are aligned to the same letter of S_c. The following is an example of the alignments of S_1 = `ATGTC` and S_2 = `CACGG` with S_c = `ATCGC` and their merging.

				S_1	`-AT-GTC`
S_1	`AT-GTC`	S_2	`CA-CGG`	S_2	`CA-CG-G`
S_c	`ATCG-C`	S_c	`-ATCGC`	S_c	`-ATCG-C`

Note that each $\mathcal{A}_i$ realizes the edit distance of S_i to S_c, and it is true also in $\mathcal{A}(c)$ that each sequence is aligned optimally with S_c, i.e.

$$d_{\mathcal{A}(c)}(S_i, S_c) = d_{\mathcal{A}_i}(S_i, S_c) = d(S_i, S_c) \qquad \forall i. \tag{7}$$

Furthermore, for all pairs i, j, the triangle inequality $d_{\mathcal{A}(c)}(S_i, S_j) \leq d(S_i, S_c) + d(S_j, S_c)$ holds, since the distance in an alignment is a metric over the sequences (assuming the cost function γ is a metric over Σ). Let c^* be such that $SP(\mathcal{A}(c^*)) = \min_r SP(\mathcal{A}(r))$. Call OPT be the optimal SP value, and HEUR $:= SP(\mathcal{A}(c^*))$. By comparing $\sum_{1 \leq i < j \leq k} d(S_i, S_j)$ (a lower bound to OPT) to $\sum_{1 \leq i < j \leq k} (d(S_i, S_{c^*}) + d(S_j, S_{c^*}))$ (an upper bound to HEUR), Gusfield proved the following 2-approximation result:

Theorem 4 *([37]) HEUR $\leq (2 - \frac{2}{k})$ OPT.*

In his computational experiments, Gusfield showed that the above approximation ratio is overly pessimistic on real data, and that his star-alignment heuristic was finding solutions within 15% from optimum on average.

Gusfield's algorithm can be generalized so as to use *any* tree, instead of just a star, and still maintain the 2-approximation guarantee. The idea was first described in Wu et al. [85], in connection to a Network Design problem, i.e., the Minimum Routing Cost Tree (MRCT). Given a complete weighted graph of k vertices, and a spanning tree T, the *routing cost* of T is defined as the sum of the weights of the $\binom{k}{2}$ paths, between all pairs of nodes, contained in T. This value is denoted by $r(T)$. Computing the minimum routing cost tree is NP-hard, but a Polynomial Time Approximation Scheme (PTAS) is possible, and is described in [85].

To relate routing cost and alignments, we start by considering the set of sequences $\mathcal{S}$ as the vertex set of a complete weighted graph G, where the weight of an edge S_iS_j is $d(S_i, S_j)$. The following procedure, attributed by folklore to Feng and Doolittle, shows that, for any spanning tree T, we can build an alignment $\mathcal{A}(T)$ which is optimal for $k-1$ of the the $\binom{k}{2}$ pairs of sequences. A sketch of the procedure is as follows: (i) pick an edge S_iS_j of the tree and align recursively the two sets of sequences induced by the cut defined by the edge, obtaining two alignments, say $\mathcal{A}_i$ and $\mathcal{A}_j$; (ii) align optimally S_i with S_j; (iii) merge $\mathcal{A}_i$ and $\mathcal{A}_j$ into a complete solution, by aligning the columns of $\mathcal{A}_i$ and $\mathcal{A}_j$ in the same way as the letters of S_i are optimally aligned to the letters of S_j.

From this procedure, the validity of the next theorem follows.

Theorem 5 *For any spanning tree $T = (\mathcal{S}, E)$ of G, there exists an alignment $\mathcal{A}(T)$ such that $d_{\mathcal{A}(T)}(S_i, S_j) = d(S_i, S_j)$ for all pairs of sequences $S_iS_j \in E$.*

Furthermore, the alignment $\mathcal{A}(T)$ can be easily computed, as outlined above. By triangle inequality, it follows

Corollary 6 *For any spanning tree $T = (\mathcal{S}, E)$ of G, there exists an alignment $\mathcal{A}(T)$ such that $SP(\mathcal{A}(T)) \leq r(T)$.*

From Corollary 6, it follows that a good tree to use for aligning the sequences is a tree T^* such that $r(T^*) = \min_T r(T)$. This tree does not guarantee an optimal alignment, but guarantees the *best possible upper bound* to the alignment value, for an alignment obtained from a tree (such as Gusfield's alignment). In [28], Fischetti et al. describe a Branch-and-Price algorithm for finding the minimum routing cost tree in a graph, which is effective for up to $k = 30$ nodes (a large enough value for alignment problems). The algorithm is based on a formulation with binary variables x_P for each path P in the graph, and constraints that force the variables set to 1 to be the $\binom{k}{2}$ paths induced by a tree. The pricing problem is solved by computing $O(k^2)$ nonnegative shortest paths. Fischetti et al. [28] report on computational experiments of their algorithm applied to alignment problems, showing that using the best routing cost tree instead of the best star produces alignments that are within 6% of optimal on average.

3.1.3 Consensus Sequences

Given a set $\mathcal{S} = \{S_1, \ldots, S_k\}$ of sequences, an important problem consists in determining their *consensus* i.e., a new sequence which agrees with all the given sequences as much as possible. The consensus, in a way, represents all the sequences in the set. Let C be the consensus. Two possible objectives for C are either *min-sum* (the total distance of C from all sequences in S is minimum) or *min-max* (the maximum distance of C from any sequence in S is minimum). Both objectives are NP-hard ([30, 77]).

Assume all input sequences have length n. For most consensus problems instead of using the edit distance, the *Hamming distance* is preferable (some biological reasons for this can be found in [54]). This distance is defined for sequences of the same length, and weighs all the positions at which there is a mismatch. If C is the consensus, of length n, the distance of a sequence S from C is defined as $\sum_{i=1}^{n} \gamma(S[i], C[i])$, where $X[i]$ denotes the i–th symbol of a sequence X.

One issue with the min-sum objective is that, when the data are biased, the consensus tends to be biased as well. Consider the following situation. A biologist searches a genomic data base for all sequences similar to a newly sequenced protein. Then, he computes a consensus of these sequences to highlight their common properties. However, since genomic data bases contain mostly human sequences, the search returns human sequences more than any other species, and they tend to dominate the definition of the consensus. In this case, an unbiased consensus should optimize the min-max objective. For this problem, an approximation algorithm based on an Integer Programming formulation and Randomized Rounding has been proposed by Ben-Dor et al [11].

The Integer Programming formulation for the consensus problem has a binary variable $x_{i,\sigma}$, for every symbol $\sigma \in \Sigma$, and every position i, $1 \le i \le n$, to indicate whether $C[i] = \sigma$. The model reads:

$$\min \ r \tag{8}$$

subject to

$$\sum_{\sigma \in \Sigma} x_{i,\sigma} = 1 \qquad \forall i = 1, \dots, n \tag{9}$$

$$\sum_{i=1}^{n} \sum_{\sigma \in \Sigma} \gamma(\sigma, S_j[i])\, x_{i,\sigma} \le r \qquad \forall j = 1, \dots, k \tag{10}$$

$$x_{i,\sigma} \in \{0,1\} \qquad \forall i = 1, \dots, n,\ \forall \sigma \in \Sigma. \tag{11}$$

Let OPT denote the value of the optimum solution to the program above. An approximation to OPT can be obtained by Randomized Rounding [74]. Consider the Linear Programming relaxation of (8)-(11), obtained by replacing (11) with $x_{i,\sigma} \ge 0$. Let r^* be the LP value and $x^*_{i,\sigma}$ be the LP optimal solution, possibly fractional. An integer solution $\bar{x}$ can be obtained from x^* by *randomized rounding*: independently, at each position i, choose a letter $\sigma \in \Sigma$ (i.e., set $\bar{x}_{i,\sigma} = 1$ and $\bar{x}_{i,\tau} = 0$ for $\tau \neq \sigma$) with probability of $\bar{x}_{i,\sigma} = 1$ given by $x^*_{i,\sigma}$. Let HEUR be the value of the solution $\bar{x}$, and $\Gamma = \max_{a,b \in \Sigma} \gamma(a,b)$. The first approximation result for the consensus was given in [11], and states that, for any constant $\epsilon > 0$,

$$Pr\left(\text{HEUR} > \text{OPT} + \Gamma\ \sqrt{3\,\text{OPT}\,\log\frac{k}{\epsilon}}\right) < \epsilon. \tag{12}$$

This probabilistic algorithm can be de-randomized using standard techniques of conditional probabilities ([73]). Experiments with the exact solution of (8)-(11) are described in Gramm et al. [35].

The above IP model is utilized in a string of papers on consensus as well as other, related, problems [54, 61, 64], such as finding the *farthest* sequence (i.e., a sequence "as dissimilar as possible" from each sequence of a given set $\mathcal{S}$).

The main results obtained on these problems are PTAS based on the above IP formulation and Randomized Rounding, coupled with random sampling. In [61] and [64], Li et al. and Ma respectively, describe PTAS for finding the consensus, or a consensus substring. The main idea behind the approach is the following. Given a subset of r sequences from $\mathcal{S}$, line them up in an $r \times n$ array, and consider the positions where they all agree. Intuitively, there is a high likelihood that the consensus should also agree with them at these positions. Hence, all one needs to do is to optimize on the positions where they do not agree, which is done by LP relaxation and randomized rounding.

A PTAS for the farthest sequence problem, by Lanctot et al., is described in [54]. Also this result is obtained by LP relaxation and Randomized Rounding of an IP similar to (8)-(11).

3.2 Sequence vs Structure Alignments

Contrary to DNA molecules, which are double-stranded, RNA molecules are single-stranded and the exposed bases tend to form hydrogen bonds within the same molecule, according to Watson-Crick pairing rules. This bonds lead to structure formation, also called *secondary structure* of the RNA sequence. As an example, consider the following RNA sequence, $S =$ `UCGUGCGGUAACUUCCACGA`. Since the two ends of S are self-complementary, part of the sequence may self-hybridize, leading to the formation of a so-called *loop*, illustrated below:

```
              G G
            C     U
U C G U G           A
A G C A C           A
            C     C
              U U
```

Determining the secondary structure from the nucleotide sequence is not an easy problem. For instance, complementary bases may not hybridize if they are not sufficiently apart in the molecule. It is also possible that not all the feasible Watson-Crick pairings can be realized at the same time, and Nature chooses the most favorable ones according to an energy minimization which is not yet well understood.

All potential pairings can be represented by the edge set of a graph $G_S = (V_S, E_S)$. The vertex set $V_S = \{v_1, \ldots, v_n\}$ represents the nucleotides of S, with v_i corresponding to the i-th nucleotide. Each edge $e \in E_S$ connects two nucleotides that may possibly hybridize to each other in the secondary structure. For instance, the graph of Figure 3 describes the possible pairings for sequence S, assuming a pairing can be achieved only if the bases are at least 8 positions apart from each other. Only a subset $E'_S \subseteq E_S$ of pairings is realized by the actual secondary structure. This subset is drawn in bold in Figure 3. The graph $G'_S = (V_S, E'_S)$ describes the secondary structure. Note that each node in G'_S has degree ≤ 1.

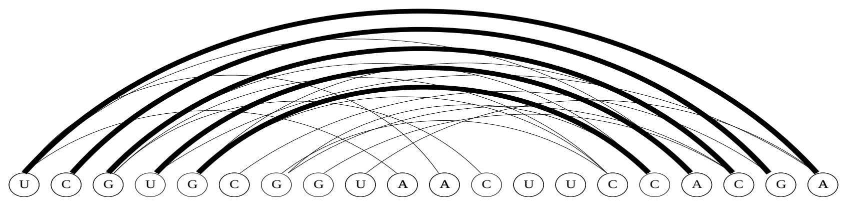


Figure 3: Graph of RNA secondary structure.

Two RNA sequences that look quite different as strings of nucleotides, may have similar secondary structure. A generic sequence alignment algorithm, such as those described in the previous section, would not spot the structure similarity, and hence a different model for comparison is needed for this case.

One possible model was proposed in Lenhof et al. [60] (see also Kececioglu et al. [50]). In this model, a sequence U of unknown secondary structure is compared to a sequence K of known secondary structure. This is done by comparing the graph $G_U = (V_U, E_U)$ to $G'_K = (V_K, E'_K)$.

Assume $V_U = [n]$ and $V_K = [m]$. An alignment of G_U and G'_K is defined by a noncrossing matching of W_{nm} (which, recalling (1), is the complete bipartite graph (V_U, V_K, L)) Given an edge $e = ij \in E_U$ and two noncrossing lines $l = [i, u] \in L$, $h = [j, v] \in L$, we say that l and h *generate* e if $uv \in E'_K$. Hence, two lines generate a possible pairing (edge of E_U) if they align its endpoints to elements that hybridize to each other in the secondary structure of K.

For each line $l = [i, j] \in L$ let w_l be the profit of aligning the two endpoints of the line (i.e., $w_l = \gamma(U[i], K[j])$). Furthermore, for each edge $e \in E_U$, let w_e be a nonnegative weight associated to the edge, measuring the "strength" of the corresponding bond. The objective of the RNA Sequence/Structure Alignment (RSA) problem is to determine an alignment which has maximum *combined* value: the value is given by the sum of the weights of the alignment lines (as for a typical alignment) and the weights of the edges in E_U that these lines generate.

Let $\prec$ be an arbitrary total order defined over the set of all lines L. Let $\mathcal{G}$ be the set of pairs of lines (p, q) with $p \prec q$, such that each pair $(p, q) \in \mathcal{G}$ generates an edge in E_U. For each $(p, q) \in \mathcal{G}$, define $w_{pq} := w_e$, where $p = [i, u]$, $q = [j, v]$ and $e = ij \in E_U$.

The Integer Programming model in [60] for this alignment problem has variables x_l for lines $l \in L$ and y_{pq} for pairs of lines $(p, q) \in \mathcal{G}$:

$$\max \sum_{l \in L} w_l\, x_l + \sum_{(p,q) \in \mathcal{G}} w_{pq}\, y_{pq} \tag{13}$$

subject to

$$x_l + x_h \le 1 \qquad \forall\, l, h \in L \mid l \text{ and } h \text{ cross} \tag{14}$$

$$y_{pq} \le x_p \qquad \forall\, (p, q) \in \mathcal{G} \tag{15}$$

$$y_{pq} \le x_q \qquad \forall\, (p, q) \in \mathcal{G} \tag{16}$$

$$x_l, y_{pq} \in \{0, 1\} \qquad \forall l \in L, \quad \forall (p, q) \in \mathcal{G}. \tag{17}$$

The model can be readily tightened as follows. First, the clique inequalities (2) can be replaced for the weaker constraints (14). Secondly, for a line $l \in L$, let $\mathcal{G}_l = \{(p, q) \in \mathcal{G} \mid p = l \vee q = l\}$. Each element (i, j) of $\mathcal{G}_l$ identifies a pairing $e \in E_U$ such that l is one of the generating lines of e. Note that if an edge $e \in E_U$ can be generated by l and $a \in L$ and, alternatively, by l and $b \ne a \in L$, then a and b must cross (in particular, they share an endpoint). Hence, of all the generating pairs in $\mathcal{G}_l$, at most one can be selected in a feasible solution. The natural generalization for constraints (15) and (16) is therefore

$$\sum_{(a,b) \in \mathcal{G}_l} y_{ab} \le x_l \qquad \forall l \in L. \tag{18}$$

These inequalities are shown in [60] to be facet-defining for the polytope

$$\operatorname{conv}\{(x, y) \,|\, (x, y) \text{ satisfies (14)-(17)}\}.$$

A Branch-and-Cut procedure based on the above inequalities is described in [60, 50]. The method was validated by aligning RNA sequences of known structure to RNA sequences of known structure, but without using the structure information for one of the two sequences. In all cases tested, the algorithm retrieved the correct alignment, as computed by hand by the biologists. For comparison, a "standard" sequence alignment algorithm (which ignores structure in both sequences) was used, and failed to retrieve the correct alignments.

In a second experiment, the method was used to align RNA sequences of unknown secondary structure, of up to 1,400 nucleotides each, to RNA sequences of known secondary structure. The optimal solutions found were compared with alternative solutions found by "standard" alignment, and their merits were discussed. The results showed that structure information is essential in retrieving biologically relevant alignments.

It must however be remarked that, in aligning large sequences, not all the variables were present in the model. In particular, there were variables only for a relatively small subset of all possible lines (which would otherwise be more than a million). This subset was determined heuristically in a pre-processing phase meant at keeping only lines that have a good chance of being in an optimal solution.

3.3 Structure vs Structure Alignments

As described in Section 2.2, a protein is a chain of molecules known as amino acids, or *residues*, which folds into a peculiar 3-D shape (called its *tertiary* structure) under several molecular forces (entropic, hydrophobic, thermodynamic). A protein's fold is perhaps the most important of all protein's features, since it determines how the protein functions and interacts with other molecules. In fact, most biological mechanisms at the protein level are based on shape-complementarity, and proteins present particular concavities and convexities that allow them to bind to each other and form complex structures, such as skin, hair and tendon.

The comparison of protein structures is a problem of paramount importance in structural genomics, and an increasing number of approaches for its solution have been proposed over the past years (see Lemmen and Lengauer [59] for a survey). Several protein structure classification servers (the most important of which is the Protein Data Bank, PDB [13]) have been designed based on them, and are extensively used in practice.

Loosely speaking, the structure comparison problem is the following: *Given two 3-D protein structures, determine how similar they are.* Some of the reasons motivating the problem are:

1. *Clustering.* Proteins can be clustered in families based on structure similarity. Proteins within a family are functionally and evolutionarily closely related.

2. *Function determination.* The function of a protein can oftentimes be determined by comparing its structure to some known ones, whose function is already

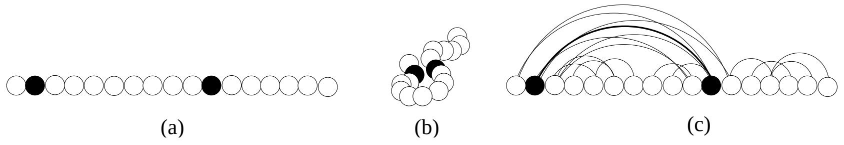


Figure 4: (a) An unfolded protein. (b) After folding. (c) The contact map graph.

understood.

3. *Fold Prediction Assessment.* This is the problem faced by the CASP (Critical Assessment of Structure Prediction [65, 66]) jurors, in a bi-annual competition in which many research groups try to predict a protein structure from its amino acid sequence. Given a set of "tentative" folds for a protein, and a correct one (determined experimentally), the jurors need to decide which guess comes closest to the true answer.

Comparing two structures implies to "align" them in some way. Since, by their nature, three-dimensional computational problems are inherently more complex than the similar one-dimensional ones, there is a dramatic difference between the complexity of two-sequences alignment and two-structures alignment. Not surprisingly, various simplified versions of the structure comparison problems were shown NP-hard [34].

Pairwise structure comparison requires a structure similarity scoring scheme that captures the biological relevance of the chemical and physical constraints involved in molecular recognition. Determining a satisfactory scoring scheme is still an open question. The most used schemes, follow mainly three themes: *RMSD (Root Mean Square Deviation) of rigid-body superposition* [47], *distance map similarity* [44] and *contact map overlap* (CMO) [32]. All these similarity measures use distances between amino acids and raise computational issues that at present do not have effective solutions.

Due to the inherent difficulty of the problem, most algorithms for structure comparison in the literature are heuristics of some sort, with a notable exception, as far as the the CMO measure is concerned. In fact, the CMO measure is the only one for which performance-guarantee approximation algorithms, as well as exact algorithms based on Mathematical Programming techniques, were proposed over the last few years. In the remainder of this subsection we will overview the main ideas underlying the exact algorithms for the CMO problem.

3.3.1 Contact Maps

A *contact map* is a 2-D representation of a 3-D protein structure. When a proteins folds, two residues that were not adjacent in the protein's linear sequence, may end up close to each other in the 3-D space (Figure 4 (a) and (b)). The contact map of a protein with n residues is defined as a 0-1, symmetric,

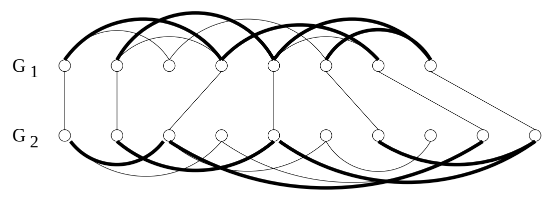


Figure 5: An alignment of value 5.

$n \times n$ matrix, whose 1–elements correspond to pairs of residues that are within a "small" distance (typically around 5Å) in the protein's fold, but are not adjacent in the linear sequence. We say that such a pair of residues are in *contact*. It is helpful to regard the contact map of a protein as the adjacency matrix of a graph G. Each residue is a node of G, and there is an edge between two nodes if the corresponding residues are in contact (see Figure 4 (c)).

The CMO problem tries to evaluate the similarity in the 3-D folds of two proteins by determining the similarity of their contact maps (the rationale being that a high contact map similarity is a good indicator of high 3-D similarity). This measure was introduced in [33], and its optimization was proved NP-hard in [34], thus justifying the use of sophisticated heuristics or enumerative methods.

Given two folded proteins, the CMO problem calls for determining an alignment between the residues of the first protein and of the second protein. The alignment specifies the residues that are considered equivalent in the two proteins. The goal is to find the alignment which highlights the largest set of common contacts. The value of an alignment is given by the number of pairs of residues in contact in the first protein which are aligned with pairs of residues that are also in contact in the second protein. Given the graph representation for contact maps, we can phrase the CMO problem in graph–theoretic language. The input consists of two undirected graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, with $V_1 = [n]$ and $V_2 = [m]$. For each edge ij, with $i < j$, we distinguish a left endpoint (i) and a right endpoint (j), and therefore we denote the edge by the ordered pair (i, j). A solution of the problem is an alignment of V_1 and V_2, i.e., a noncrossing matching $L' \subseteq L$ in W_{nm}. Two edges $e = (i, j) \in E_1$ and $f = (i', j') \in E_2$ contribute a *sharing* to the value of an alignment L' if $l = [i, i'] \in L'$ and $h = [j, j'] \in L'$. In this case, we say that l and h *generate* the sharing (e, f). The CMO problem consists in finding an alignment which maximizes the number of sharings. Figure 5 shows two contact maps and an alignment with 5 sharings.

From the compatibility of lines one can derive a notion of compatible sharings. Let $e = (i, j), e' = (i', j') \in E_1$ and $f = (u, v), f' = (u', v') \in E_2$. The sharings (e, f) and (e', f') are said *compatible* if the following lines are noncrossing: $[i, u]$, $[j, v]$, $[i', u']$ and $[j', v']$. After defining a new graph G_S (the Sharings Conflict Graph), the CMO problem can be reduced to the STABLE SET (SS) problem. In G_S, there is a node N_{ef} for each $e \in E_1$ and $f \in E_2$ and two nodes N_{ef} and $N_{e'f'}$ are connected by an edge if the sharings (e, f) and (e', f') are not compatible. A feasible set of sharings is a stable set in G_S, and the optimal solution to CMO is identified by the largest stable set in G_S.

3.3.2 Mathematical Programming approaches to the CMO problem

Attacking the CMO problem as a SS problem is not viable, since the graph G_S is very large for real proteins, even of moderate size. In fact, a small protein has about 50 residues and 100 contacts, giving rise to a graph G_S with about 10,000 nodes, beyond the capabilities of the best codes for the SS problem [46].

The most successful approaches for the CMO solution rely on formulating the problem as an Integer Program and solving it by Branch-and-Bound. In two different approaches, the bound has been obtained either from the Linear Programming relaxation [53] of an Integer Linear Programming formulation, or from a Lagrangian relaxation of a Integer Quadratic Programming formulation [20]. Both formulations are based on binary variables x_l for each line l, and y_{pq}, for pairs of lines p, q which generate a sharing.

The first formulation of the problem was given by Lancia et al. [53]. This formulation is very similar to the formulation for the RSA problem, described in Section 3.2. For completeness, we restate it here. We denote with $\mathcal{G}$ the set of all pairs of lines that generate a sharing. For each pair $(l, t) \in \mathcal{G}$, where $l = [i, i']$, and $t = [j, j']$, the edge $(i, j) \in E_1$ and the edge $(i', j') \in E_2$. For a line $l \in L$, let $\mathcal{G}_l^- = \{(p, l) \in \mathcal{G}\}$ and $\mathcal{G}_l^+ = \{(l, p) \in \mathcal{G}\}$. The model in [53] reads

$$\max \sum_{(l,t) \in \mathcal{G}} y_{lt} \tag{19}$$

subject to the clique inequalities (2) and the constraints

$$\sum_{(p,l) \in \mathcal{G}_l^-} y_{pl} \leq x_l \qquad \forall l \in L \tag{20}$$

$$\sum_{(l,p) \in \mathcal{G}_l^+} y_{lp} \leq x_l \qquad \forall l \in L \tag{21}$$

$$x_l, y_{pq} \in \{0, 1\} \qquad \forall l \in L, \quad \forall (p, q) \in \mathcal{G}. \tag{22}$$

In fact, the CMO problem can be seen as a special case of the RSA problem (13)-(17), in which each line l has weight $w_l = 0$ and each pair (p, q) of generating lines have weight $w_{pq} = 1$. Another difference between the two problems is that in the RSA problem the degree of each node in G_1 is at most 1, while for the CMO problem this is not necessarily true.

A Branch-and-Cut procedure based on the above constraints, as well as other, weaker, cuts, was used in [53] to optimally align, for the first time, real proteins from the PDB. The algorithm was run on 597 protein pairs, with sizes ranging from 64 to 72 residues. Within the time limit of 1 hour per instance, 55 problems were solved optimally and for 421 problems (70 percent) the gap between the best solution found and the upper bound was ≤ 5, thereby providing a certificate of near-optimality. The feasible solutions, whose value was used as a lower bound to prune the search tree, were obtained via heuristics such as *Genetic Algorithms* and fast *Steepest Ascent Local Search*, using several neighborhoods. The solutions found by the genetic algorithms resulted close to optimum on many runs, and in general their quality prevailed over the local search solutions.

In a second experiment, the CMO measure was validated from a biological point of view. A set of 33 proteins, known to belong to four different families, were pairwise compared. The proteins were then re-clustered according to their computed CMO value. In the new clustering, no two proteins were put in the same cluster which were not in the same family before (a 0% *false positives* rate) and only few pairs of proteins that belonged to the same family were put in different clusters (a 1.3% *false negatives* rate).

3.3.3 The Lagrangian approach

Although the approach in [53] allowed to compute, for the first time, the optimal structure alignment of real proteins from the PDB, the method had several limitations. In particular, due to the time required to solve many large and expensive LPs, the method could only be applied to problems of moderate size (proteins of up to 80 residues and 150 contacts each). To overcome these limitations, in [20] Caprara and Lancia described an alternative formulation for the problem, based on a Quadratic Programming formulation and Lagrangian Relaxation.

The theory of Lagrangian Relaxation is a well established branch of Combinatorial Optimization, and perhaps the most successful approach to tackle very large problems (such as large instances of the well known Set Covering Problem, the Optimization problem most frequently solved in real-world applications [19, 69]).

The CMO algorithm proposed in [20] is based on an approach which was successfully used for Binary Quadratic Programming problems, such as the Quadratic Assignment Problem [23]. This problem bears many similarities with the structure alignment problem. In particular, there are profits p_{ij} in the objective function which are attained when two binary variables x_i and x_j are *both* set to 1 in a solution. Analogously, in the alignment problem, there may be a profit in aligning two specific residues of the proteins *and* some other two.

For $(p, q) \in L \times L$, let b_{pq} denote a profit achieved if the lines p and q are both in the solution. Two lines p and q such that $x_p = 1$ and $x_q = 1$ contribute $b_{pq} + b_{qp}$ to the objective function. Hence, the value of an alignment can be computed as $\sum_{p \in L} \sum_{q \in L} b_{pq} x_p x_q$, provided that $b_{pq} + b_{qp} = 1$ when either $(p, q) \in \mathcal{G}$ or $(q, p) \in \mathcal{G}$, and $b_{pq} + b_{qp} = 0$ otherwise. Given such a set of profits b, the problem can be stated as

$$\max \sum_{p \in L} \sum_{q \in L} b_{pq}\, x_p\, x_q \tag{23}$$

subject to

$$\sum_{l \in Q} x_l \leq 1 \qquad \forall Q \in \text{clique}(L) \tag{24}$$

$$x_l \in \{0,1\} \qquad \forall l \in L. \tag{25}$$

Problem (23), (24), (25) is a *Binary Quadratic Program*. The formulation shows how the problem is closely related to the Quadratic Assignment Problem. The difference is that the CMO problem is in maximization form, the matching to be found does not have to be perfect and it must be noncrossing.

The objective function (23) can be linearized by introducing variables y_{pq}, for $p, q \in L$, and replacing the product $x_p x_q$ by y_{pq}. By adopting a standard procedure used in the convexification of ILPs [1, 7], a linear model for the problem is obtained as follows. Each constraint (24) associated with a set Q is multiplied by x_p for some $p \in L$ and $x_l x_p$ is replaced by y_{lp}, getting

$$\sum_{l \in Q} y_{lp} \leq x_p.$$

By doing the above for all constraints in (24) and variables x_m, $m \in L$, we get the model in [20]:

$$\max \sum_{p \in L} \sum_{q \in L} b_{pq}\, y_{pq} \tag{26}$$

subject to

$$\sum_{l \in Q} x_l \leq 1 \qquad \forall Q \in \text{clique}(L) \tag{27}$$

$$\sum_{l \in Q} y_{lp} \leq x_p \qquad \forall Q \in \text{clique}(L), \quad \forall p \in L \tag{28}$$

$$y_{pq} = y_{qp} \qquad \forall p, q \in L \,|\, p \prec q \tag{29}$$

$$x_l, y_{pq} \in \{0,1\} \qquad \forall l, p, q \in L. \tag{30}$$

The constraints (29) can be relaxed in a *Lagrangian* way, by associating a *Lagrangian multiplier* λ_{pq} to each constraint (29), and adding to the objective function (26) a linear combination of constraints (29), each weighed by its Lagrangian multiplier.

By defining for convenience $\lambda_{qp} := -\lambda_{pq}$ for $p \prec q$, and by letting $c_{pq} := b_{pq} + \lambda_{pq}$, the objective function of the Lagrangian relaxation becomes

$$U(\lambda) := \max \sum_{p \in L} \sum_{q \in L} c_{pq}\, y_{pq} \tag{31}$$

and we see that, intuitively, the effect of the Lagrangian relaxation is to redistribute the profit $b_{pq} + b_{qp}$ between the two terms in the objective function associated with y_{pq} and y_{qp}.

In [20] it is shown how the Lagrangian relaxed problem (31) subject to (27), (28) and (30) can be effectively solved, via a decomposition approach. In this problem, besides the integrality constraints (30), each variable y_{lp} appears only in the constraints (28) associated with x_p. For each $p \in L$, this implies that, if $x_p = 0$, all variables y_{lp} take the same value, whereas, if variable $x_p = 1$, the optimal choice of y_{lp} for $l \in L$ amounts to solving the following:

$$\max \sum_{l \in L} c_{lp} y_{lp} \tag{32}$$

subject to

$$\sum_{l \in Q} y_{lp} \leq 1 \qquad \forall Q \in \text{clique}(L) \,|\, p \notin Q \tag{33}$$

$$\sum_{l \in Q} y_{lp} \leq 0 \qquad \forall Q \in \text{clique}(L) \,|\, p \in Q \tag{34}$$

$$y_{lp} \in \{0, 1\} \qquad \forall l \in L. \tag{35}$$

In other words, the profit achieved if $x_p = 1$ is given by the optimal solution of (32)–(35). Let d_p denote this profit. Interpreting c_{lp} as the weight of line l and y_{lp} as the selection variable for line l, we see that the problem (32)–(35) is simply a *maximum weight noncrossing matching* problem. This problem can be solved in quadratic time by Dynamic Programming, as explained in the introduction of Section 3.

Once the profits d_p have been computed for all $p \in L$, the optimal solution to the overall Lagrangian problem can be obtained by solving one more max-weight noncrossing matching problem, in which d are the weights of the lines and x are the selection variables.

As far as the overall complexity, we have the following result:

Proposition 7 *([20]) The Lagrangian relaxation defined by (31) subject to (27), (28) and (30) can be solved in $O(|E_1||E_2|)$ time.*

Let $U(\lambda^*) := \min_\lambda U(\lambda)$ be the best upper bound that can be obtained by Lagrangian relaxation. By denoting with U_0 the upper bound obtained in [53], corresponding to the LP relaxation of (19)-(22) and (2), it can be shown that

$$U_0 \geq U(\lambda^*) \tag{36}$$

and the inequality can be tight.

In order to determine the optimal multipliers λ^*, or at least near-optimal multipliers, the approach of [20], employs a standard *subgradient optimization* procedure (see Held and Karp [43]).

The exact algorithm for CMO proposed in [20] is a Branch-and-Bound procedure based on the above Lagrangian relaxation. Furthermore, a greedy heuristic procedure is also described in [20], which performed very well in practice. The procedure constructs a solution by choosing a line which maximizes a suitable score, fixing it in the solution and iterating. As to the score, the procedure chooses the line p such that the value of the Lagrangian relaxed problem, with the additional constraint $x_p = 1$, is maximum.

In their computational experiments, Caprara and Lancia showed how the Lagrangian Relaxation approach can be orders of magnitude faster than the Branch-and-Cut approach of [53]. For instance, a set of 10,000 pairs of moderate-size proteins were aligned optimally or near-optimally in about a day. The optimal solution can be readily found if the two proteins have a similar enough structure. In this case, optimal alignments can be computed even for proteins of very large size, within few seconds. For instance, in less than one minute, the algorithm in [20] could find, for the first time, an optimal alignment of two large proteins from the same family, with about 900 residues and 2000 contacts each.

It must be remarked, however that the optimal solution of instances associated with substantially different proteins appears to be completely out of reach not only for this algorithm, but also for the other methods method currently known in Combinatorial Optimization. Such a situation is analogous to the case of the Quadratic Assignment Problem, for which instances with 40 nodes are quite far from being solved to optimality.

4 Single Nucleotide Polymorphisms

The recent whole-genome sequencing efforts [80, 45] have confirmed that the genetic makeup of humans is remarkably well conserved, with different people sharing some 99% identity at the DNA level. A DNA region whose content varies in a statistically significant way within a population is called a *genetic polymorphism*. The smallest possible region consists of a single nucleotide, and hence is called a *Single Nucleotide Polymorphism*, or SNP (pronounced "snip"). A SNP is a nucleotide site, in the middle of a DNA region which is otherwise identical for everybody, at which we observe a statistically significant variability in a population. For some reasons still unclear, the variability is restricted to only two alleles out of the four possible (`A`, `G`, `C`, `G`). The alleles can be different for different SNPs. It is believed that SNPs are the predominant form of human genetic variation [25], so that their importance cannot be overestimated for medical, drug-design, diagnostic and forensic applications.

Since DNA of diploid organisms is organized in pairs of chromosomes, for each SNP one can either be *homozygous* (same allele on both chromosomes) or *heterozygous* (different alleles). The values of a set of SNPs on a particular chromosome copy define a *haplotype*. *Haplotyping* consists in determining a pair of haplotypes, one for each copy of a given chromosome, that provides full information of the SNP fingerprint for an individual at that chromosome. In Figure 6 we give a simplistic example of a chromosome with three SNP sites. This individual is heterozygous at SNPs 1 and 3 and homozygous at SNP 2. The haplotypes are `CCA` and `GCT`.

In recent years, several optimization problems have been defined for SNP data. Some of these were tackled by Mathematical Programming techniques, which we recall in this section. Different approaches were also followed, aimed, e.g., at determining approximation algorithms, dynamic programming proce-

```
Chrom.  c, paternal:  ataggtccCtatttccaggcgcCgtatacttcgacgggActata
Chrom.  c, maternal:  ataggtccGtatttccaggcgcCgtatacttcgacgggTctata

Haplotype 1 →                 C             C               A
Haplotype 2 →                 G             C               T
```

Figure 6: A chromosome and the two haplotypes

dures, and computational complexity results. However, these approaches do not fall within the scope of this survey.

We consider the haplotyping problems relative to a single individual and to a set of individuals (a population). In the first case, the input is inconsistent haplotype data, coming from sequencing, with unavoidable sequencing errors. In the latter case, the input is ambiguous *genotype* data, which specifies only the multiplicity of each allele for each individual (i.e., it is known if individual i is homozygous or heterozygous at SNP j, for each i and j).

4.1 Haplotyping a single individual

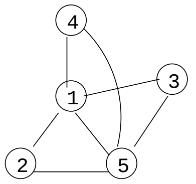
As discussed in Section 2.3, sequencing produces either single fragments or pairs of fragments (mate pairs) from the same chromosome copy. As of today, even with the best possible technology, sequencing errors are unavoidable. These consist in bases which have been miscalled or skipped altogether. Further, there can be the presence of *contaminants*, i.e. DNA coming from another organism which was wrongly mixed with the one that had to be sequenced. In this framework, the *Haplotyping Problem For an Individual* can be informally stated as: "given inconsistent haplotype data coming from fragment sequencing, find and correct the errors so as to retrieve a consistent pair of SNPs haplotypes." The mathematical framework for this problem is as follows. Independently of what the actual alleles at a SNP are, in the sequel we denote the two values that each SNP can take by the letters `A` and `B`. A haplotype, i.e. a chromosome content projected on a set of SNPs, is then a string over the alphabet $\{\mathtt{A},\mathtt{B}\}$. Let $\mathcal{S} = \{1,\ldots,n\}$ be a set of SNPs and $\mathcal{F} = \{1,\ldots,m\}$ be a set of fragments. Each SNP is covered by some of the fragments, and can take the values `A` or `B`. For a pair $(f,s) \in \mathcal{F} \times \mathcal{S}$, denote with the symbol "`-`" the fact that fragment f does not cover SNP s. Then, the the data can be represented by an $m \times n$ matrix over the alphabet $\{\mathtt{A},\mathtt{B},-\}$, called the *SNP matrix*, where each row represents a fragment and each column a SNP.

A *gapless* fragment is one covering a set of consecutive SNPs (i.e., in the row corresponding to the fragment, between any two entries in $\{\mathtt{A},\mathtt{B}\}$ there is no "`-`" entry), otherwise, the fragment has *gaps*. There can be gaps for two reasons: (i) thresholding of low-quality reads (i.e., when the sequencer cannot call a SNP `A` or `B` with enough confidence, it is marked with a `-`); (ii) mate-pairing in shotgun sequencing (one pair of mates is the same as a single fragment with one gap in

SNP

$$\begin{array}{c|cccccc} \text{fragment} & 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 1 & \mathtt{A} & \mathtt{B} & - & \mathtt{A} & \mathtt{A} & \mathtt{B} \\ 2 & \mathtt{B} & \mathtt{A} & - & - & \mathtt{B} & - \\ 3 & - & \mathtt{A} & \mathtt{B} & \mathtt{A} & \mathtt{B} & \mathtt{A} \\ 4 & - & \mathtt{A} & \mathtt{B} & - & \mathtt{B} & \mathtt{A} \\ 5 & \mathtt{B} & - & \mathtt{A} & \mathtt{B} & \mathtt{A} & - \end{array}$$

(a) M



(b) $G_{\mathcal{F}}(M)$

Figure 7: A SNP matrix M and its fragment conflict graph.

the middle).

Two fragments i and j are said to be *in conflict* if there exists a SNP k such that $M[i,k] = \mathtt{A}$ and $M[j,k] = \mathtt{B}$ or $M[i,k] = \mathtt{B}$ and $M[j,k] = \mathtt{A}$. This implies that either i and j are not from the same chromosome copy, or there are errors in the data. Given a SNP matrix M, the *fragment conflict graph* is the graph $G_{\mathcal{F}}(M) = (\mathcal{F}, E_{\mathcal{F}})$ with an edge for each pair of fragments in conflict (see Figure 7).

The data are consistent with the existence of two haplotypes (one for each chromosome copy) if and only if $G_{\mathcal{F}}(M)$ is a bipartite graph. In the presence of contaminants, or of some "bad" fragments (fragments with many sequencing errors), to correct the data one must face the problem of removing the fewest number of rows from M so that $G_{\mathcal{F}}(M)$ becomes bipartite. This problem is called the *Minimum Fragment Removal* (MFR) problem. The problem was shown to be polynomial when all fragments are gapless, by Lancia et al. [52]. When the fragments contains gaps, however, it was shown that the problem is NP-hard. In order to solve the problem in the presence of gaps [63], the following IP formulation can be adopted so as to minimize the number of fragments (nodes) removed from $G_{\mathcal{F}}(M)$ until it is bipartite. Introduce a 0-1 variable x_f for each fragment. The variables for which $x_f = 1$ in an optimal solution define the nodes to remove. Let $\mathcal{C}$ be the set of all odd cycles in $G_{\mathcal{F}}(M)$. Since a graph is bipartite if and only if there are no odd cycles, one must remove at least one node from each cycle in $\mathcal{C}$. For a cycle $C \in \mathcal{C}$, let $V(C)$ denote the nodes in C. We obtain the following Integer Programming formulation:

$$\min \sum_{f \in \mathcal{F}} x_f \tag{37}$$

s.t.

$$\sum_{f \in V(C)} x_f \geq 1 \qquad \forall C \in \mathcal{C} \tag{38}$$

$$x_f \in \{0,1\} \qquad \forall f \in \mathcal{F}. \tag{39}$$

To solve the LP relaxation of (37)-(39), one must be able to solve in polynomial time the following separation problem: Given fractional weights x^* for the fragments, find an odd cycle C, called *violated*, for which $\sum_{f \in V(C)} x^*_f < 1$.

This problem can be solved as follows. First, build a new graph $Q = (W, F)$ made, roughly speaking, of two copies of $G_{\mathcal{F}}$. For each node $i \in \mathcal{F}$ there are two nodes $i', i'' \in W$, and for each edge $ij \in E_{\mathcal{F}}$, there are two edges $i'j'', i''j' \in F$. Edges in F are weighted, with weights $w(i'j'') = w(i''j') := x^*_i/2 + x^*_j/2$. By a parity argument, a path P in Q of weight $w(P)$, from i' to i'', corresponds to an odd cycle C through i in $G_{\mathcal{F}}$, with $\sum_{j \in V(C)} x^*_j = w(P)$. Hence, the *shortest* path corresponds to the "smallest" odd cycle, and if the shortest path has weight ≥ 1 then there are no violated cycles (through node i). Otherwise, the shortest path represents a violated cycle.

To find good solutions of MFR in a short time, Lippert et al. [63] have also experimented with a heuristic procedure, based on the above LP relaxation and randomized rounding. First, the LP relaxation of (37)-(39) is solved, obtaining an optimal solution x^*. If x^* is fractional, each variable x_f is rounded to 1 with probability x^*_f. If the solution is feasible, stop. Otherwise, fix to 1 all the variables that were rounded so far (i.e., add the constraints $x_f = 1$) and iterate a new LP. This method converged very quickly on all tests on real data, with a small gap in the final solution (an upper bound to the optimum) and the first LP value (a lower bound to the optimum).

4.2 Haplotyping a population

Haplotype data is particularly sought after in the study of complex diseases (those affected by more than one gene), since it can give complete information about which set of gene alleles are inherited together. However, because polymorphism screens are conducted on large populations, in such studies it is not feasible to examine the two copies of each chromosome separately, and *genotype*, rather than haplotype, data is usually obtained. A genotype describes the multiplicity of each SNP allele for the chromosome of interest. At each SNP, three possibilities arise: either one is homozygous for the allele `A`, or homozygous for the allele `B`, or heterozygous (a situation denoted by the symbol `X`). Hence a genotype is a string over the alphabet $\{\mathtt{A}, \mathtt{B}, \mathtt{X}\}$, where each position of the letter `X` is called an *ambiguous* position. A genotype $g \in \{\mathtt{A}, \mathtt{B}, \mathtt{X}\}^n$ is *resolved* by the pair of haplotypes $h, q \in \{\mathtt{A}, \mathtt{B}\}^n$, written $g = h \oplus q$, if for each SNP j, $g[j] = \mathtt{A}$ implies $h[j] = q[j] = \mathtt{A}$, $g[j] = \mathtt{B}$ implies $h[j] = q[j] = \mathtt{B}$, and $g[j] = \mathtt{X}$ implies $h[j] \neq q[j]$. A genotype is called *ambiguous* if it has at least two ambiguous positions (a genotype with at most one ambiguous positions can be resolved uniquely). A genotype g is said to be *compatible* with a haplotype h if h agrees with g at all unambiguous positions. The following inference rule, given a genotype g and a compatible haplotype h, defines q such that $g = h \oplus q$:

Inference Rule: Given a genotype g and a compatible haplotype h, obtain a new haplotype q by setting $q[j] \neq h[j]$ at all ambiguous positions and $q[j] = h[j]$ at the remaining positions.

The *Haplotyping Problem For a Population* is the following: given a set $\mathcal{G}$ of m genotypes over n SNPs, find a set $\mathcal{H}$ of haplotypes such that each genotype is resolved by one pair of haplotypes in $\mathcal{H}$.

To turn this problem into an optimization problem, one has to specify the objective function, i.e., the cost of each solution. One such objective has been studied by Gusfield [39, 40], and is based on a greedy algorithm for haplotype inference, also known as *Clark's rule.*

This rule was was proposed by the biologist Andy Clark in 1990 [27], with arguments from theoretical population genetics in support of its validity. The goal is to derive the haplotypes in $\mathcal{H}$ by successive applications of the inference rule, starting from the set of haplotypes obtained by resolving the unambiguous genotypes (of which it is assumed here there is always at least one).

In essence, Clark proposed the following, nondeterministic, algorithm: Let $\mathcal{G}'$ be the set of non-ambiguous genotypes. Start with setting $\mathcal{G} := \mathcal{G} - \mathcal{G}'$ and $\mathcal{H}$ to the set of haplotypes obtained unambiguously from $\mathcal{G}'$. Then, repeat the following: take a $g \in \mathcal{G}$ and a compatible $h \in \mathcal{H}$ and apply the inference rule, obtaining q. Set $\mathcal{G} := \mathcal{G} - \{g\}$, $\mathcal{H} := \mathcal{H} \cup \{q\}$ and iterate. When no such g and h exist, the algorithm has succeeded if $\mathcal{G} = \emptyset$ and failed otherwise.

For example, suppose $\mathcal{G} = \{\mathtt{XAAA}, \mathtt{XXAA}, \mathtt{BBXX}\}$. The algorithm starts by setting $\mathcal{H} = \{\mathtt{AAAA}, \mathtt{BAAA}\}$ and $\mathcal{G} = \{\mathtt{XXAA}, \mathtt{BBXX}\}$. The inference rule can be used to resolve `XXAA` from `AAAA`, obtaining `BBAA`, which can, in turn, be used to resolve `BBXX`, obtaining `BBBB`. However, one could have started by using `BAAA` to resolve `XXAA` obtaining `ABAA`. At that point, there would be no way to resolve `BBXX`. The non-determinism in the choice of the pair g, h to which we apply the inference rule can be settled by fixing a deterministic rule based on the initial sorting of the data. Clark in [27] used a large (but tiny with respect to the total number of possibilities) set of random initial sortings to run the greedy algorithm on real and simulated data sets, and report the best solution overall. To find the best possible order of application of the inference rule, Gusfield considered the following optimization problem: find the ordering of application of the inference rule that leaves the fewest number of unresolved genotypes in the end. Gusfield showed the problem to be APX-hard in [40].

As for practical algorithms, Gusfield [39, 40] proposed a graph-theoretic formulation of the problem and an Integer Programming approach for its solution. The problem is first transformed (by an exponential-time reduction) into a problem on a digraph $G = (N, A)$, defined as follows. Let $N = \bigcup_{g \in \mathcal{G}} N(g)$, where $N(g) := \{(h, g) \,|\, h \text{ is compatible with } g\}$. $N(g)$ is (isomorphic to) the set of possible haplotypes obtainable by setting each ambiguous position of a genotype g to one of the 2 possible values. Let $N' = \bigcup_{g \in \mathcal{G}'} N(g)$ be (isomorphic to) the subset of haplotypes unambiguously determined from the set $\mathcal{G}'$ of unambiguous genotypes. For each pair $v = (h, g')$, $w = (q, g)$ in N, there is an arc $(v, w) \in A$ iff g is ambiguous, $g' \neq g$ and $g = h \oplus q$ (i.e., q can be inferred from g via h). Then, any directed tree rooted at a node $v \in N'$ specifies a feasible history of successive applications of the inference rule starting at node $v \in N'$. The problem can then be stated as: Find the largest number of nodes in $N - N'$ that can be reached by a set of node–disjoint directed trees, where each tree is rooted at a node in N' and where for every ambiguous genotype g, *at most* one node in $N(g)$ is reached.

The above graph problem was also shown to be NP-hard [40] (note that the reduction of the haplotyping problem to this one is exponential–time, and hence it does not imply NP–hardness trivially). For its solution Gusfield proposed the following formulation. Let x_v be a 0-1 variable associated to node $v \in N$.

$$\max \sum_{v \in N - N'} x_v$$

subject to

$$\sum_{v \in N(g)} x_v \leq 1 \qquad \forall g \in \mathcal{G} - \mathcal{G}'$$

$$x_w \leq \sum_{v \in \delta^-(w)} x_v \qquad \forall w \in N - N'$$

with $x_v \in \{0,1\}$ for $v \in N$.

Gusfield [39] used this model to solve the Haplotyping problem. To reduce the problem dimension and make the algorithm practical, he actually defined variables only for nodes in a subgraph of G, i.e. the nodes reachable from nodes in N'. He observed that the LP solution, on real and simulated data, was almost always integer, thus requiring no branching in the branch and bound search. He also pointed out that, although there is a possibility of an integer solution containing directed cycles, this situation never occurred in the experiments, and hence he did not need to add *subtour elimination*-type inequalities to the model.

5 Genome Rearrangements

Thanks to the large amount of genomic data that has become available in the past years, it is now possible to compare the genomes of different species, in order to find their differences and similarities. This is a very important problem because, when developing new drugs, we typically test them on animals before humans. This motivates questions such as determining, e.g., how close to a human a mouse is, or, in a way, how much evolution separates us from mice.

In principle, a genome can be thought of as a (very long) sequence and hence one may want to compare two genomes via a sequence alignment algorithm. Besides the large time needed to obtain an optimal alignment, there is a biological reason why sequence alignment is not the right model for large-scale genome comparisons. In fact, at the genome level, differences should be measured not in terms of insertions/deletions/mutations of single nucleotides, but rather rearrangements (misplacements) of long DNA regions which occurred during evolution.

Among the main evolutionary events known are *inversions*, *transpositions*, and *translocations*. Each of these events affects a long fragment of DNA on a chromosome. When an inversion or a transposition occurs, a DNA fragment is detached from its original position and then is reinserted, on the same chromosome. In an *inversion*, it is reinserted at the same place, but with opposite

```
ATTGTTataggttagAATTG   ATTgtttataGGCTAGATCCGCCAGA   CTGGATgcaggcat    TCATTGAaata
         ↓                           ↓                               ↓
ATTGTTgattggataAATTG   ATTGGCTAGATCCGCgtttataCAGA   CTGGATaata    TCATTGAgcaggcat

     (Inversion)                (Transposition)                (Translocation)
```

Figure 8: Evolutionary events

orientation than it originally had. In a *transposition*, it keeps the original orientation but ends up in a different position. A *translocation* causes a pair of fragments to be exchanged between the ends of two chromosomes. Figure 8 illustrates these events, where each string represents a chromosome.

Since evolutionary events affect long DNA regions (several thousand bases) the basic unit for comparison is not the nucleotide, but rather the *gene*.

The general *Genome Comparison Problem* can be informally stated as: Given two genomes (represented by two sets of ordered lists of genes, one list per chromosome) find a sequence of evolutionary events that, applied to the first genome, turn it into the second. Under a common parsimony principle, the solution sought is the one requiring the minimum possible number of events. Pioneering work in the definition of genome rearrangement problems is mainly due to Sankoff and colleagues [76].

In the past decade, people have concentrated on evolution by means of some specific event alone, and have shown that even these special cases can be already very hard to solve. Since inversions are considered the predominant of all types of rearrangements, they have received the most attention. For historical reasons, they have become known as *reversals* in the computer science community.

In the remainder of this section, we will outline a successful Mathematical Programming approach for the solution of computing the evolutionary distance between two genomes evolved by reversals only, a problem known as *Sorting by Reversals*. The approach is due to Caprara et al [21, 22].

Two genomes are compared by looking at their common genes. After numbering each of n common genes with a unique label in $\{1, \dots, n\}$, each genome is a permutation of the elements $\{1, \dots, n\}$ (we assume here to focus on a single, specific chromosome). Let $\pi = (\pi_1 \ \dots \ \pi_n)$ and $\sigma = (\sigma_1 \ \dots \ \sigma_n)$ be two genomes. By possibly relabeling the genes, we can always assume that $\sigma = \iota := (1\ 2\ \dots\ n)$, the identity permutation. Hence, the problem becomes turning π into ι, i.e., *sorting* π.

A *reversal* is a permutation ρ_{ij}, with $1 \le i < j \le n$, defined as

$$\rho_{ij} = (1 \ \dots \ i-1 \quad \underbrace{\boxed{j \; j-1 \ \dots \ i+1 \; i}}_{\text{reversed}} \quad j+1 \ \dots \ n).$$

By applying ρ_{ij} to π, one obtains $(\pi_1 \dots \pi_{i-1},\ \pi_j\ \pi_{j-1} \dots \pi_i,\ \pi_{j+1} \dots \pi_n)$, i.e., the order of the elements $\pi_i, \dots, \pi_j$ has been reversed. Let $\mathcal{R} = \{\rho_{ij}, 1 \le i <$

$j \leq n\}$. Each permutation π can be expressed (non-uniquely) as a product $\iota\rho^1\rho^2 \ldots \rho^D$ with $\rho^i \in \mathcal{R}$ for $i = 1 \ldots D$. The minimum value D such that $\iota\rho^1\rho^2 \ldots \rho^D = \pi$ is called the *reversal distance* of π, and denoted by $d(\pi)$. *Sorting by Reversals* (SBR) is the problem of finding $d(\pi)$ and a sequence $\rho^1\rho^2 \ldots \rho^{d(\pi)}$ satisfying the previous equation.

The first optimization algorithm for computing the reversal distance was a branch-and-bound procedure, due to Kececioglu and Sankoff [51], based on a combinatorial bound. This algorithm is only suitable for small problems ($n \leq 30$), and becomes quickly impractical for larger values of n (note that, as of this writing, the data for real-life genome comparison instances involve a number n of common genes in the range $50 \leq n \leq 200$). The algorithm was developed at a time when the complexity of the problem was still unknown, although it was conjectured that SBR was NP-hard. Settling the complexity of SBR became a longstanding open question, eventually answered by Caprara [18] who showed that in fact, SBR is NP-hard.

A major step towards the practical solution of the problem was made by Bafna and Pevzner [6], who found a nice combinatorial characterization of π in terms of its breakpoints. A *breakpoint* is given by a pair of adjacent elements in π that are not adjacent in ι — that is, there is a breakpoint at position i, if $|\pi_i - \pi_{i-1}| > 1$. Let $b(\pi)$ denote the number of breakpoints. A trivial bound is $d(\pi) \geq \lceil b(\pi)/2 \rceil$, since a reversal can remove at most two breakpoints, and ι has no breakpoints. However, Bafna and Pevzner showed how to obtain a much better bound from the *breakpoint graph* $G(\pi)$. $G(\pi)$ has a node for each element of π and edges of two colors, say red and blue. There is a red edge between π_i and π_{i-1} for each position i at which there is a breakpoint, and a blue edge between each h and k such that $|h - k| = 1$ and h, k are not adjacent in π. Each node has the same number of red and blue edges incident on it (0, 1 or 2), and $G(\pi)$ can be decomposed into a set of edge-disjoint color-alternating cycles. Note that this cycles are not necessarily simple, i.e., they can repeat nodes. Let $c(\pi)$ be the maximum number of edge-disjoint alternating cycles in $G(\pi)$. Bafna and Pevzner [6] proved the following

Theorem 8 *([6]) For every permutation π, $d(\pi) \geq b(\pi) - c(\pi)$.*

The lower bound $b(\pi) - c(\pi)$ turns out to be very tight, as observed first experimentally by various authors and then proved to be almost always the case by Caprara [17], who showed in [18] that determining $c(\pi)$ is essentially the same problem as determining $d(\pi)$, and hence, NP-hard as well.

Since computing $c(\pi)$ is hard, the lower bound $b(\pi) - c(\pi)$ should not be used directly in a branch-and-bound procedure for SBR. However, for any upper bound $c'(\pi)$ to $c(\pi)$, also the value $b(\pi) - c'(\pi)$ is a lower bound to $d(\pi)$, which may be quite a bit easier to compute. Given an effective Integer Linear Programming (ILP) formulation to find $c(\pi)$, a good upper bound to $c(\pi)$ can be obtained by LP relaxation. Let $\mathcal{C}$ denote the set of all the alternating cycles of $G(\pi) = (V, E)$, and for each $C \in \mathcal{C}$ define a binary variable x_C. The following is the ILP formulation of the maximum cycle decomposition:

$$c(\pi) := \max \sum_{C \in \mathcal{C}} x_C \tag{40}$$

subject to

$$\sum_{C \ni e} x_C \leq 1 \qquad \forall e \in E \tag{41}$$

$$x_C \in \{0,1\} \qquad \forall C \in \mathcal{C}. \tag{42}$$

The LP relaxation to the above problem is obtained by replacing the constraints $x_C \in \{0,1\}$ with $x_C \geq 0, \forall C \in \mathcal{C}$ (it is easy to see that the constraints $x_C \leq 1$ can be omitted). Caprara et al showed in [21] how to solve the exponentially large LP relaxation of (40)-(42) in polynomial time by column-generation techniques. To price-in the variables, the solution of some non-bipartite min-cost perfect matching problems is required.

Consider the dual of the LP relaxation of (40)-(42), which reads

$$c'(\pi) := \min \sum_{e \in E} y_e \tag{43}$$

subject to

$$\sum_{e \in C} y_e \geq 1 \qquad \forall C \in \mathcal{C} \tag{44}$$

$$y_e \geq 0, \quad \forall e \in E. \tag{45}$$

The separation problem for (43)-(45) is the following: given a weight y_e^* for each $e \in E$, find an alternating cycle $C \in \mathcal{C}$ such that

$$\sum_{e \in C} y_e^* < 1, \tag{46}$$

or prove that none exists. Call an alternating cycle satisfying (46) a *violated cycle*. To solve (43)-(45) in polynomial time, one must be able to identify a violated cycle in polynomial time, i.e., an alternating cycle of $G(\pi)$ having weight < 1, where each edge $e \in E$ is given the weight y_e^*.

We illustrate the idea for doing this, under the simplifying assumption that the decomposition contains only cycles that are *simple* (i.e., they do not go through the same node twice). Consider the following construction, analogous to one used by Grötschel and Pulleyblank for finding minimum-weight odd and even paths in an undirected graph [36]. Define the graph H, depending on $G(\pi)$, with $H = (V_R \cup V_B, L_R \cup L_B \cup L_V)$, as follows. For each node i of $G(\pi)$, H contains two *twin* nodes $i_R \in V_R, i_B \in V_B$. For each red edge ij in $G(\pi)$, H has an edge $i_R j_R \in L_R$; for each blue edge ij in $G(\pi)$, H has an edge $i_B j_B \in L_B$. Each edge in $L_R \cup L_B$ is given the same weight as its counterpart in $G(\pi)$. Finally twin nodes are connected in H by means of edges $i_R i_B \in L_V$, for each $i \in V$, each having weight 0.

The following proposition justifies the use of H.

Proposition 9 *([21]) There is a bijection between perfect matchings of H and (possibly empty) sets of node-disjoint simple alternating cycles of $G(\pi)$.*

From this proposition it follows

Proposition 10 *There is a simple alternating cycle in G of weight < 1 if and only if there exists a perfect matching $M \neq L_V$ in H of weight < 1.*

Accounting for alternating cycles that are not simple requires to first break up the nodes of degree 4 in G into pairs of nodes, and then apply the above reduction to the resulting graph (technical details can be found in [21]). Note that the graph H obtained is, in general, non-bipartite.

Although polynomial, the solution of the weighted matching problem on general graphs is computationally expensive to find. In [22] Caprara et al. describe a weaker bound, obtained by enlarging the set $\mathcal{C}$ in (40)-(42) to include also the *pseudo-alternating* cycles, i.e., cycles that alternate red and blue edges but may possibly use an edge twice. With this new set of variables, the pricing becomes much faster, since only *bipartite* matching problems must be solved.

Let $\mathcal{P}$ be the set of pseudo-alternating cycles. Similarly to before, one can now formulate the problem of finding the decomposition of $G(\pi)$ into a set of edge-disjoint pseudo-alternating cycles. The associated column generation problem requires finding, given a weighting u^* for the edges, a pseudo-alternating cycle $C \in \mathcal{P}$ of weight $u^*(C) < 1$. The weight of C is the sum of the weights of its edges, where each edge e which is traversed twice by the cycle, contributes $2u^*(e)$ to the sum. We next show how to solve this problem.

Construct an arc-weighted directed graph $D = (V, A)$ from $G(\pi)$ and u^* as follows. D has the same node set as $G(\pi)$ and, for each node pair $i, j \in V$, D has an arc $(i, j) \in A$ if there exists $k \in V$ such that ik is a blue edge of $G(\pi)$ and kj is a red edge. The arc weight for (i, j) is given by $u^*_{ik} + u^*_{kj}$ (if there exist two such k, consider the one yielding the minimum weight of arc (i, j)). Call *dicycle* a simple (i.e. without node repetitions) directed cycle of D. Then, each dicycle of D corresponds to a pseudo-alternating cycle of $G(\pi)$ of the same weight. Vice versa, each pseudo-alternating cycle of $G(\pi)$ corresponds to a dicycle of D whose weight is not larger.

It follows that, for a given u^*, $G(\pi)$ contains a pseudo-alternating cycle $C \in \mathcal{P}$ of weight < 1 if and only if D contains a dicycle of weight < 1.

To find such a cycle, introduce *loops* in D, i.e. arcs of the form (i, i) for all $i \in V$, where weight of the loops is initially set to 0. The *Assignment Problem* (i.e., the perfect matching problem on bipartite graphs) can then be used to find a set of dicycles in D, spanning all nodes, and with minimum total weight.

From this proposition it follows

Proposition 11 *There is a pseudo-alternating cycle in $G(\pi)$ of weight < 1 if and only there is a solution to the assignment problem on D not containing only loops and with weight < 1.*

Although weaker than the bound based on alternating cycles, the bound based on pseudo-alternating cycles turns out to be still very tight. Based on this bound, Caprara et al. developed a branch-and-price algorithm [22] which can routinely solve, in a matter of seconds, instances with $n = 200$ elements, a large enough size for all real-life instances available so far. Note that no effective ILP formulation has ever been found for modeling SBR directly. Finding such a formulation constitutes an interesting theoretical problem.

With this branch and price algorithm, SBR is regarded as practically solved, being one of the few NP-hard problems for which a (worst-case) exponential algorithm is fairly good on most instances.

We end this section by mentioning a closely related problem, for which an elegant and effective solution has been found by Hannenhalli and Pevzner. However, this solution does not employ Mathematical Programming techniques and hence falls only marginally within the scope of this survey.

Since a DNA molecule has associated a concept of “reading direction” (from the so-called *5'-end* to the *3'-end*), after a reversal, not only the order of some genes is inverted, but also of the nucleotide sequence of each such gene. To account for this situation, a genome can be represented by a *signed* permutation, i.e. a permutation in which each element is signed either '+' or '−'. For a signed permutation, the effect of a reversal is not only to flip the order of some consecutive elements, but also to complement their sign. For instance, the reversal ρ_{24} applied to the signed permutation $(+1\ -4\ +3\ -5\ +2)$ yields the signed permutation $(+1\ +5\ -3\ +4\ +2)$. The *signed* version of Sorting by Reversals (SSBR) consists in determining a minimum number of reversals that turn a signed permutation π into $(+1\ +2\ \ldots\ +n)$.

With a deep combinatorial analysis of the cycle decomposition problem for the permutation graph of a signed π, SSBR was shown to be polynomial, by Hannenhalli and Pevzner [42]. The original $O(n^4)$ algorithm of Hannenhalli and Pevzner for SSBR was improved over the years to an $O(n^2)$ algorithm for finding the optimal solution [48] and an $O(n)$ algorithm [5] for finding the optimal value only (i.e., the signed reversal distance).

6 Genomic Mapping and the TSP

In the last years (especially prior to the completion of the sequencing of the human genome), a great deal of effort has been devoted to determining *genomic maps*, of various detail.

Similarly to a geographic map, the purpose of a genomic map is to indicate the location of some landmarks in a genome. Each landmark (also called a *marker*) corresponds to a specific DNA region, whose content may or may not be known. Typical markers are genes, or short DNA sequences of known content (called STSs, for Sequence Tagged Sites), or restriction-enzyme sites (which are short, palindrome, DNA sequences).

At the highest resolution, to locate a marker one needs to specify the chromosome on which it appears and the distance in base pairs, from one specific end of the chromosome. A map of lower resolution, instead of the absolute position of the markers, may give their relative positions, e.g., in the form of the order with which the markers appear on a chromosome.

The construction of genomic maps gives us the opportunity to describe a Computational Biology problem for which the solution proposed consists in a reduction to a well-known optimization problem, in this case the TSP. In particular, we will discuss the *mapping* problem for either *clones vs. probes* or *radiation hybrids vs. markers*. For both problems, the input data consists of a 0-1, $m \times n$, matrix M.

In the first problem considered, there are m *clones* and n *probes*. Both clones and probes are DNA fragments, but of quite different length. In particular, a clone is a long DNA fragment (several thousand bases), from a given chromosome, while a probe is a short DNA fragment (a few bases), also from the same chromosome. An *hybridization* experiment can tell, for each clone i and probe j, if j is contained in i. The experiment is based on the Watson-Crick pairing rules, according to which j finds its complement in i, if present, and forms strong bonds with it. By attaching to it a fluorescent label, it is possible to determine if j has hybridized with its complement in i.

The results of the experiments are stored in the 0-1 matrix M, which can be interpreted as follows: $M_{ij} = 1$ if and only if probe j appears in clone i. Due to experimental errors, some clones may be *chimeric*. A chimeric clone consists of two or more distinct fragments from the chromosome instead of only one. Other experimental errors are *false positives* ($M_{ij} = 1$ but i does not contain j) and *false negatives* ($M_{ij} = 0$ but i contains j). The *Probe Mapping Problem* consists in: given a clone vs probe matrix M, determine in which order the probes occur on the chromosome. The solution corresponds to a permutation of the columns of M. Note that, when there are no chimeric clones and no false positive/negatives, the correct ordering of the probes is such that, after rearranging the columns, in each clone (row of M) the elements "1" appear consecutively. Such a matrix is said to have the *consecutive 1 property*. Testing if a matrix has the consecutive 1 property, and finding the order that puts the 1s consecutively in each row, is a well known polynomial problem [16].

Under the assumption of possible errors in the data, the problem of determining the correct probe order is NP-hard, and Alizadeh et al. [3] have described an optimization model in which the best solution corresponds to the order minimizing

$$v(M') = \sum_{j=1}^{n-1} c(M'_j, M'_{j+1}). \tag{47}$$

Here, M' is a matrix obtained by permuting the columns of M, M'_k denotes the k-th column of M' and $c(x, y)$ is the cost of following a column x with a column y, both viewed as binary vectors. Assuming there are no false positive/negatives, and that there is a low probability for a clone to be chimeric, $c(x, y)$ can be taken as the Hamming distance of x and y. Then, the objective becomes that of minimizing the total number of times that a block of consecutive 1s is followed by some 0s and then by another block of consecutive 1s, on the same row (in which case, the row would correspond to a chimeric clone).

It is easy to see that the optimization of $v(M')$ is a TSP problem (in fact, a *Shortest Hamiltonian Path* problem). In their code, Alizadeh et al. used some standard TSP heuristics, such as 2-OPT and 3-OPT neighborhood search, and a Simulated Annealing approach. The procedure was tested on simulated data, for which the correct solution was known, as well as real data for chromosome 21. The experiments showed that, for several cost functions c, depending on the simulation model adopted, the TSP solutions approximate very well the correct solutions.

A very similar problem concerns the construction of *Radiation Hybrid* (RH) maps. Radiation hybrids are obtained by (i) breaking, by radiation, a human chromosome into several random fragments (the markers) (ii) fusing the radiated cells with rodent normal cells. The resulting hybrid cells, may retain one, none or many of the fragments from step (i).

For each marker j and each radiation hybrid i, an experiment can tell if j was retained by i or not. Hence, the data for m radiation hybrids and n markers can be represented by a 0-1 matrix M. As before, the problem consists in ordering the markers, given M. The problem is studied in Ben Dor et al. [9, 10]. Two different measures are widely used to evaluate the quality of a possible solution. The first is a combinatorial measure, called Obligate Chromosome Breaks (OCB), while the second is a statistically based, parametric method of maximum likelihood estimation (MLE). For a solution π, let M' be the matrix obtained from M after rearranging the columns according to π. The first measure is simply given by the number of times that, in some row of M', a 0 is followed by a 1, or vice versa. As we already saw, the objective of minimizing this measure is achieved by minimizing $v(M')$, defined in (47), when $c(x, y)$ is the Hamming distance.

In the MLE approach, the goal is to find a permutation of the markers and estimate the distance (breakage probability) between adjacent markers, such that the likelihood of the resulting map is maximized. The likelihood of a map (consisting of a permutation of the markers, and distances between them) is the probability of observing the RH data, given the map.

The reduction of MLE to TSP is carried out in three steps, as described in Ben Dor [8]. First, the retention probability is estimated. Then, the breakage probability between each pair of markers is estimated. Finally, the costs $c(x, y)$ are defined such that the total value of (47) is equal to the negative logarithm of the likelihood of the corresponding permutation. Therefore, minimizing $v(M')$ is equivalent to maximizing the logarithm of the likelihood, and hence, to maximizing the likelihood.

Although the OCB and MLE approaches are different, it can be shown that, under the assumption of evenly spaced markers, these two criteria are roughly equivalent. As a result, in this case it is enough to optimize the OCB objective, and the order obtained will optimize MLE as well. However, if the assumption is not met, the two objectives differ. At any rate, the optimization of both objectives corresponds to the solution of a TSP problem. In their work, Ben Dor at al. [9, 10] used *Simulated Annealing*, with the 2-OPT neighborhood structure of Lin and Kernighan [62], for the solution of the TSP.

The software package RHO (Radiation Hybrid Optimization), developed by Ben Dor et al., was used to compute RH maps for the 23 human chromosomes. For 18 out of 23 chromosomes, the maps computed with RHO were identical or nearly identical to the the corresponding maps computed at the Whitehead Institute, by experimental, biological, techniques. For the remaining 5 chromosomes, the maps computed by using RHO were superior to the Whitehead maps with respect to both optimization criteria [10].

In [2], Agarwala et al. describe an improvement of the RHO program, obtained by replacing the simulated annealing module for the TSP with the state-of-the-art TSP software package, CONCORDE [4]. In a validating experiment, public data (i.e., the panels Genebridge 4 and Stanford G3) were gathered from a radiation hybrid databank, and new maps were computed by the program. The quality of a map was accessed by comparing how many times two markers, that appear in some order in a published sequence deposited on GenBank, are put in the opposite order by the solution (i.e., the map is inconsistent with the sequence). It was shown that the public maps for Genebridge 4 and G3 were inconsistent for at least 50% of the markers, while the maps computed by the program were far more consistent.

We end this section by mentioning one more mapping problem, the *Physical Mapping with end-probes*, for which a Branch-and-Cut approach was proposed by Christof et al. [26]. The problem has a similar definition that the problems discussed above. The main difference is that some probes are known to be *end-probes*, i.e., coming from the ends of some clone.

7 Applications of Set Covering

Together with the TSP, the Set Covering (SC) problem is perhaps the only other notorious Combinatorial Optimization problem to which a large number of Computational Biology problems have been reduced. In this section we will review a few applications of this problem to areas such as *primer design* for PCR reactions and *tissue classification in microarrays.*

As described in Section 2.3, PCR is an experiment by which a DNA region can be amplified several thousand of times. In order to do so, two short DNA fragments are needed, one which must precede, and the other which must follow, the region to be amplified. These fragments are called *primers*, and their synthesis is carried on by specialized laboratories. The preparation of primers is delicate and expensive and hence, in the past years, an optimization problem concerning primer design has emerged. The goal is to determine the least expensive set of primers needed to perform a given set of reactions. The problem, in its most simplified version, can be described as follows.

Given a set $\mathcal{S} = \{S_1, \ldots, S_n\}$ of strings (DNA sequences), find a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of pairs of strings, where each $p_i = (s_i, e_i)$. For each $i \in \{1, \ldots, n\}$, s_i and e_i must be substrings of S_i, of length at least k (a given input parameter). The objective requires to minimize $|\mathcal{P}|$.

The problem can be readily recognized as a SC, where the strings in $\mathcal{S}$ are the elements to be covered. Let $\mathcal{A}$ be the set of all substrings of length k. Viewing each pair $(a_i, a_j) \in \mathcal{A} \times \mathcal{A}$ as a set A_{ij}, consisting of all strings which contain both a_i and a_j, the primer design problem corresponds the SC problem with sets A_{ij} and universe $\mathcal{S}$.

A heuristic approach to this problem was proposed by Pearson et al. [71] who, instead of solving once the SC described above, solved two times a different SC, to find both ends of each primer-pair. In their SC reduction, each element $a \in \mathcal{A}$ is viewed as a set, consisting of all strings which contain it. One of the main uses of PCR is that it provides an effective membership test for a sequence (e.g., a protein) and a certain family. The idea is that, given a set of primers designed for a family $\mathcal{F}$ and a new sequence s, if one of the primer-pairs amplifies part of s, then s belongs to $\mathcal{F}$, and otherwise it does not. In many cases (e.g., by performing a multiple alignment of the known family sequences) it is possible to find a short sequence common to all the elements of $\mathcal{F}$ (but perhaps also to elements not in $\mathcal{F}$), which can be used as one of the primers in each primer-pair. In this case, the SC reduction described by Pearson et al. correctly minimizes the total number of primers to synthesize. Finally, a heuristic, greedy, approach to minimize the number of primers to use in multiplex PCR reactions was employed by Nicodeme and Steyaert [70].

A second application of the SC problem arises in the context of microarrays, also called *DNA chips*. A microarray is a device which allows for a massively parallel experiment, in which several samples (targets) are probed at once. The experiment can determine, for each sample, the amount of production of mRNA for each of a given set of genes. The production of mRNA is directly proportional to the level of expression of a gene. In an application of this technology, one may compare samples from healthy cells and cancer cells, and highlight an excess in the expression of some genes, which may hint at the cause of the disease.

Physically, a microarray is an array on a chip, in which the rows (up to some hundreds) are indexed by a set of samples (e.g. tissues), while the columns (some thousands) are indexed by a set of genes. To each cell of the chip are attached a large number of identical short DNA sequences (probes). The probes along each column (corresponding to a gene g) are designed specifically to bind (under Watson-Crick complementarity) to the mRNA sequence encoded by the gene g.

After letting a set of fluorescently-labeled samples hybridize to each of the chip cells, an image processing software detects the amount of DNA that has been "captured" in each cell. The result is a matrix M of numbers, where $M[s, g]$ represents the amount of expression of gene g in sample s. The matrix is usually made binary by using a threshold, so that $M[s, g] = 1$ if and only if a gene g is expressed (at a sufficiently high level) by a sample s.

Given a binary matrix M containing the results of microarray hybridization, several optimization problems have been studied which aim at retrieving useful information from M. One such problem concers the use of gene expression to distinguish between several types of tissues (e.g., brain cells, liver cells, etc.). The objective is to find a subset G of the genes such that, for any two tissues $s \neq s'$, there exists at least a gene $g \in G$ with $M[s, g] \neq M[s', g]$, and the cardinality of G is minimum. This problem is equivalent to a well known NP-hard problem, the Minimum Test Collection (problem [SP96], Garey and Johnson [31]), and has been studied by Halldorsson et al. and De Bontridder et al.[41, 15, 14]. In order to solve the problem, Haldorsson et al. propose the following reduction to the SC. Each pair of tissues (s, s') is an element of the universe set, and each gene g corresponds to the subset of those pairs (s, s') for which $M[s, g] \neq M[s', g]$.

A variant of this problems, that is also of interest to Computational Biology, is the following: the input tissues are partitioned in k classes, and the problem consists again in finding a minimum subset of genes G such that, whenever two samples s and s' are not from the same class, there is a gene $g \in G$ with $M[s, g] \neq M[s', g]$. The special case of $k = 2$ is very important in the following situation: Given a set of healthy cells and a set of tumor cells, we want to find a set of genes G such that there is always always a gene $g \in G$ whose expression level can distinguish any healthy sample from any tumor one. The knowledge of the expression levels for such a set of genes would constitute a valid diagnostic test to determine the presence/absence of a tumor in a new input sample.

8 Conclusions

In this survey we have reviewed some of the most successful Mathematical Programming approaches to Computational Biology problems of the last few years. The results described show, once more, how profitably the sophisticated optimization techniques of Operations Research can be applied to a non-mathematical domain, provided the difficult phase of problem modeling has been successfully overcome. Clearly, the help of a biology expert is vitally needed in the modeling phase. However, we cannot stress hardly enough how important it is for the optimization expert to learn the language of molecular biology, in order to become a more active player during the critical problem-analysis phase.

Although this survey has touched many problems from different application areas, the number of new results which employ Mathematical Programming techniques is steadily increasing, and it is easy to foresee applications of these techniques to more and more new domains of Computational Biology. For instance, very recently a Branch-and-Cut approach was proposed for the solution of a protein fold prediction problem. The approach, by Xu et al. [86], tries to determine the best "wrapping"(called *threading*) of a protein sequence to a known 3-D protein structure, and bears many similarities to the models described in Section 3.2 and Section 3.3 for structure and sequence alignments.

Finally, besides the problems described in this survey, we would like to mention that Mathematical Programming techniques have been also successfully applied to related problems, arising in clinical biology and medicine. Some of these problems are concerned with the optimization of an instrument's use, or the speed-up of an experiment. Among the most effective approaches for this class of problems, we recall the works of Eva Lee et al., who developed Integer Programming models for the optimal schedule of clinical trials, and for the radiation treatment of prostate cancer [58, 56, 55, 57].

References

[1] W.P. Adams and H.D. Sherali. A tight linearization and an algorithm for zero-one quadratic programming problems. *Management Science*, 32:1274–1290, 1986.

[2] R. Agarwala, D. Applegate, D. Maglott, G. Schuler, and A. Schaffer. A fast and scalable radiation hybrid map construction and integration strategy. *Genome Research*, 10:230–364, 2000.

[3] F. Alizadeh, R. Karp, D. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 489–500, New York, NY, 1994. ACM press.

[4] D. Applegate, R. Bixby, V. Chvatal, and W.Cook, editors. *CONCORDE: A code for solving Traveling Salesman Problems.* World Wide Web, `http://www.math.princeton.edu/tsp/concorde.html`.

[5] D. A. Bader, B. M. Moret, and M. Yan. A linear-time algorithm for computing inversion distances between signed permutations with an experimental study. *Journal of Computational Biology*, 8(5):483–491, 2001.

[6] V. Bafna and P.A. Pevzner. Genome rearrangements and sorting by reversals. *SIAM Journal on Computing*, 25:272–289, 1996.

[7] E. Balas, S. Ceria, and G. Cornuejols. A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993.

[8] A. Ben-Dor. *Constructing Radiation Hybrid Maps of the Human Genome.* PhD thesis, Technion, Israel Institute of Technology, Haifa, 1997.

[9] A. Ben-Dor and B. Chor. On constructing radiation hybrid maps. *Journal of Computational Biology*, 4:517–533, 1997.

[10] A. Ben-Dor, B. Chor, and D. Pelleg. Rho—Radiation Hybrid Ordering. *Genome Research*, 10:365–378, 2000.

[11] A. Ben-Dor, G. Lancia, J. Perone, and R. Ravi. Banishing bias from consensus sequences. In *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 1264 of *Lecture Notes in Computer Science*, pages 247–261. Springer, 1997.

[12] D. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, B. Rapp, and D. L. Wheeler DL. Genbank. *Nucleic Acids Research*, 30(1):17–20, 2002.

[13] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000. The PDB is at `http://www.rcsb.org/pdb/`.

[14] K. M. De Bontridder, B. Halldorsson, M. Halldorsson, C. A. J. Hurkens, J. K. Lenstra, R. Ravi, and L. Stougie. Approximation algorithms for the test cover problem. *Mathematical Programming-B*, To appear.

[15] K. M. De Bontridder, B. J. Lageweg, J. K. Lenstra, J. B. Orlin, and L. Stougie. Branch-and-bound algorithms for the test cover problem. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, volume 2461 of *Lecture Notes in Computer Science*, pages 223–233. Springer, 2002.

[16] K. S. Booth and G. S. Lueker. Testing for the consecutive ones property, intervals graphs and graph planarity testing using PQ-tree algorithms. *J. Comput. System Sci.*, 13:335–379, 1976.

[17] A. Caprara. On the tightness of the alternating-cycle lower bound for sorting by reversals. *Journal of Combinatorial Optimization*, 3:149–182, 1999.

[18] A. Caprara. Sorting permutations by reversals and eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12:91–110, 1999.

[19] A. Caprara, M. Fischetti, and P. Toth. A heuristic method for the set covering problem. *Operations Research*, 47:730–743, 1999.

[20] A. Caprara and G. Lancia. Structural alignment of large-size proteins via lagrangian relaxation. In *Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 100–108, New York, NY, 2002. ACM Press.

[21] A. Caprara, G. Lancia, and S.K. Ng. A column-generation based branch-and-bound algorithm for sorting by reversals. In M. Farach-Colton, F.S. Roberts, M. Vingron, and M. Waterman, editors, *Mathematical Support for Molecular Biology*, volume 47 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 213–226. AMS Press, 1999.

[22] A. Caprara, G. Lancia, and S.K. Ng. Sorting permutations by reversals through branch and price. *INFORMS journal on computing*, 13(3):224–244, 2001.

[23] P. Carraresi and F. Malucelli. A reformulation scheme and new lower bounds for the QAP. In P.M. Pardalos and H. Wolkowicz, editors, *Quadratic Assignment and Related Problems*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 147–160. AMS Press, 1994.

[24] D. Casey. *The Human Genome Project Primer on Molecular Genetics.* U.S. Department of Energy, World Wide Web, `http://www.ornl.gov/hgmis/publicat/primer/intro.html`, 1992.

[25] A. Chakravarti. It's raining SNP, hallelujah? *Nature Genetics*, 19:216–217, 1998.

[26] T. Christof, M. Junger, J. Kececioglu, P. Mutzel, and G. Reinelt. A branch-and-cut approach to physical mapping with end-probes. In *Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB)*, New York, NY, 1997. ACM Press.

[27] A. Clark. Inference of haplotypes from PCR–amplified samples of diploid populations. *Molecular Biology Evolution*, 7:111–122, 1990.

[28] M. Fischetti, G. Lancia, and P. Serafini. Exact algorithms for minimum routing cost trees. *Networks*, 39(3):161–173, 2002.

[29] W. M. Fitch. An introduction to molecular biology for mathematicians and computer programmers. In M. Farach-Colton, F.S. Roberts, M. Vingron, and M. Waterman, editors, *Mathematical Support for Molecular Biology*, volume 47 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–31. AMS Press, 1999.

[30] M. Frances and A. Litman. On covering problems of codes. *Theory of Computing Systems*, 30(2):113–119, 1997.

[31] M.R. Garey and D.S. Johnson. *Computers and Intractability, a Guide to the Theory of NP-Completeness*. W.H. Freeman and Co., San Francisco, CA, 1979.

[32] A. Godzik and J. Skolnick. Flexible algorithm for direct multiple alignment of protein structures and sequences. *CABIOS*, 10(6):587–596, 1994.

[33] A. Godzik, J. Skolnick, and A. Kolinski. A topology fingerprint approach to inverse protein folding problem. *Journal of Molecular Biology*, 227:227–238, 1992.

[34] D. Goldman, S. Istrail, and C. Papadimitriou. Algorithmic aspects of protein structure similarity. In *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 512–522, New York, NY, 1999. IEEE.

[35] J. Gramm, F. Huffner, and R. Niedermeier. Closest strings, primer design, and motif search. In L. Florea, B. Walenz, and S. Hannenhalli, editors, *Posters of Annual International Conference on Computational Molecular Biology (RECOMB)*, Currents in Computational Molecular Biology, pages 74–75, 2002.

[36] M. Grötschel and W. R. Pulleyblank. Weakly bipartite graphs and the max–cut problem. *Operations Research Letters*, 1:23–27, 1981.

[37] D. Gusfield. Efficient methods for multiple sequence alignment with guaranteed error bounds. *Bulletin of Mathematical Biology*, 55:141–154, 1993.

[38] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, Cambridge, UK, 1997.

[39] D. Gusfield. A practical algorithm for optimal inference of haplotypes from diploid populations. In R. Altman, T.L. Bailey, P. Bourne, M. Gribskov, T. Lengauer, I.N. Shindyalov, L.F. Ten Eyck, and H. Weissig, editors, *Proceedings of the Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 183–189, Menlo Park, CA, 2000. AAAI Press.

[40] D. Gusfield. Inference of haplotypes from samples of diploid populations: Complexity and algorithms. *Journal of Computational Biology*, 8(3):305–324, 2001.

[41] B. V. Halldrsson, M. M. Halldrsson, and R. Ravi. On the approximability of the minimum test collection problem. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Computer Science*, pages 158–169. Springer, 2001.

[42] S. Hannenhalli and P. A. Pevzner. Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals). In *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*, pages 178–189. ACM press, 1995 (full version appeared in Journal of the ACM, 46, 1–27, 1999).

[43] M. Held and R. M. Karp. The traveling salesman problem and minimum spanning trees: Part II. *Mathematical Programming*, 1:6–25, 1971.

[44] L. Holm and C. Sander. 3-D lookup: fast protein structure searches at 90% reliability. In *Proceedings of the Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 179–187, Menlo Park, CA, 1995. AAAI Press.

[45] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001.

[46] D. S. Johnson and M. A. Trick, editors. *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.* AMS Press, 1996.

[47] W. Kabash. A solution for the best rotation to relate two sets of vectors. *Acta Cryst.*, A32:922–923, 1978.

[48] H. Kaplan, R. Shamir, and R. E. Tarjan. Faster and simpler algorithm for sorting signed permutations by reversals. *SIAM Journal on Computing*, 29(3):880–892, 1999.

[49] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. In *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 684 of *Lecture Notes in Computer Science*, pages 106–119. Springer, 1993.

[50] J. Kececioglu, H.-P. Lenhof, K. Mehlhorn, P. Mutzel, K. Reinert, and M. Vingron. A polyhedral approach to sequence alignment problems. *Discrete Applied Mathematics*, 104:143–186, 2000.

[51] J. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, 13:180–210, 1995.

[52] G. Lancia, V. Bafna, S. Istrail, R. Lippert, and R. Schwartz. SNPs problems, complexity and algorithms. In *Proceedings of the Annual European Symposium on Algorithms (ESA)*, volume 2161 of *Lecture Notes in Computer Science*, pages 182–193. Springer, 2001.

[53] G. Lancia, R. Carr, B. Walenz, and S. Istrail. 101 optimal PDB structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem. In *Proceedings of the Annual International Conference on Computational Biology (RECOMB)*, pages 193–202, New York, NY, 2001. ACM Press.

[54] J. Lanctot, M. Li, B. Ma, S. Wang, and L. Zhang. Distinguishing string selection problems. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 633–642, 1999.

[55] E. Lee, T. Fox, and I. Crocker. Optimization of radiosurgery treatment planning via mixed integer programming. *Medical Physics*, 27(5):995–1004, 2000.

[56] E. Lee, T. Fox, and I. Crocker. Integer programming applied to intensity-modulated radiation treatment planning optimization. *Annals of Operations Research, Optimization in Medicine*, 119(1-4):165–181, 2003.

[57] E. Lee, R. Gallagher, and M. Zaider. Planning implants of radionuclides for the treatment of prostate cancer: an application of mixed integer programming. *Optima (Mathematical Programming Society Newsletter)*, 61:1–10, 1999.

[58] E. Lee and M Zaider. Mixed integer programming approaches to treatment planning for brachytherapy – application to permanent prostate implants. *Annals of Operations Research, Optimization in Medicine*, 119(1-4):147–163, 2003.

[59] C. Lemmen and T. Lengauer. Computational methods for the structural alignment of molecules. *Journal of Computer–Aided Molecular Design*, 14:215–232, 2000.

[60] H.-P. Lenhof, K. Reinert, and M. Vingron. A polyhedral approach to RNA sequence structure alignment. *Journal of Computational Biology*, 5(3):517–530, 1998.

[61] M. Li, B. Ma, and L. Wang. On the closest string and substring problems. *Journal of the ACM*, 49(2):157–171, 2002.

[62] S. Lin and B. Kernigan. An efficient heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2), 1973.

[63] R. Lippert, R. Schwartz, G. Lancia, and S. Istrail. Algorithmic strategies for the SNPs haplotype assembly problem. *Briefings in Bioinformatics*, 3(1):23–31, 2002.

[64] B. Ma. A polynomial time approximation scheme for the closest substring problem. In *Proceedings of the Annual Symposium on Combinatorial Pattern Matching (CPM)*, volume 1848 of *Lecture Notes in Computer Science*, pages 99–107. Springer, 2000.

[65] J. Moult, T. Hubbard, S. Bryant, K. Fidelis, J. Pedersen, and Predictors. Critical assessment of methods of proteins structure prediction (CASP): Round II. *Proteins*, Suppl.1:dedicated issue, 1997.

[66] J. Moult, T. Hubbard, K. Fidelis, and J. Pedersen. Critical assessment of methods of proteins structure prediction (CASP): Round III. *Proteins*, Suppl.3:2–6, 1999.

[67] E. Myers. Whole-genome DNA sequencing. *IEEE Computational Engineering and Science*, 3(1):33–43, 1999.

[68] E. Myers and J. Weber. Is whole genome sequencing feasible? In S. Suhai, editor, *Computational Methods in Genome Research*, pages 73–89, New York, 1997. Plenum Press.

[69] G. L. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization.* John Wiley and Sons, 1988.

[70] P. Nicodeme and J. M. Steyaert. Selecting optimal oligonucleotide primers for multiplex PCR. In *Proceedings of the Annual International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 210–213, Menlo Park, CA, 1997. AAAI Press.

[71] W. R. Pearson, G. Robins, D. E. Wrege, and T. Zhang. On the primer selection problem in polymerase chain reaction experiments. *Discrete Applied Mathematics*, 71:231–246, 1996.

[72] P.A. Pevzner. *Computational Molecular Biology.* MIT Press, Cambridge, MA, 2000.

[73] P. Raghavan. A probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences*, 37:130–143, 1988.

[74] P. Raghavan and C. D. Thompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, 1987.

[75] K. Reinert, H.-P. Lenhof, P. Mutzel, K. Mehlhorn, and J. Kececioglu. A branch-and-cut algorithm for multiple sequence alignment. In *Proceedings of the Annual International Conference on Computational Molecular Biology (RECOMB)*, pages 241–249, New York, NY, 1997. ACM Press.

[76] D. Sankoff, R. Cedergren, and Y. Abel. Genomic divergence through gene rearrangement. In *Molecular Evolution: Computer Analysis of Protein and Nucleic Acid Sequences*, pages 428–438. Academic Press, 1990.

[77] J. S. Sim and K. Park. The consensus string problem for a metric is NP-complete. In *Proceedings of the Annual Australasian Workshop on Combinatorial Algorithms (AWOCA)*, pages 107–113, 1999.

[78] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[79] G. Stoesser, W. Baker, A. van den Broek, , E. Camon, M. Garcia-Pastor, C. Kanz, T. Kulikova, R. Leinonen, Q. Lin, V. Lombard, R. Lopez, N. Redaschi, P. Stoehr, M. Tuli, K. Tzouvara, and R. Vaughan. The EMBL nucleotide sequence database. *Nucleic Acids Research*, 30(1):21–26, 2002.

[80] J.C. Venter *et al.* The sequence of the human genome. *Science*, 291:1304–1351, 2001.

[81] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1:337–348, 1994.

[82] M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences, and Genomes.* Chapman Hall, 1995.

[83] J. D. Watson, M. Gilman, J. Witkowski, and M. Zoller. *Recombinant DNA*. Scientific American Books. W.H. Freeman and Co, 1992.

[84] J. Weber and E. Myers. Human whole genome shotgun sequencing. *Genome Research*, 7:401–409, 1997.

[85] B.Y. Wu, G. Lancia, V. Bafna, K.M. Chao, R. Ravi, and C.Y. Tang. A polynomial–time approximation scheme for minimum routing cost spanning trees. *SIAM Journal on Computing*, 29(3):761–778, 1999.

[86] J. Xu, M. Li, D. Kim, and Y. Xu. RAPTOR: Optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 1(1):95–117, 2003.