Algorithmic Techniques in Computational Genomics

by

Laxmi Parida

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Computer Science
New York University
September 1998

Approved: _____
Bud Mishra

Dedicated to those who are reading this thesis.

Acknowledgements

I am grateful to my advisor, Bud Mishra, who treated me more as an equal, and less as a struggling student. I am grateful to David Schwartz at the Department of Chemistry, NYU, for introducing me to Optical Mapping. My sincere thanks to Richard Cole, Davi Geiger, Rohit Parikh and Alan Siegel for their continuing support and interest in my work and career.

My sincere thanks to Aris Floratos, Muthu Muthukrishnan and Isidore Rigoutsos for interesting collaborative efforts and Chandrasekar for his careful reading of various versions of the thesis.

I owe it to the many people around me for making my graduate years a pleasant learning experience. Thanks in particular to Saugata Basu, Aris Floratos and Ian Jermyn for their infinite patience in being a kind audience to my ramblings and providing useful insights. Thanks to Juan Carlos Porras and Archisman Rudra for the long hours spent in the racquet courts that helped me retain my sanity and provided data for the *apeseque* index.

Alpana, allow me to thank you for teaching me that the impossible is sometimes possible.

I owe, much more than I can possibly express, to Tuhina, Ma and Bapa for their patience, encouragement and support. My sincere gratitude to Tuhina for her incredible understanding for so young a mind!

Contents

List of Figures

List of Appendices

Chapter 1

Introduction

This thesis explores the application of algorithmic techniques in understanding and solving computational problems arising in Genomics (called *Computational Genomics*). In the first part of the thesis we focus on the problem of reconstructing physical maps from data, related to "reading" the genome of an organism, and in the second part we focus on problems related to "interpreting" (in a very limited sense) the genome.

The first part depends on the underlying DNA technology used in a laboratory: we study problems arising from one such DNA technology called Optical Mapping (see Chapter 3 for a brief overview). At the time of writing this thesis, one of the historic goals in biomedical research, that of sequencing the three billion bases of the human genome, is yet to be achieved. It is only a matter of time (three to ten years) before this goal is reached. However, the task of sequencing genomes of a host of other microbial organisms and other living forms will be of continuing interest to scientists. At this point, it is unclear as to which particular DNA technology is here to stay. Keeping this volatile nature of the subject in mind, in Chapter 2 we give a brief overview of some basic concepts in molecular biology. In the next chapter we describe the Physical Map problem, and in Chapter 4, we present a uniform framework for the computational problems. We describe two combinatorial models of the problem termed Exclusive Binary Flip Cut (EBFC) and Weighted Consistency Graph (WCG) problems (Chapter 5). We show that both the problems are

MAX SNP hard and give bounds on the approximation factors achievable. We give polynomial time 0.878-approximation algorithm for the EBFC problem and 0.817-approximation algorithm for the WCG problem (Chapter 6). We also give a low polynomial time practical algorithm that works well on simulated and real data (Chapter 7). *Naksha* is an implementation of this algorithm and a demonstration is available at `http://www.cs.nyu.edu/parida/naksha.html`. We also have similar results on complexity for generalizations of the problem which model various other sources of errors (Chapter 8). We have generalized our complexity and algorithmic results to the case where there is more than one population in the data (which we call the K-populations problem).

In the second part of the thesis, we focus on "interpreting" the genome. We consider the problem of discovering patterns (or motifs) in strings on a finite alphabet: we show that by appropriately defining irredundant motifs, the number of irredundant motifs is only quadratic in the input size (Chapter 9). We use these irredundant motifs in designing algorithms to align multiple genome or protein sequences (Chapter 10). Alignment of sequences aids in comparing similarities, in structure and function of the proteins.

Chapter 2

Molecular Biology - a whirlwind tour[1]

The term *molecular biology* was coined in 1939 by Warren Weaver, in a report to the president at a time when X-ray crystallography was being fervently pursued to unravel the structure of proteins. By now, about six decades later, it is a well established field with open problems challenging natural scientists and mathematicians alike.

In this chapter we briefly describe the basics to understand the source of the various computational problems, and familiarize ourselves with the vocabulary of biologists and biochemists as much as we possibly can. This chapter is intended to give only a nodding acquaintance with the terms and concepts in molecular biology and the reader is referred to the citations mentioned for further details.

Roadmap. We begin by describing the classification of all living organisms based on the structure of the building unit of an organism – the cell, in Section 2.1. The focus of our study is the DNA molecule(s) that resides in every cell: we discuss its structure in Section 2.2. The DNA molecule is responsible for each protein found in a living organism which is vital to the existence of the organism: we describe the

[1] Portions of this chapter also appear in the survey paper entitled "Computational Molecular Biology: Problems and Tools", in the Journal of the Indian Institute of Science [42].

relationship between DNA and proteins in Section 2.3. We conclude the chapter by describing the various prevalent molecular genetic techniques in Section 2.4.

2.1 Living Organisms

We start at the very beginning, a very good place to start: the classification of living organisms.

Virus: A sub-microscopic organism that is incapable of reproduction outside a host cell. It consists of a genome, DNA or RNA, and a protein body. Thus a virus has no bio-synthetic activities: it resides in a host bacteria and multiplies using the bacteria's mechanisms.

Prokaryotes: Unicellular; the only organelles in the cell are *ribosomes* and a genome consisting of a single closed loop of DNA[2]. Note the absence of a nucleus in the prokaryotes. Prokaryotes show almost no genetic diversity and reproduce asexually via cell division[3].

Example: bacteria.

Eukaryotes: Mostly multi-cellular; have various specialized organelles as shown in Figure 2.1. Sexual reproduction, a mechanism for increasing the genetic diversity is common.

Example: all higher order organisms like plants, mice, humans etc.

2.2 Structure of DNA/RNA

Our focus is on the chromosomes, found in the nucleus, which contain the blueprint for the entire organism. In the following sections we study its structure and the mechanism by which the blueprint is interpreted.

The "genetic material" in organisms are genes, which are composed of deoxyribonucleic acid, DNA. DNA is a very large molecule, and is made of small molecules called *neulceotides*. It consists of two complementary chains twisted about each

[2]Further, prokaryotes have no *introns* in their DNA sequence.

[3]*omnis cellula e cellula*: cells divide to multiply!

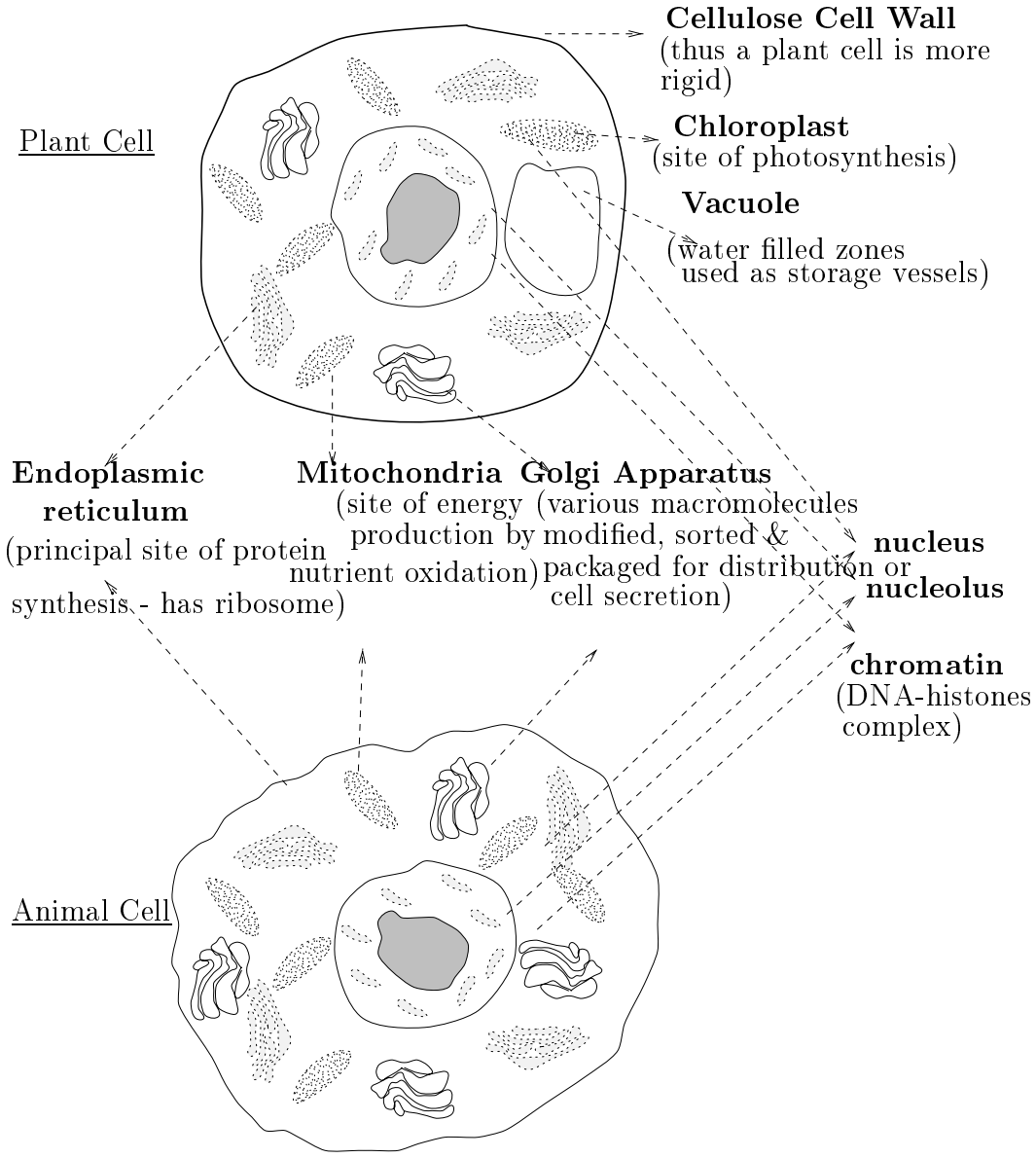


Figure 2.1: Membrane-bound internal structures, called *organelles*, and their functions in eukaryotic cells.

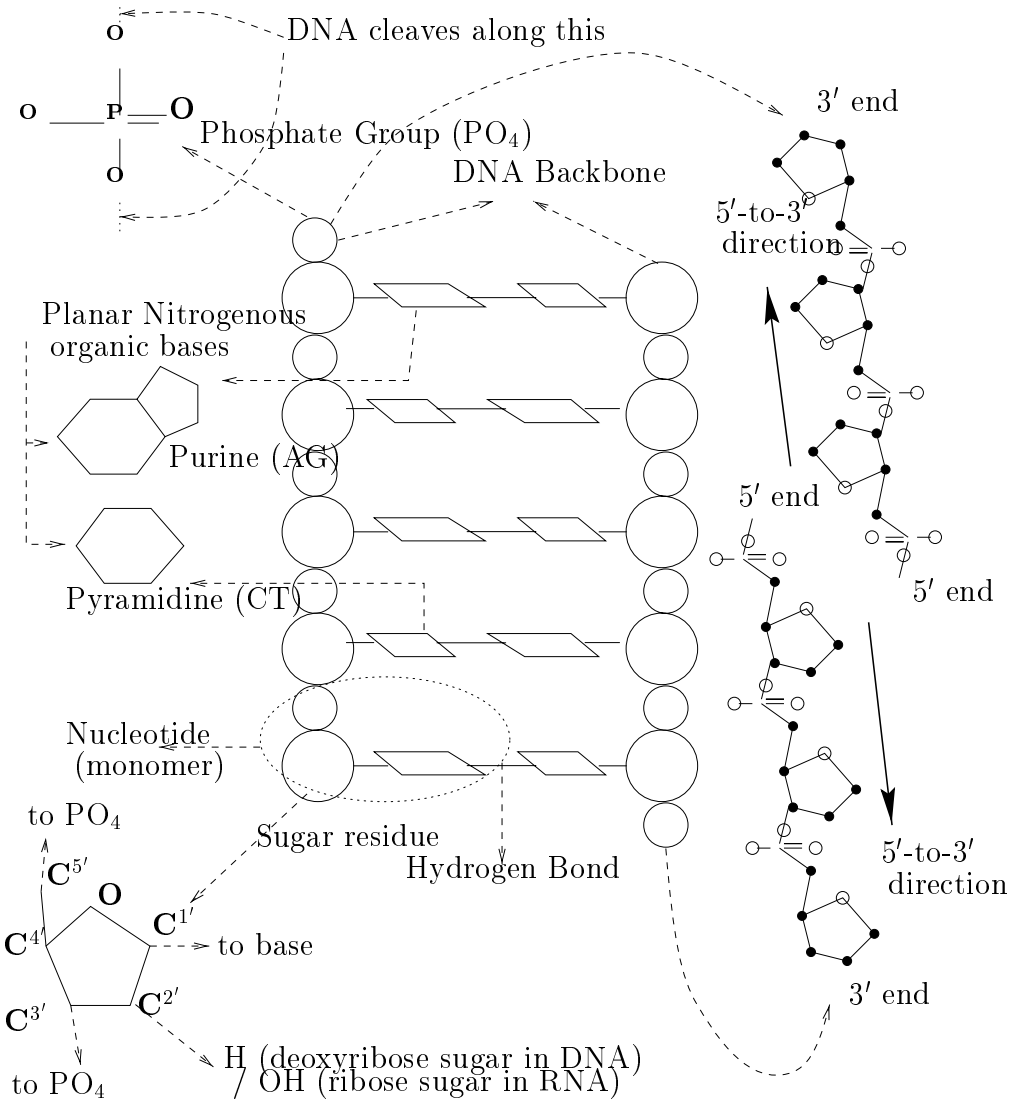


Figure 2.2: Structure of DNA. The plane of the planar bases (A,G,C,T) is perpendicular to the helix axis, shown as a string of balls in the picture. Note the opposite directions of the two backbones. The backbone is made of the phosphate group $(PO_4)^{-3}$ and the sugar, $C_5O_4H_{10}$. The base, the phosphate and the sugar form the unit of nucleotide: this is also the unit that is added during the synthesis of DNA.

other in the form of a double helix. Each chain is composed of four nucleotides that contain a deoxyribose residue, a phosphate, and a pyramidine or a purine base. The pyramidine bases are thymine (T) and cytosine (C); the purine bases are adenine (A) and guanine (G). The "sides" of the double helix consist of deoxyribose residues linked by phosphates. The "rungs" are made of an irregular order of pyramidine and purine bases. The two strands are joined together by hydrogen bonds existing between the pyramidine and the purine bases: Adenine is always paired with thymine (AT) and guanine is always paired with cytosine (GC). See Figure 2.2. These are also called the base pairs.

RNA is very similar to DNA with the following differences:

1. It is single stranded, i.e., only one backbone, with the bases, is present;
2. OH is attached to $C^{2'}$ of the sugar residue, instead of H, in the backbone, as shown in Figure 2.2, and,
3. Uracil replaces the pyramidine base Thyamine.

Orientation of DNA strands: Note that the structure of the sugar is asymmetric, i.e., its bottom and top ends (where it is attached to the neighboring phosphates) are not identical. These are designated $5'$ and $3'$ to distinguish the two ends. Also the backbone pair is oppositely directed. Thus, the ends of the DNA strands are designated $5'$ and $3'$.

Size of DNA: How large is the macromolecule? Let us look at its size in terms of the base pairs. The human genome contains 23 pairs of chromosomes which consists of approximately 3.6×10^9 base pairs. In contrast, the chromosome of *Escherichia Coli* contains only 4×10^6 DNA base pairs. Also, mitochondrial DNA (mt-DNA)[4] is about 16×10^3 bases in length, and is circular rather than linear.

DNA can be classified in at least two ways:
By structure.

[4] As the name suggests this is non-genomic DNA found in the mitochondria. It codes some 13 proteins in humans and mutation in the mt-DNA is responsible for diseases like *Leber's optic atrophy* and is transmitted solely by mothers – an example of nonmendelian inheritance.

1. Repetitive DNA (SINES & LINES) : The major human SINE (short interspersed repeated sequences) is the *Alu* DNA sequence family which is repeated between $300,000$ and $900,000$ times in the human genome. The function of *Alu* is unknown and the reason for their very high frequency in the human genome remains a mystery. LINES (long interspersed repeated sequences) has a consensus sequence of 6400 base pairs. This is repeated between 4000 and $100,000$ times. As with the *Alu* sequence the function of these sequences are unknown.

2. Unique sequence DNA: This contain sequences that code for mRNA (messenger RNA). In general, genes are comprised of unique sequence DNA that encodes information for RNA and protein synthesis. Mitochondrial DNA consists mostly of unique sequence DNA.

By function.

1. Exons: These are the functional portions of the gene sequences that code for proteins. Roughly speaking, these correspond to the *genes*. A gene is a hereditary unit that is responsible for a particular characteristic in an organism.

2. Introns: These are the noncoding DNA sequences of unknown function that interrupt most mammalian genes.

2.2.1 Chromosome Structure

The genome of an organism consists of smaller units, *chromosomes*: corn has 20, certain fruitflies have 8 chromosomes, rhinoceroses 84, humans and bats have 46.

Each chromosome contains a single molecule of DNA organized into several orders of packaging to construct a metaphase chromosome: the length of this is about 0.0001 times the length of its DNA. DNA, along with the binding proteins, is called *chromatin*. *Histones* are the structural proteins of the chromatin and are the most abundant proteins in the nucleus. Figure 2.3 shows some interesting details of the packaging of DNA.

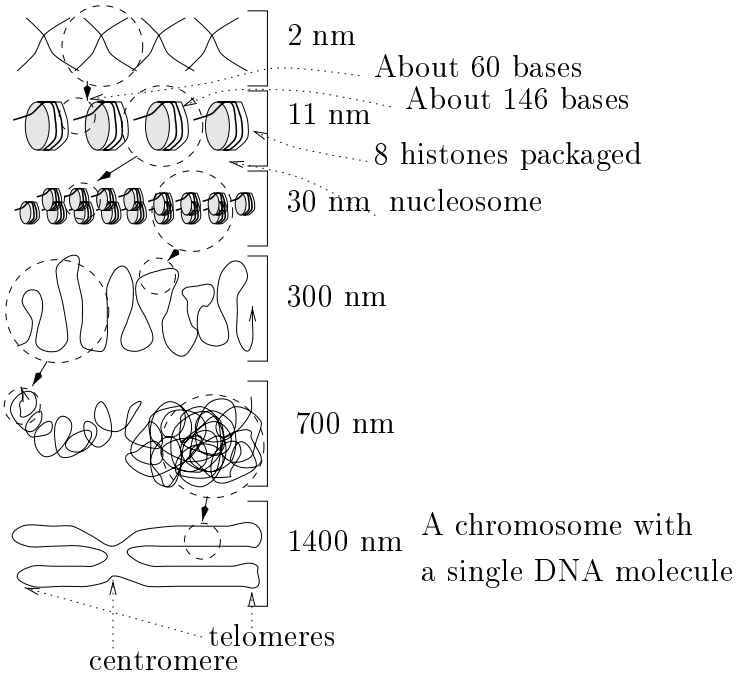


Figure 2.3: Packaging of DNA: the figure gives a sense of the scale we are dealing with.

Euchromatin forms the main body of the chromosome and has relatively high density of coding regions or genes. The *chromosome bands* define alternating partitions of euchromatin with differing properties.

R bands: These stain light with a procedure called the *G banding procedure*. They have a relatively high content of guanine and cytosine, have the majority of SINES (see Section 2.2), and have the highest gene density.

G bands: These stain dark with the *G banding procedure*. They have a higher content of adenine and thymine, have the majority of LINES (see Section 2.2), and have relatively fewer genes.

Heterochromatin is chromatin that is either devoid of genes or has inactive genes.

Each chromosome consists of two parallel strands, the *sister chromatids*, which are held together by a *centromere*. The centromere consists of specific DNA sequences that bind proteins. *Telomeres* are DNA sequences found at the ends of the chromosomes, which are required to maintain chromosome stability. Chromosomes without telomeres that tend to recombime with other chromatin segments are generally subject to breakage, fusion, and eventual loss. The terminal segments of all chromosomes have a similar sequence (TTAGGG), which is present in several thousand copies. Telomere sequences facilitate DNA replication at the end of chromosomes.

2.3 Protein Synthesis (DNA→RNA→Protein)

This section describes how the "blueprint" is put into effect. Every protein, found in the living body, is synthesized by "executing the program encoded in the DNA".

The protein synthesis occurs in the following steps:

1. Transcription: A DNA segment, a *gene*, serves as the template for the synthesis of a single stranded RNA, *messenger* RNA (mRNA). Note that the base Uracil (U) replaces the base T.

   This is similar to DNA replication (See Section 2.4.3.) and requires the essential ingredients: catalytic agent, the RNA polymerase; the Master template,

the DNA segment and the building blocks, the NTPs (ribonucleoside triphosphates). Note the absence of the primer.

(a) The RNA polymerase attaches itself to the *promoter* segment of the double stranded DNA.

(b) The DNA segment denatures.

(c) The RNA polymerase facilitates the *hydrogen bonding* of exposed bases with the complementary NTPs. The RNA polymerase further catalyzes the *covalent bonding* between the bases (see DNA replication for more details). Thus the mRNA grows in the $5'$-to-$3'$ direction.

(d) The DNA segment renatures.

2. Splicing: In eukaryotes, both the exons and the introns are transcribed. The resulting primary transcript is spliced; that is, each intron is removed and the exons are linked together.

   This step is absent in prokaryotes. The mRNA leaves the nucleus via the pores and enters the cytoplasm for the next step.

3. Translation: The mRNA serves as a template for stringing together the amino acids in the protein. The succession of *codons* (triplets of adjacent ribonucleotides) determines the amino acid that composes the protein.

   Again, this process is similar to DNA replication and requires the essential ingredients: the ribosome[5] functions as the catalytic agent; mRNA as the master template. The building blocks are the amino acid monomers, but the process of assembly requires a transfer RNA (tRNA).

   (a) Getting the building blocks ready: A tRNA is a tiny clover-leaf-shaped molecule that has at one end a triplet of ribonucleotides, an *anticodon*, that binds with a complementary codon on the mRNA, and, an attachment site for a single amino acid at the other end. A catalyst, aminoacyl

[5] A very large molecule composed of ribosomal RNA and at least fifty different proteins.

synthetase, converts the tRNA to an aminoacyl-tRNA by attaching the appropriate amino acid to the other end.

(b) The ribosome travels along the mRNA in the 5′-to-3′ direction synthesizing a polymer of amino acids, a protein.

i. An aminoacyl-tRNA attaches itself to the START codon of the mRNA.

ii. An appropriate aminoacyl-tRNA attaches to the next codon and the amino acid at its end forms a *peptide bond* with the previous amino acid. The tRNA of the previous one is released. Thus a chain of amino acids is formed with the last monomer still attached to the tRNA.

iii. The process continues until a STOP codon is reached.

The ribosome detaches itself and the protein is released into the cytoplasm.

2.4 Molecular Genetic Techniques

This section is intended to familiarize the reader with ways of manipulating DNA. However in the rest of the thesis (especially Part I) we deal with yet another DNA technology called Optical Mapping.

We look at the prevalent techniques for manipulating and analyzing DNA. They can be categorized as:

1. DNA Fragmentation,
2. Fractionating DNA fragments,
3. DNA Amplification,
4. DNA Sequencing, and
5. Hybridization.

2.4.1 DNA Fragmentation (Molecular Scissors)

Since a single molecule of DNA has about 130 million base pairs, it is important to "chop" the molecule into manageable pieces.

DNA molecules are fragile, and mechanical aspects of sample preparation, such as stirring and pipetting, break some of the covalent bonds of the backbones. But the disadvantage is that it is not repeatable, that is, is not expected to break at the same sites. *Restriction Enzymes*[6] are biochemicals capable of cutting the double-stranded DNA, by breaking two -O-P-O- bridges on each backbone of the DNA pair, at specific sequences called *restriction sites*.

Note that restriction enzyme is a naturally occurring protein in a bacteria that defends the bacteria from invading viruses by cutting up the DNA of the latter. How does the bacterium's own DNA escape the assault? The bacterium produces another enzyme that methylates the restriction sites of its own DNA – this prevents the cleaving action of the restriction enzyme.

Restriction Sites: Let us look at an example to see the effect of the cleaving. The restriction enzyme *Eco*RI recognizes and binds to the palindromic sequence

$$5'\text{-GAATTC-}3'$$

$$3'\text{-CTTAAG-}5'$$

If allowed to interact for a sufficiently long time, it cuts the DNA as shown below (| denotes a cut):

$$5'\text{-G|AATT C-}3'$$

$$3'\text{-C TTAA|G-}5'$$

The staggered cuts produce fragments with very "sticky" single stranded ends. These can combine with other matching strands. Some restriction enzymes might cut straight without producing "sticky ends". For example, *Hae*III,

$$5'\text{-GG|CC-}3'$$

$$3'\text{-CC|GG-}5'$$

[6] Arber, Smith, and Nathan received the Nobel prize for their discovery of restriction enzymes in 1978.

What is the average length of a fragment cut by a restriction enzyme? This is easy to compute: let it be an "n-base cutter", then the pattern occurs on an average every 4^n base pairs. This is the best estimate we have in the absence of any more information: reality might be quite different.

2.4.2 Fractionating DNA fragments

The DNA could be fragmented by any method including the one above and sorted as follows.

By Length – Gel Electrophoresis: This is a process whereby the fragments are separated according to their size or electrical charge, on a slab of gelatinous material, under the influence of an electric field.

The phosphate groups in the DNA are negatively charged; hence under the influence of an electric field, the fragments migrate towards the anode. The rate at which it migrates is approximately inversely proportional to the logarithm of its length.

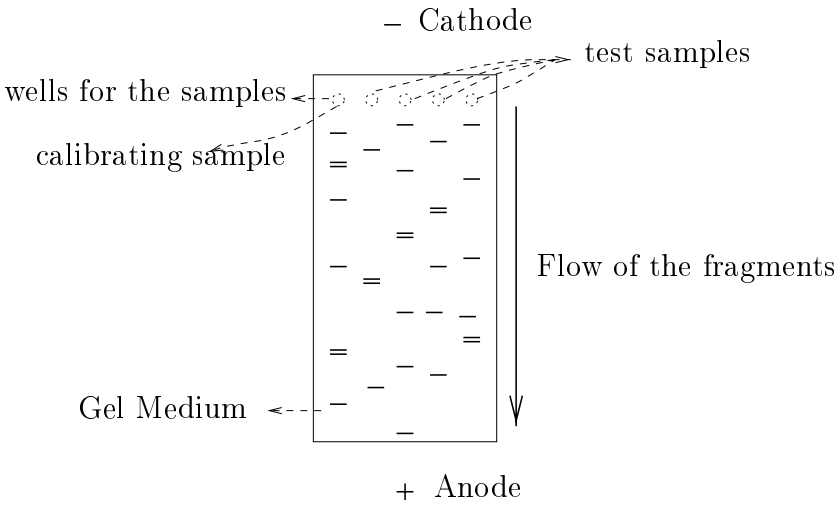


Figure 2.4: Gel Electrophoresis: The fragments are separated by lengths. The left-most known sample calibrates the tracks (vertically) - thus the lengths of the remaining samples can be read off using the first reference.

On a slab of gel, wells are made at the top (see Figure 2.4). The leftmost well contains the calibrating fragments, that is, a sample whose lengths are known. The relative positions of the rest of the columns of fragments with respect to this calibrator give an estimate of the lengths.

Large fragments, over about $50,000$ base pairs, do not move well under the influence of steady electric field: *pulsed-field* gel electrophoresis employs a field that is temporarily constant in both direction and magnitude. This solves the problem of large fragments.

By Structure – Renaturing: A double strand of DNA *denatures* at around 100°C. When the temperature is lowered, the strands randomly *renature*. Rapid renaturing implies high repetitive sequences and slow renaturing indicates unique sequence DNA. This technique is used to separate sequences by the repetitive pattern.

2.4.3 DNA Amplification (Molecular Copier)

Most techniques used in the analysis of DNA rely on the availability of many copies of the segment. We first discuss the DNA replication process appearing in nature (during cell division – mitosis and meiosis), and then discuss the molecular genetic technique of making copies.

Cell Division (Mitosis & Meiosis). There are two kinds of cells: *somatic* and *germ*, (also called *gamete*) cells. For lack of a better description, somatic cells are regular cells and germ cells are the reproductive cells. Both the cells multiply by division, in a process called *mitosis*. Germ cells also have a special cell division called *meiosis*. We shall not get into the details of each of this but give a general overview of the processes.

Chromosomes eluded mankind until early this century, even after the advent of powerful microscopes. It was seen, early this century, that during the cell division process, mitosis, certain structures became visible when appropriately dyed – hence the term *chromosomes*. These condense during mitosis, becoming visible under a microscope.

The meiosis process is more interesting than the mitosis, in the sense that

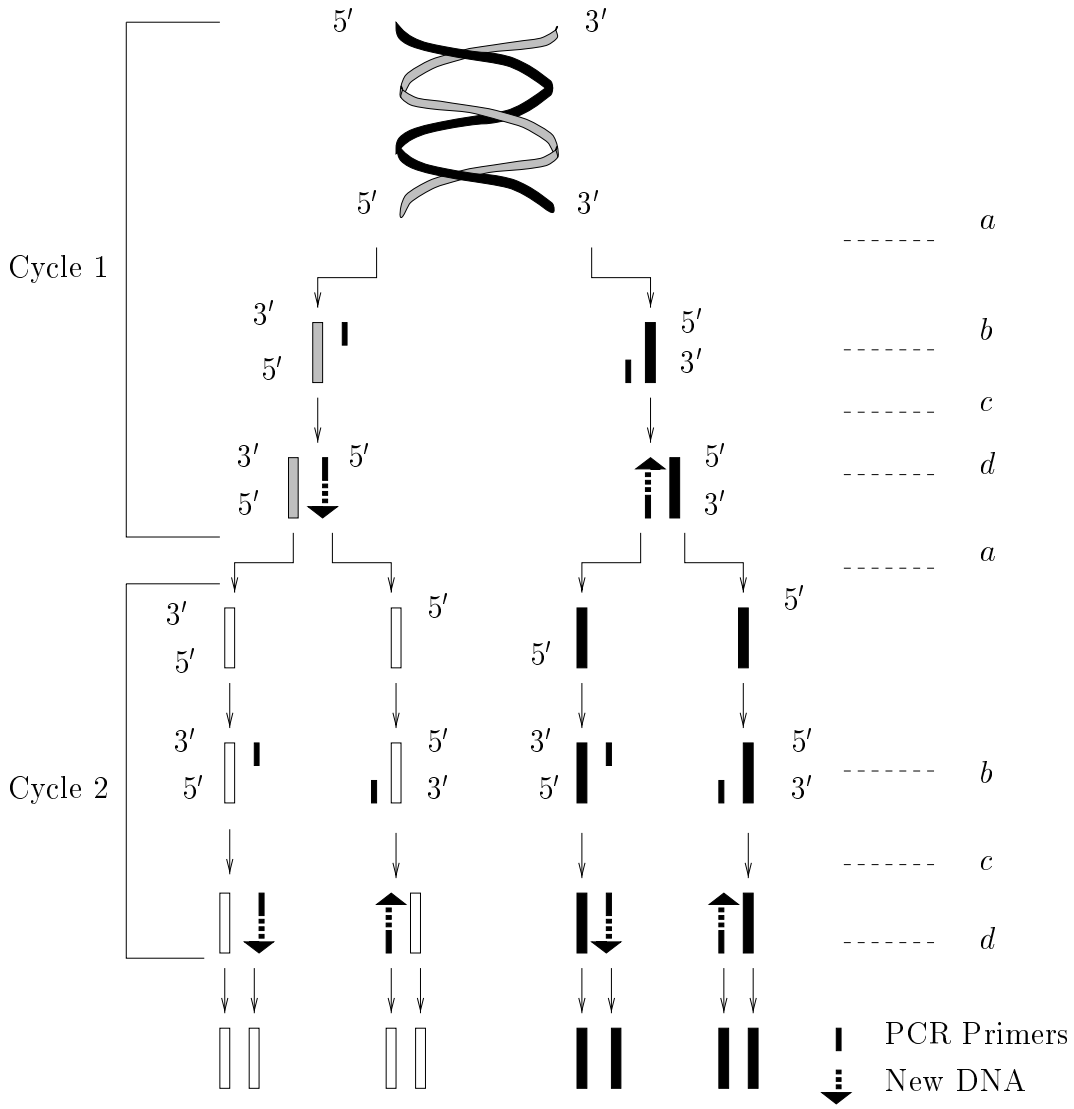


Figure 2.5: PCR is based on the amplification of a DNA fragment flanked by two primers that are complementary to opposite strands of the sequence being investigated. In each cycle, (a) heat denaturation separates the strands. (b) Primers are added in excess and hybridized to complementary fragments. (c) dNTPs and polymerase are added while the temperature increases. (d) The primer is extended in the $3'$ direction as new DNA is extended in the $5'$ direction.

mitosis is an *exact copying* mechanism so far as the chromosomes are concerned, whereas meiosis produces some variations, called *cross-over*, leading to genetic variations. In this, the pair of homologous chromosomes in *diploids* exchange material from corresponding regions. The cross-over information can be used to form a *genetic map* based on traits. Many of these traits are linked, in the sense that these traits are passed on together, as a single package, to the offsprings: for example, color of eyes, size of wings and color of the body in *Drosophila*. But occasionally, the traits switch groups and this is attributed to cross-over. The more often a linked pair get separated, the further apart they are on the chromosome. Thus a map can be formed for each chromosome, listing the traits (or the genes corresponding to the traits) in linear order with rough distances between them. The unit of this distance was named *morgan* [7], by the biologist J. B. S. Haldane.

DNA Replication. Let us look at the chief actors and the roles they play in the replication process:

Primer: This is the initiator of the new strand. The usual primer is a very short strand of RNA with four to twelve nucleotides.

Catalytic Agents(DNA polymerase): An enzyme that catalyzes the polymer formation process.

A Master template: The parental DNA strand.

Building Blocks(dNTPs/deoxyribonucleoside triphosphates): As expected, they are of four kinds : dATP, dCTP, dTTP and dGTP corresponding to the four bases.

Starting from a single double-stranded parental DNA molecule, the replication process gives two identical double-stranded daughter molecules. Both the daughter molecules are such that one of the strands is that of the parent and the other is the new synthesized strand.

The replication of the entire strand of DNA occurs in parallel in short strands all over the molecule and merges finally in a rather complex way. Let us look at the steps involved in the replication at a single site, flanked by two *origins of*

[7] Thomas Hunt Morgan and colleagues, early this century, gave a physical basis to the then forgotten Mendelian theory by demonstrating a structural relationship between genes and chromosomes.

replication.

1. The parent strand uncoils or *denatures.*
2. The two uncoiled strands replicate.
   (a) A base in the parental strand attaches to the dNTP, by *hydrogen bonds*, containing the complementary base. Thus the dNTP is fixed in position.
   (b) The DNA polymerase catalyzes the creation of an -O-P-O- bridge between the bases, thus forming *covalently bonded* bases.

   Note that the chain grows only in one particular direction, the $5'$-to-$3'$ direction. As a result one strand duplicates continuously but the other (which must proceed in the opposite $3'$-to-$5'$ direction) does so in short strands called *Okazaki fragments.*
3. The two new pairs of strands recoil or *renature* or *anneal.*

How erroneous is the process? The chances are one in a billion that a base in the synthesized daughter DNA would be incorrect!

Controlled DNA Amplification. Two molecular genetic techniques for making copies are:

1. Molecular Cloning: In this method some living cells are used to replicate the DNA sequences. The necessary ingredients are:
   (a) Insert: The DNA segment that is to be amplified.
   (b) Host Organism: This is the host cell, usually a bacterium, whose replication mechanism is being exploited.
   (c) Vector: This is a DNA segment, with which the *insert* is combined. This is usually a *plasmid*, a nongenomic DNA in the host organism. Some common examples of host organism, vector pairs are shown below along with an approximate size of the vectors (in kilo base pairs) that it can carry stably.

	Host	Vector	Sizes (in kbp)
1	*Escherichia Coli* (found in a vertebrate's intestine)	(a) λ phage genome	40
		(b) plasmid (natural)	4
		(c) cosmid (synthetic)	40
		(d) bacterial artificial chromosome (BAC)	150
2	*Saccharomyces cerevisiae* (baker's yeast)	yeast artificial chromosome (YAC) (synthetic)	1000

The following steps are involved:

(a) Preparing the recombinant DNA: This is done *in vitro*, that is, outside a living cell. The *insert* is combined with the *vector*, say the plasmid. The plasmid is circular, hence it is linearized by digesting with an appropriate restriction enzyme. The DNA strand is digested with the same restriction enzyme, so that the "sticky" ends ligase in the presence of the enzyme, *DNA ligase*.

The rest of the steps are carried out *in vivo*, that is, inside the living cell.

(b) Host Cell Transformation: The *host cell* is exposed to the ligation mixture so that the recombinant DNA may enter the cell. This process is not fully understood, although it can be fairly well controlled.

(c) Cell Multiplication: The solution with the transformed host cells is moved to culture dishes and allowed to multiply in a solid growth medium.

(d) Colony Selection: A colony of cells is produced in the dishes. Note that there is a possibility that the recombinant DNA failed to transform a host cell in step (b). At this step, the different colonies are checked for the presence of the recombinant DNA by various methods (say checking for the expression of a characteristic of the recombinant DNA).

2. **Polymerase Chain Reaction (PCR)**[8]: This is an *in vitro* process which is remarkably simple to understand. The requirement is to amplify a segment of the paired DNA. Recall the essential ingredients for DNA replication:primer, catalysts, template and the dNTPs. But here we wish to replicate only a certain segment; hence two *primers*, which form the complementary ends of the segment are used. Further, we replicate many times, hence repeated denaturing and renaturing is carried out by changing the temperature. The steps are shown in Figure 2.5. Every cycle doubles the number of existing segments, thus the number of cloned segments increases geometrically.

2.4.4 DNA Sequencing

We discuss two sequencing methods, the Maxam-Gilbert Method and, the Sangar Method[9] which have the following important characteristics:

1. They work on short segments of about 500 to 2000 base pairs.
2. The final step involves reading off the sequence from a radiogram of a gel electrophoresis process. Hence it can be done mechanically and thus automatic sequencing machines exist.

For the details of the methods the reader is directed to the references in [12]. We briefly sketch the underlying principle: both operate on the ability to identify the base at one end of the segment, with the other end being fixed or labeled. Since the gel electrophoresis fractionates by length, the longest length gives the rightmost base, the shortest gives the leftmost and so on. Thus the rows in the Gel Electrophoresis correspond to the different lengths and the four columns correspond to the four bases A, C, G and T. The natural question is whether Gel Electrophoresis can actually resolve segments differing in length by a single base: the answer is yes.

[8] Mullis and Smith received the Nobel prize in 1993 for this technique, which has become a household term after the O. J. Simpson trial.

[9] Sangar and Gilbert received the Noble prize, in 1980, for their work on sequencing techniques.

- In the Maxam-Gilbert Method, a clever scheme of cleaving at base A or C or G or T is employed. Thus one has four test tubes each with segments cleaved at one of the bases.
- In the Sangar Method, a complementary segment is allowed to grow and stop selectively at the four bases. Thus this newly synthesized strand terminates at the four different bases in a controlled manner in different test tubes.

The samples from the separate test tubes, in both methods, are used to produce the four different lanes in the gel electrophoretic method.

2.4.5 Hybridization

This refers to the hydrogen bonding that occurs between any two single-stranded nucleic-acid fragments that are complementary along some portion of their lengths.

If a short fragment of known sequence is labeled with a fluorescent molecule and allowed to interact with denatured chromosomes, the presence or absence of the complementary segment in the chromosome can be ascertained. In particular, in-situ Hybridization (ISH), can be used to locate a sequence in a chromosome, or the position within a single chromosome. Southern Hybridization is used for identifying among a sample of many different DNA fragments, the fragment(s), identified by length, containing the particular sequence.

In the following chapter we discuss the Human Genome Project and the related techniques and problems.

Part I

Physical Map Reconstruction

Chapter 3

The Physical Map Problem

3.1 The Human Genome Project

Genomics encompasses the study of the genome, the totality of a cell's genetic information, and related issues. The purpose of the Human Genome Project is to understand the DNA sequences that determine an organism's *phenotypic* (physical or expressed) characteristics.

Classical Genetics refers to those aspects of genetics that can be studied by observing traits or phenotypes. *Molecular Genetics* is studied with reference to the molecular details of genes.

The Human Genome Project [12] is the first internationally coordinated effort to read the entire genetic DNA text of three billion base pairs that constitute the human genome. This is a fifteen year project started in 1990, and coordinated by the U.S. Department of Energy and the National Institutes of Health. Its main objectives are:

1. Identify the 100,000 genes in the human DNA.

2. Determine the sequences of the DNA, store this information in databases and develop tools for data analysis.

3. Although not the primary goal, study the genetic makeup of several non-human organisms such as *Escherichia coli*, the fruit fly and the laboratory

mouse which would help achieve the other goals.

The practical benefits to learning about DNA are at least two-fold.

1. In the course of the analysis, disease causing genes would be identified. Also, DNA sequence differences between people can possibly reveal susceptibility to diseases such as cancer. It would perhaps be feasible then for new strategies to be developed for their diagnosis, prevention, therapy and cure.

2. Learning about other nonhuman organisms will help in understanding their natural capabilities; these may help solve challenges in energy sources and even environmental cleanup apart from uses in health care [55]. The genome project is also providing enabling technologies essential to the future of the emerging biotechnology industry, and catalyzing its growth. These technologies will allow us to efficiently characterize the organisms, say in the ocean, with applications such as better fuels from biomass, bioremediation, and waste control. They will also lead to a greater understanding of global cycles, such as the carbon cycle, and the identification of potential biological interventions. See [55] for details.

3.1.1 Reading the genome

What is the problem in reading the DNA sequence? Recall from Section 2.2 that a single DNA molecule is very long! In its native state it is highly coiled and condensed as shown in Figure 2.3. It is inconceivable that a single DNA molecule can be read (or sequenced) at one go and so the following steps are used.

1. Dis-assemble: Break up a single DNA molecule into smaller segments. Mere handling of large DNA molecules breaks them up mechanically in a rather unrepeatable fashion. They can be further broken up into smaller pieces by restriction enzymes (in a repeatable fashion). This is the task of producing clones and clone libraries. See [12] for details.

2. Read/Sequence: At this step, the smaller pieces are handled. The task is so daunting, that as an intermediate first step, it suffices to locate critical sites

along the molecule (called the *Physical Map*): the sites are locations where a restriction enzyme cleaves the DNA molecule. Reading the entire sequence is called *sequencing*, and the information obtained is called the *Sequence Map*.

Classical linkage analysis is used to determine the arrangement of genes on the chromosomes. By tracing how often different forms of two variable traits are co-inherited, we can infer whether the genes for the traits are on the same chromosome: such genes are said to be linked. The *genetic distance*, measured in *centi-morgans*, is a measure of the proclivity of the genes to *crossing-over*[1] (which could very well be just the distance in base pairs!). This map is called the *genetic map*. Of course, the study of this map started off before the inception of *Molecular Genetics*. Nevertheless, they continue to remain an important objective as they determine the locus of the gene or allele of a trait. Figure 3.1 shows a comparison of the maps.

Restriction enzymes *cut* a DNA molecule at certain specific base pair patterns termed the *sites*. A biologist obtains valuable information from the order of the *sites* along with the distances between them. This is called the *Physical Mapping Problem* or the *Ordered Restriction Map Problem*.

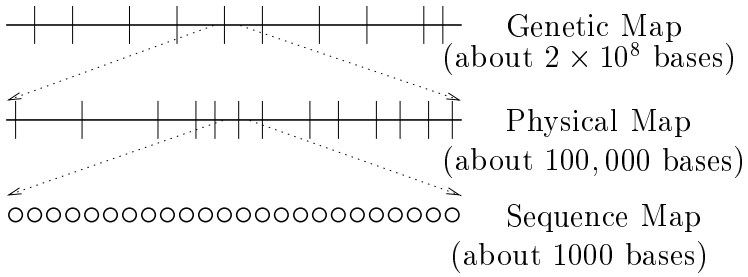


Figure 3.1: The relative "resolution" of each map.

3. Assemble: Now the task is to put the "annotated" (with the location of

[1] Also see Section 2.4.3.

the restriction sites), segments, of the second step, back together again (the inverse of the first step). This is called the contig problem.

3.2 The Physical Map Problem

As we have seen earlier, building physical maps of a chromosomal region is an important step towards the ultimate goal of many efforts in Molecular Biology (including the Human Genome Project), namely to determine the entire sequence of Human DNA and to extract the genetic information from it [28, 12]. A physical map merely specifies the location of some identifiable markers (restriction sites of up to 20 base pairs) along a DNA molecule. Physical maps provide useful information about the arrangement of the DNA, and they serve as recognizable posts to help search it.

3.2.1 Gel-based physical mapping

There are several known technological approaches to building physical maps; each has associated computational problems [1, 21, 31, 47, 46, 35]; most approaches use restriction enzymes. Recall that a *restriction enzyme* is an enzyme that recognizes a unique sequence of nucleotides; it cleaves every occurrence (called a *restriction site*) of that sequence in a DNA molecule (Section 2.4.1).

In a well-established approach to physical mapping, a restriction enzyme is applied to cleave the double stranded DNA molecule at the restriction sites producing pieces of the molecule. The sizes of the restriction fragments, i.e., the number of nucleotides they contain, are measured using *gel electrophoresis* (see Section 2.4.2). However, in this process, the information about their relative positioning is lost. Thus we are faced with the problem of assembling these pieces into their relative order: this leads to difficult combinatorial and computational problems most of which are NP-hard, and many of which have been extensively studied from the point of workable heuristics (See [28, 21, 1] etc. and Section 3 of [48] for several open problems in this area).

A major effort along these lines is the current Multiple Complete Digest (MCD) mapping project at the University of Washington Genome Center [15]. In MCD mapping two or more restriction enzymes are used. Single Complete Digest (SCD) mapping and Double Complete Digest (DCD) mapping are cases where exactly one and exactly two enzymes respectively are used to generate the data. The project uses cosmid clones. We use the DCD problem to illustrate the nature of the underlying computational problems. For the DCD problem, each clone has two sets of fragment lengths (since this is a *double* digest problem). Let the problem have N clones X_i, $1 \leq i \leq N$ and let the fragments associated with each clone be as follows: $X_i = \{\{x^i_{11}, x^i_{12}, \ldots, x^i_{1n^i_1}\}, \{x^i_{21}, x^i_{22} \ldots, x^i_{2n^i_2}\}\}$, where n^i_j, $j = 1, 2$, is the number of fragments using the restriction enzyme numbered j of clone X_i. Since the ordering information is lost, $x^i_{11}, x^i_{12}, \ldots, x^i_{1n^i_1}$ are in no particular order. The task is to obtain a consensus ordering of fragment lengths such that both set of fragments of each clone agree with this ordering. Using different criteria for the "best solution" (optimization) gives rise to interesting computational problems as discussed in [15]. Most of the problems, not surprisingly, are NP-hard and the authors discuss the various heuristic based algorithms they employ to handle the data in practice.

3.2.2 Physical mapping with probes

This is the construction of physical maps using an approach called *STS-content mapping* [12]. In this strategy, each *clone* corresponds to an interval of the chromosome and each *probe* corresponds to a unique point (or a very small interval) on the chromosome. While the order of the clones is unknown, it can be determined whether a probe belongs to a clone or not by *hybridization* (see Section 2.4.5). Thus given a set of clones $C_1, C_2, \ldots, C_n$ and a set of probes $P_1, P_2, \ldots, P_m$, along with information for every pair (C_i, P_j), $1 \leq i \leq n$, $1 \leq j \leq m$, whether probe P_j is present or not in clone C_i, the task is to obtain an ordering of the probes that would indicate the ordering of the markers giving a physical map. The problem is compounded by the presence of false positive and false negative errors in the experiments. See [1] and the references therein for the modeling of the corresponding

computational problems.

3.2.3 Optical mapping

An alternative approach to physical mapping is based on a technology invented by David Schwartz at the W. M. Keck Laboratory for Biomolecular Imaging, Dept. of Chemistry, NYU, called the *Optical Mapping* technology [54, 36, 53, 58]: currently it is a collaborative effort between researchers from various departments including the Departments of Pediatric Genetics, Biology and Computer Science [8]. At a very high level, here is an overview of the method. A single strand of a DNA molecule is attached to the surface of a slide by electrostatic forces. Then it is treated in a controlled manner with a restriction enzyme. The molecule still remains attached to the slide although the restriction sites get digested by the enzyme. Now by applying appropriate fluorescent dyes, the molecule may be viewed under a microscope or recorded by a camera as an image on a computer. For a more detailed description of this complex process, see [58, 36, 54].

As it is clear from our overview of the method, the relative order of the pieces is not lost. In fact, the image itself is a physical map (although perhaps not at desirable levels of resolution, and not in a form compatible with genomic data we handle now). See Figures 3.2 and 3.3 for example images (reproduced from [19]). However, Optical Mapping also faces difficulties which are discussed in Section 4.1.

A vision process identifies the DNA molecules along with the restriction sites and estimates the mass of these molecule fragments. The critical issue in this phase is not just assigning the fragments of the molecule to the right molecule (the problem is compounded by criss-crossing or very closely laid out molecules) but also making an accurate estimate of the mass (or length in terms of base nucleotides) of each fragment. We then formulate the restriction map problem to extract the maps from the estimated fragment masses.

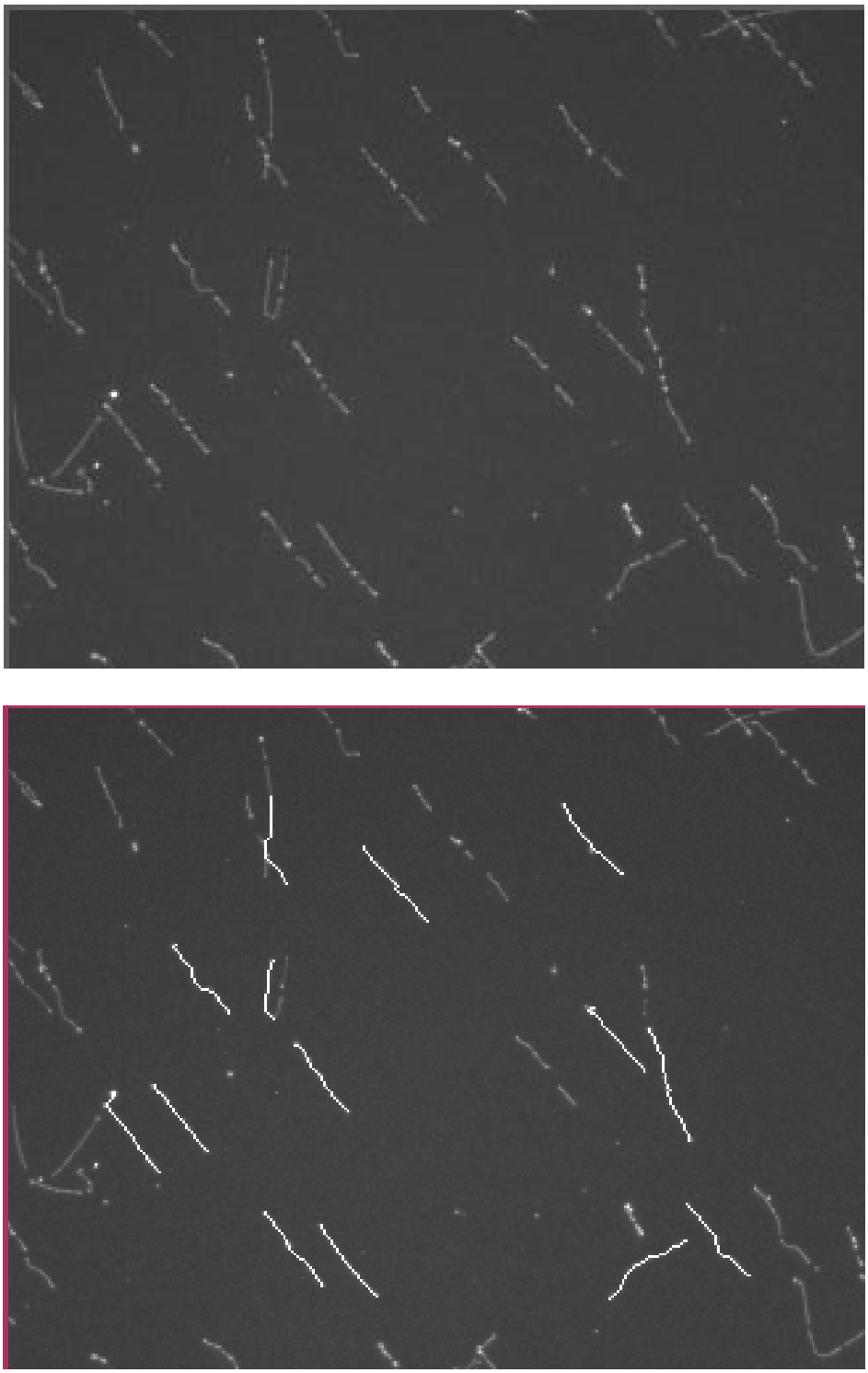


Figure 3.2: A prototypical image reduced by a factor 4 and the detection of its many DNA fragments.

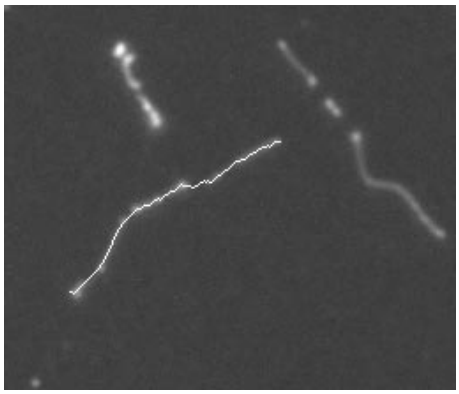


Figure 3.3: The detection of the "backbone" of a DNA fragment is shown as a bright contour running along the length of the molecule.

3.3 On Shotgun Sequencing of the Genome

Recently, there was a shift in the sponsor of the Human Genome Project from the government sponsor National Institutes of Health to a private venture Perkin-Elmer who propose to complete the task of sequencing the human genome in three short years at a cost of only $200 million dollars (as opposed to the earlier $3 billion) [56].

The critical factor in favor of this "audacious" (as reported in New York Times, May 12, 1998) takeover is the increase in the capacity of the capillary-based sequencing machine that can process up to 1000 samples a day with minimal hands-on operator time (about fifteen minutes compared with eight hours for the same number of samples earlier on) [56] (See Section 2.4 for the process of automatic sequencing). The reduction in operation labor coupled with automation makes the task of sequencing the genome with adequate coverage feasible. The proposal is to use BAC clones with 46 times (called 46X) coverage. The next challenge is to assemble the data into contiguous blocks and to assigning these to the correct locations in the genome. They propose to use a set of algorithms called the TIGR Assembler aided by a large number of sequence tagged sites (STS) markers and manual inspection to correct ambiguous or conflicting assembly structures.

3.4 Summary

As can be seen a variety of DNA technologies are being used to attain the goal of sequencing the human genome. It is only a matter of time (three to ten years) before the human genome will be sequenced using either one of the prevalent DNA technologies or a combination of them. However, the underlying biotechnology will be of continuing interest to scientists for sequencing and studying various other non-human organisms.

Chapter 4

A Uniform Framework[1]

We begin this chapter by giving a problem abstraction for the Ordered Restriction Map problem arising in Optical Mapping. It is plausible to tackle this problem using various approaches: in the next section we present a uniform framework for the problem and show that various models reported in literature for this problem are such that each is a specific instance of this basic framework. We achieve this by identifying two "signature" functions $f()$ and $g()$ that characterize the models. We identify the constraints these two functions must satisfy, thus opening up the possibility of exploring other plausible models. We show that for all of the combinatorial models proposed in literature, the signature functions are semi-algebraic. We also analyze a proposed statistical method in this framework and show that the signature functions are transcendental for this model. We believe that this framework will provide useful guidelines for dealing with other inferencing problems arising in practice.

4.1 The Problem Abstraction

In this section, we present a simplified model for the problem in an effort

1. to gain insight into the problem with respect to its computational complexity

[1] This chapter also appears as "A Uniform Framework for Ordered Restriction Map Problems", in the Journal of Computational Biology, 1998 (in press).

and study the interplay of various error sources, and,

2. to develop algorithms (theoretical or practical) for the problem.

We define the Physical Map/ Ordered Restriction Map problem informally as follows: we view this as a game played by Ann and John. John has a string S, of length n, of 0's and 1's. He makes m copies of this string and using some process, *alters* the m copies in some controlled manner. John assures Ann that the number of these alterations is not very large. Now, this altered set of m strings, called the *data set*, is made available to Ann and she is required to guess the original string S John started with. Ann makes a (reasonable) guess by providing an S'. The problem that Ann solves is the Ordered Restriction Map problem.

We now look at the (reasonable) alterations John can make.

False Positives: John can change some 0's to 1's in the m copies. But he must assure Ann that the number of such changes is very small.

In practice, these may be due to actual false cuts or due to errors in the pre-processing stage.

False Negatives: John can change some 1's to 0's in the m copies. But he must assure Ann that the number of such changes is no more than mp_j for each column j. Note that in the absence of this restraint on John (and with False Positives), Ann will have no way of guessing a reasonable S'.

p_j is the *digestion rate* of the experiment or mp_j is the minimum number of 1's required for a column j to be designated a *consensus cut site* [2].

Sizing Errors: John moves the positions of some 1's in a *small* neighborhood, that is, for some integer $\delta > 0$, he can move the position of a 1 in the molecule at j to anywhere between $j - \delta$ and $j + \delta$.

This corresponds to the possible sizing errors of the fragments. The input data does not depict the location of restriction sites accurately because of the error inherent in measuring the lengths of fragments that remain after digestion by the restriction enzyme. Thus a 1 at some site in the molecule might in fact signal

[2] It may be noted that if the number of false positives for column j is mq_j, then Ann cannot make a reasonable guess if the following holds: $p_j + q_j \approx 1$, for any j.

a restriction site in one of its neighbors. This fuzziness is the result of coarse resolution and discretization, other experimental errors, or errors in preprocessing the data prior to constructing physical maps such as in the image processing phase. Orientation Uncertainties: John flips some of the strings: if $s = x_1 x_2 \ldots x_{n-1} x_n$ is a string with $x_i = 0$ or $1, i = 1, 2, \ldots, n$, the flipped string is $x_n x_{n-1} \ldots x_2 x_1$.

When the molecule is laid out on a surface, the left-to-right or right-to-left order is lost. However, the orientation information may be given in the data (using a more elaborate chemical protocol) with a vector arm on one fixed side of the molecule [58]. The model can view this as a consensus cut site at one end of the map. Notwithstanding this, there is a non-zero probability of the orientation of the molecule still being unknown.

The correspondence of the Ann and John game to the Ordered Restriction Map problem is as follows: a string is a molecule, the length of the string corresponds to the number of sites on each molecule, the 1's in the string refer to cuts and the 0's refer to no-cuts at that site. The string S is called the map, and the 1's on S are the consensus cut sites. The changes that John makes correspond to the various experimental and/or pre-processing inaccuracies that creep in at various stages.

Ann is required to produce an S' or a map, which is an n-length string that designates each site as a *consensus cut* site or not. Thus the map enables the following assignments:

1 & 2) Assigning each cut as a true positive or a false positive.
3) Assigning a location to each consensus cut site.

This map gives an *alignment* of the molecules that optimizes a suitable cost function.

Alignment of the rows/molecules refers to assignment of the following:
4) Labeling the orientation of the molecule as *flipped* or not.

Circular Ordered Restriction Map Problem. If John take the string S and glues the two free ends producing a "seamless" ring, the corresponding problem is the circular DNA problem. In this version John makes m altered *rings* (instead

of linear molecules as in the previous case) available to Ann. The seamlessness refers to Ann not having any information about where John glued the ends. The problems in the linear version also appear in the circular configuration and we do not explicitly categorize any of these in the rest of the chapter.

The Cost of an alignment is a function (measure) of the alignment with respect to a map, which we optimize. In the next chapter we explore various forms of "reasonable" cost functions. Recall that a 1 in S' at location j implies that there are at least mp_j 1's in the aligned data set in column j. For the rest of the thesis let $mp_j = c_j$, that is c_j is the *minimum number* of 1's required in column j for it to be a consensus cut column.

4.2 Modeling the problem

Given the problem as described in the last section, we can identify two natural approaches to the problem of inferring an ordered restriction map from a set of erroneous samples: (1) using data consensus or agreement, and, (2) optimizing a characteristic function of the data. We discuss these two approaches in the following sections.

4.2.1 Consensus/Agreement with data

This approach uses the mutual agreement between the molecules to obtain an alignment of the molecules and a map. There are two views to this: one uses an explicit hypothesis and the other does not. We discuss these views in the next two sections.

Consensus/Agreement with a hypothesis

Let hypothesis $\mathcal{H}$ have K restriction sites each at location $l_j, j = 1, 2, \ldots K$. As the location of a site is not exact, assume that it has a distribution $G_j()$ about the correct location l_j in $\mathcal{H}$, with standard deviation σ_j. Further assume that given $\mathcal{H}$ and a molecule i with some fixed alignment, we can designate every cut site in the

molecule as *true* or *false*. A *true* site will correspond to l_j of $\mathcal{H}$, for some j, at a small distance d_j from it, and, a *false* site will have no such correspondence. For an alignment of the rows/molecules define the following:

$$T_j = \sum_{i=1}^{m} (\text{number of } \textit{true} \text{ sites in molecule } i \text{ at } j), \tag{4.1}$$

$$F_j = \sum_{i=1}^{m} (\text{number of } \textit{false} \text{ sites in molecule } i \text{ at } j). \tag{4.2}$$

Let $F = \sum_j F_j$. Then $\mathcal{M}(\mathcal{H}, i)$, the match for an alignment of molecule i with hypothesis $\mathcal{H}$ is defined as

$$\mathcal{M}(\mathcal{H}, i) = \sum_j \left\{ f_j(T_j) G_j(d_j) + g_j(F_j) \right\}, \tag{4.3}$$

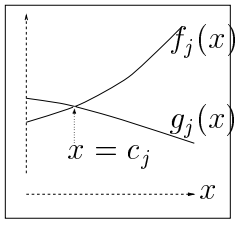
and the problem is to maximize $\mathcal{M}$. $f_j()$ and $g_j()$ are "suitable" functions on the number of *true* and *false* sites respectively at a location j, depending on whether j is a *consensus cut site* or not in the hypothesis $\mathcal{H}$. We call these functions the signature functions of the model. An alternate form of $\mathcal{M}$ can be obtained by defining function $\tilde{g}()$ on F instead of F_j in equation (4.3). See Section 4.3 for an example.

Agreement with a hypothesis uses the following optimization function:

$$\max_{(\text{over all hypotheses } \mathcal{H} \ \& \ \text{alignments})} \left\{ \sum_{i=1}^{m} \mathcal{M}(\mathcal{H}, i) \right\}. \tag{4.4}$$

Properties of functions $f_j()$ and $g_j()$. What must be the conditions on $f_j()$ and $g_j()$ (or $\tilde{g}()$) so that the "mutual agreement" of the molecules (or data) is not violated?

1. $f_j(x) < f_j(y), \quad \forall x < y, \text{ and } x, y \geq c_j.$
2. $g_j(x) \leq g_j(y), \quad \forall x > y, \text{ and } x, y \leq c_j.$
3. $g_j(x) > f_j(x), \quad \forall x < c_j,$
4. $g_j(x) < f_j(x), \quad \forall x > c_j.$



The first condition states that the "agreement" must increase with increase in matches; the second condition has the same spirit. These conditions ensure that a consensus cut column at j has at least c_j 1's. Hence one can see that the cost functions can be "designed" using the constants c_j (or probabilities p_j) by defining appropriate $f_j()$ and $g_j()$ that satisfy the above conditions.

Let d_j in equation (4.3) be very small, for all j. This leads to the *idealized* version of the problem, where the location of a site is supposed to be exact; thus the molecules can be represented as a string of 0's and 1's. Thus the data can be represented in a $m \times n$ binary matrix $[M_{ij}]$ with each entry as either 0 or 1. Each row represents a molecule and each column refers to a site on the molecule: thus there are m molecules and n sites. A 1 at position (i, j) means that the jth site (column) of the ith molecule (row) is a cut. A 0 indicates the absence of a cut. A hypothesis is a map or a n-length vector with 1's representing a consensus cut site and 0 representing its absence. Let $D = \{p_1, p_2, \ldots, p_n\}$ be the digestion rates of the locations $j = 1, 2, \ldots, n$. The different problems that respect the "mutual agreement" criteria appear in the following sections.

Problem Instances. Binary Flip Cut (BFC) [38, 18]: Given M_{ij} and the digestion rates D, find an alignment of the rows/molecules and a map that maximizes the number of 1's in the consensus cut columns (which is at least mp_j for the consensus cut column j). The alignment takes orientation uncertainties into account; incorporating the missing fragment errors gives rise to Binary Shift Cut (BSC) problem [3, 14, 43], and incorporating the good/spurious molecule error gives Binary Partition Cut (BPC) problem [3, 43]. The signature functions for these problems are:

$$f(x) = x - c_j \quad \text{and} \quad g(x) = 0,$$

where c_j is the minimum number of 1's or cuts required in a column j for it to be a consensus cut column.

Exclusive Binary Flip Cut (EBFC) [38, 18, 41]: Given M_{ij} find an alignment that maximizes the number of 1's in consensus cut columns where only one of j or $\overline{j} = n - j + 1$ is a consensus cut and further the column with the higher number

of 1's (between j and $\overline{j}$) is the consensus cut.

The EBFC problem can be defined alternately in terms of the digestion rate, $p_j = c_j/m$. Note that this is an equivalent definition of the EBFC problem:

$$\begin{aligned} P_j &= |\{i | M_{ij} = 1 \text{ AND } M_{i\overline{j}} = 1\}| \\ \bar{P}_j &= |\{i | M_{ij} = 1 \text{ XOR } M_{i\overline{j}} = 1\}| \\ c_j &= P_j + \frac{\bar{P}_j}{2} \end{aligned} \tag{4.5}$$

The signature functions for this problem are:

$$f(x) = x - c_j \quad \text{and} \quad g(x) = c_j - x,$$

where c_j is as defined above.

Balanced BFC: Given M_{ij}, find an alignment of the rows/molecules and a map that maximizes the total number of 1's in the consensus cut columns and the number of 0's in the columns that are not consensus cut columns. We can model each of the other error sources similarly. The signature functions for these problems are:

$$f(x) = 2x - m \quad \text{and} \quad g(x) = m - 2x,$$

where m is the number of rows in the matrix or the number of molecules. Thus, in these problems $c_j = m/2$.

Conservative BFC: This gives a conservative evaluation of the cost per consensus cut column and is defined as follows. Given M_{ij}, find an alignment of the rows/molecules and a map that maximizes the number of 1's less the number of 0's in the consensus cut columns. In a similar spirit as before, we can model each of the other error sources as well. The signature functions for these problems are:

$$f(x) = x \quad \text{and} \quad g(x) = m - x,$$

where m is the number of rows in the matrix or the number of molecules. Thus, in these problems $c_j = m/2$.

It is interesting to note that problems with linear $f()$ such as EBFC, BFC and others give rise to inapproximable combinatorial problems as discussed in Sec-

tion 8.2.2. However, attempting to simplify the cost function any further trivializes the problem as we show in Lemma 1 below.

Lemma 1 *If $f_j(x)$ is a linear function and $f_j(x) = g_j(x)$, $0 \leq x \leq m$, $\mathcal{M}()$ is a constant function.*

Proof: Under these conditions, every column (irrespective of the alignments of the rows) can either be or not be a consensus cut column without affecting the cost. If $f_j(x) = \alpha x + \beta$, for some $\alpha > 0$, then the cost function is always $\alpha A + n\beta$ where A is the number of 1's in the input matrix and n is the number of columns. (Note that this is not true if the functions are not linear since $h(a+b) \neq h(a) + h(b)$ where $h()$ is a non-linear function.) QED

Consensus/Agreement without (explicit) hypothesis

In this section we focus on the approach which can be broadly described as guessing the correct alignment of molecules by studying a few molecules (say $d \geq 2$) at a time and building the entire solution from this (possibly with some back-tracking). We formalize the problem as the d-wise Match (dM) problem. Let the number of molecules be m, each having n sites. For a fixed d ($d \geq 2$ and $d << m$), we assume that we can orient the d molecules so that they have maximum agreement between them. This is done by enumerating all the 2^{d-1} possible configurations where each molecule can have left-to-right or right-to-left orientation with respect to a reference molecule whose orientation is fixed. This assigns an orientation to each molecule of the sample size of d molecules. We associate a cost with each configuration of the d molecules, X, as $A^X(i_1, i_2, \ldots, i_d)$, and define the cost as follows, for some fixed $\delta > 0$:

$$A^X(i_1, i_2, \ldots, i_d) = \begin{array}{l} \text{\# of cut sites that are within } \delta \text{ of each other in} \\ \textit{all} \text{ the } d \text{ molecules, given } X. \end{array} \tag{4.6}$$

Another simple extension to this cost function is

$$A^X(i_1, i_2, \ldots, i_d) = \left\{ \begin{array}{l} \text{\# of cut sites that are within } \delta \text{ of each other } - \\ \text{\# of the remaining cuts, in } \textit{all} \text{ the } d \text{ molecules,} \\ \text{given the configuration } X. \end{array} \right\} \tag{4.7}$$

and in principle, this alignment could model other errors as well. Given the sample of d molecules, we assign the configuration $X_{\min}$ that maximizes the cost (defined by equation (4.6)). This configuration $X_{\min}$ implicitly assigns an orientation to each of the d molecules. Thus, orientation can be assigned to each molecule of every possible d-sized sample of the m molecules. If the orientation of any one of the molecules is changed, the cost associated with the molecules increases by say δ. Also, a molecule belongs to $\binom{m}{d-1}$ samples and could have different orientations assigned to it in the different samples. The aim is to assign an orientation to *every* molecule, so that the sum of the deviation δ from the optimal in each of the $\binom{m}{d}$ samples is minimized, or, the cost of alignment due to each of the samples is maximized. This optimization problem is termed the d-match (dM) problem. It is assumed that once the orientation of each molecule is known, the positions of the consensus cut sites can be estimated quite simply.

We give the following lemmas to identify the signature ($f()$ and $g()$) functions of this model.

Lemma 2 *Matching with d-wise agreement is equivalent to the following optimization problems.*

$$\max_{\textit{(over all configurations)}} \sum_j \binom{T_j}{d}, \ \textit{using equation (4.6)}$$

and,

$$\max_{\textit{(over all configurations)}} \sum_j \left(2\binom{T_j}{d} - \binom{m}{d} \right), \ \textit{using equation (4.7)} \tag{4.8}$$

where T_j represents the number of cuts at the position j in that configuration.

Proof: Consider a configuration. The number of matches per column is $\binom{T_j}{d}$. Using equation (4.6), we need to maximize this over all the columns, hence the result.

The number of mis-matches is $\binom{m}{d} - \binom{T_j}{d}$. Thus, using equation (4.7), we need to maximize the following

$$\binom{T_j}{d} - \left(\binom{m}{d} - \binom{T_j}{d} \right), \tag{4.9}$$

hence the result. QED

Corollary 1 *The pair-wise matching problem is equivalent to the following optimization problem:*

$$\max_{over\ all\ configurations} \left(\sum_j T_j(T_j - 1) \right),$$

where T_j represents the number of cuts in the position j in that configuration.

For the sake of simplicity, we study the *pairwise* or 2-wise match problem which is as follows. Given m molecules with n sites each, with false positive and negative errors and orientation uncertainties, and a fixed $\delta > 0$, find an alignment to the molecules so that it has the maximum 2-*wise match* where $A^X(i_1, i_2)$ is defined as

$$A^X(i_1, i_2) = \begin{array}{l} \#\text{ of cut sites that are within } \delta \text{ of each other in both the} \\ \text{molecules, given } X, \end{array} \tag{4.10}$$

and X denotes an alignment, i.e., (1) both are in the same orientation, say left-to-right, or (2) they have an opposite orientation, say one is left-to-right and the other is right-to-left. Thus it is the following optimization problem:

$$\max_{(\text{over all alignments})} \left\{ \sum_{i_1=1}^{m} \sum_{i_2 \neq i_1}^{m} A^X(i_1, i_2) \right\}. \tag{4.11}$$

Informally, the task is to *maximize the sum of the pairwise match cost.*

For the $d = 2$ case, we map the problem to a graph problem. Notice that if there are only two molecules, there can be only one alignment (either both are in the same orientation or one of them is in the opposite orientation) and there is no conflict. If there are 3 molecules, it is possible that considering two of them assigns an orientation to each of the molecules and the third molecule may or may not support this decision. In general for n molecules we capture this in a graph structure as described below.

Given a 2-wise match problem, a complete graph $\mathcal{G}$ is constructed with every vertex v_i corresponding to a molecule i.

Edge Labels. Let $X = S$ denote an alignment where both the molecules i and

j have the same orientation, and $X = O$ denote the alignment where one of them is is left-to-right while the other is right-to-left. Every edge $e_{ij} = v_i v_j$ with $A^S(i,j) \neq A^O(i,j)$ is labeled by label $L(v_i v_j)$ as follows:

$$L(e_{ij}) = \begin{cases} \textit{Same} & A^S(i,j) > A^O(i,j), \\ \textit{Opposite} & A^S(i,j) < A^O(i,j). \end{cases}$$

Recall that $A^X(i,j)$ is the cost of the alignment X using equation (4.10). We remove those edges that have $A^S(i,j) = A^O(i,j)$; thus the graph $\mathcal{G}$ is not necessarily a complete graph.

Edge Weights. Weight $Wt(e_{ij})$ is defined as $Wt(e_{ij}) = \max(A^S(i,j), A^O(i,j))$.

We define a predicate on any set of three vertices (of the complete graph) as follows: v_i, v_j, v_k are *consistent* if either all three edges or exactly one edge is labeled *Same*.

It can be verified that the three molecules can be assigned unique orientations only if all of the three pairwise labels are *Same* or exactly two of the pairwise labels are *Opposite*. Thus a consistent set of vertices give a unique alignment to the corresponding set of molecules. Further, a labeled graph $\mathcal{G}$ is said to be *consistent* if every three vertices v_i, v_j, v_k is consistent.

Given the labeled and edge weighted graph $\mathcal{G}$, the problem is to obtain a set of edges $\mathcal{S}$, such that for all edges $e \notin \mathcal{S}$ the labels of the edges is changed from *Same* to *Opposite* or vice-versa, so that (1) the graph with these new labels is consistent, and, (2) the sum of the weights of the edges $e \in \mathcal{S}$ is maximized. This is called the Weighted Consistency Graph (WCG) problem.

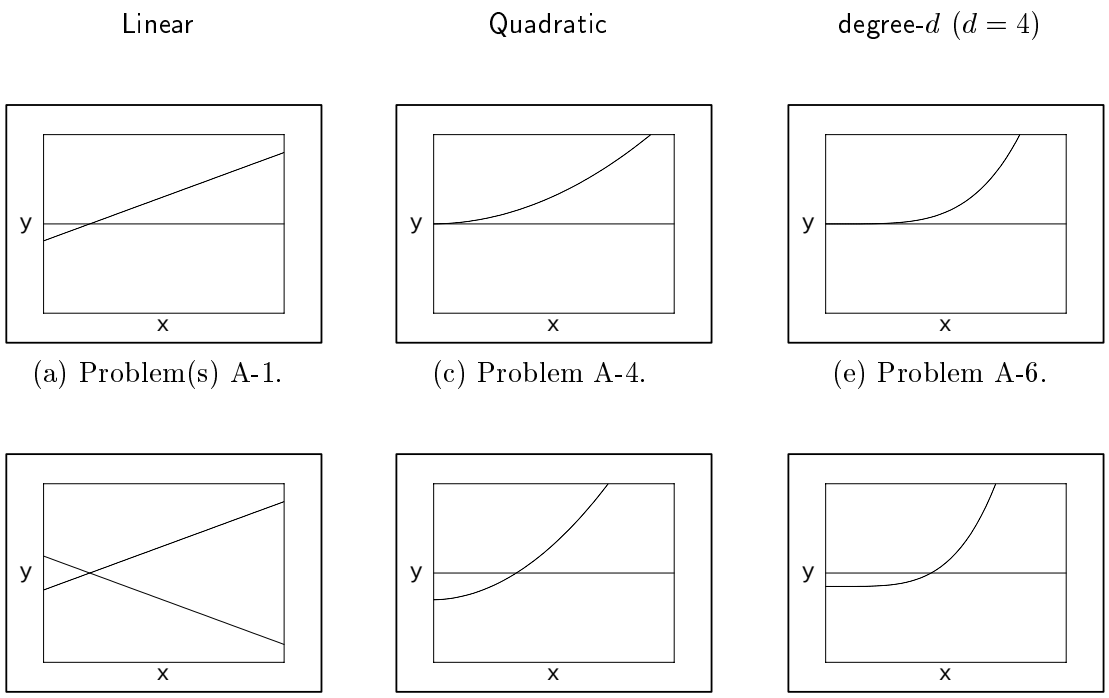
We define a special case of the WCG problem, the Consistency Graph (CG) problem: Given a labeled graph $\mathcal{G}$, find the maximum number of edges that retain the labels to get a consistent graph.

Table 4.1 summarizes these problems and their signature functions and Figure 4.1 plots these functions for convenience.

Problems using data consensus/agreement

	Problem	d_j	Errors	$f(x)$	$g(x)$	c_j
				linear form		
A-1	Excl. Binary Flip Cut (EBFC) [18, 41]	fixed	Orien.	$x - c_j$	0	implicit (eqn 4.5)
	Binary Flip Cut (BFC) [18, 41, 43]					user defined
	Binary Shift Cut (BSC) [3, 14]		Missing frags.			
	Binary Partition Cut (BPC) [3, 43]		Bad mols			
	Weighted Flip Cut (WFC) [41]	any	Orien., sizing			
	Binary Sizing-error Cut (BSeC) [3]		sizing errors			
A-2	Balanced probs	fixed	all	$2x - m$	$m - 2x$	implicit $(m/2)$
A-3	Conservative probs			x	$m - x$	
	Weighted Consistency Graph (WCG)			quadratic form		
A-4	Pairwise match under $\mathcal{M}_1$ (eqn 4.6)	$\leq \delta$	all	$\binom{x}{2}$	0	No c_j
A-5	Pairwise match under $\mathcal{M}_2$ (eqn 4.7)			$2\binom{x}{2} - \binom{m}{2}$		implicit (eqn 4.9)
	d-wise match			degree-d		
A-6	under $\mathcal{M}_1$ (eqn 4.6)	$\leq \delta$	all	$\binom{x}{d}$	0	No c_j
A-7	under $\mathcal{M}_2$ (eqn 4.7)			$2\binom{x}{d} - \binom{m}{2}$		(eqn 4.8)
				transcendental form		
A-8	Statistical Model [3]	$\leq \delta$	all	$x \ln(\frac{x}{m}) + (m - x) \ln(1 - \frac{x}{m})$	$\tilde{g}(x) = x \ln \frac{x}{m} - x$	implicit

Table 4.1: Problems using the consensus/agreement criterion (identified primarily by monotonically increasing $f_j()$). All the problems also model false positive and false negative errors. Recall that $c_j = mp_j$ is the minimum number of 1's or cuts required in a column j for it to be a consensus cut column.



(a) Problem(s) A-1. (c) Problem A-4. (e) Problem A-6.

(b) Problems A-2 to A-3. (d) Problem A-5. (f) Problem A-7.

Figure 4.1: The plots of the $f_j(x)$ and $g_j(x)$ functions for the different problems using the consensus/agreement criterion. $f_j(x)$ is the increasing function in all plots. $g_j(x)$ is the decreasing function in (b) and the horizontal line in the others. In each case, c_j is given by the zero of the polynomial $f_j(x) - g_j(x)$. (To create the plots certain values of m were assumed; since the purpose of these plots is to indicate the shape of these curves, we skip the other details that are not vital to the study of these functions.)

4.2.2 Optimizing the characteristic of an alignment

We would like to identify reasonable characteristics of an alignment that can be quantified and formulate the problem as optimizing this quantity. We give some such formulations in the following sections.

Optimizing K, the number of consensus cuts

Minimizing the number of consensus columns ($\mathsf{BFC}_{\min K}$): Given M_{ij} and the digestion rates S, find an alignment of the rows/molecules and a map that minimizes the number of consensus cut columns. In a similar spirit one can define $\mathsf{BSC}_{\min K}$ and $\mathsf{BPC}_{\min K}$. The signature functions for this model are:

$$f(x) = \begin{cases} 1 & x < c_j \\ 0 & \text{otherwise} \end{cases} \qquad \text{and} \quad g(x) = 0.$$

One can see that this is not a good model since the signature functions do not satisfy properties (1), (3) and (4) described in Section 4.2.1.

Maximizing the number of consensus columns ($\mathsf{BFC}_{\max K}$): Given M_{ij} and the digestion rates S, find an alignment of the rows/molecules and a map that maximizes the number of consensus cut columns. This cost function was suggested in [3]. In a similar spirit one can define $\mathsf{BSC}_{\max K}$ and $\mathsf{BPC}_{\max K}$. The signature functions for this model are:

$$f(x) = \begin{cases} 0 & x < c_j \\ 1 & \text{otherwise} \end{cases} \qquad \text{and} \quad g(x) = 0.$$

Notice that the $f()$ function for both these models is a step function.

Optimizing discrepancy

Define a *discrepancy* d_j, between a column j and its conjugate $\overline{j}$ as the difference in the number of 1's between the column j and $\overline{j}$, given an alignment. Now, different cost functions can be defined based on d_j. Note that discrepancy is meaningful only in the context of EBFC problems.

1. $\mathsf{EBFC}_{\max,\Sigma}$, $\mathsf{EBFC}_{\min,\Sigma}$: Obtain an alignment and a map that maximizes the sum of the discrepancies. $\mathrm{EBFC}_{\min,\Sigma}$ can be defined in a similar manner. It can be verified that $\mathrm{EBFC}_{\max\,\Sigma}$ attains its optima at the same alignment as EBFC (thus these two problems are identical). The signature functions for $\mathrm{EBFC}_{\max,\Sigma}$ are:

$$f(x) = x - c_j \quad \text{and} \quad g(x) = c_j - x,$$

where c_j is defined by equation (4.5).

$\mathrm{EBFC}_{\min,\Sigma}$ is a dual of the $\mathrm{EBFC}_{\max,\Sigma}$ problem and attains the the optima at the same configuration as that of $\mathrm{EBFC}_{\max,\Sigma}$. Thus in a sense EBFC, $\mathrm{EBFC}_{\max,\Sigma}$ and $\mathrm{EBFC}_{\min,\Sigma}$ are equivalent problems!

2. $\mathsf{EBFC}_{\min,\max}$, $\mathsf{EBFC}_{\max,\min}$: Obtain an alignment and a map that minimizes the maximum of the individual ($j = 1,2, \ldots,$ n/2) discrepancies. $\mathrm{EBFC}_{\max,\min}$ is defined in a similar manner. Note that the former attempts to concentrate the 1's in the consensus cut columns while the latter attempts to distribute the 1's as evenly as possible.

   The signature functions for both these models are:

$$f(x) = x \quad \text{and} \quad g(x) = m - x.$$

   However the optimizing functions for $\mathrm{EBFC}_{\min,\max}$ and $\mathrm{EBFC}_{\max,\min}$ are

$$\min_j \left\{ \max \left\{ \mathcal{M}_j, \mathcal{M}_{\bar{j}} \right\} \right\} \quad \text{and} \quad \max_j \left\{ \min \left\{ \mathcal{M}_j, \mathcal{M}_{\bar{j}} \right\} \right\}$$

   respectively where $\mathcal{M}_j = f_j(T_j)G_j(d_j)+g_j(F_j)$ and $\bar{j}$ is the conjugate position of j.

3. $\mathsf{EBFC}_{\max,\max}$, $\mathsf{EBFC}_{\min,\min}$: These can be defined in the same spirit but do not appear to be interesting functions and they are very simple to compute.

These forms are summarized in Table 4.2.

Problems using characteristic optimization

	Problem	Errors Modeled	Cost Function		On c_j
			$f_j(x)$	$g_j(x)$	
			step function		
B-1	Minimizing K	all errors	$\begin{cases} 1 & x < c_j \\ 0 & \text{otherwise} \end{cases}$	0	user defined
B-2	Maximizing K	all errors	$\begin{cases} 0 & x < c_j \\ 1 & \text{otherwise} \end{cases}$	0	user defined
			linear function		
B-3	$\text{EBFC}_{\max,\Sigma}$	Orientation uncertainties	$x - c_j$	$c_j - x$	implicit (eqn 4.5)
			x	$m - x$	
B-4	$\text{EBFC}_{\min,\max}$	orientation uncertainties	$\min_j \left\{ \max \left\{ \mathcal{M}_j, \mathcal{M}_{\bar{j}} \right\} \right\}$		implicit (eqn 4.5)
B-5	$\text{EBFC}_{\max,\min}$		$\max_j \left\{ \min \left\{ \mathcal{M}_j, \mathcal{M}_{\bar{j}} \right\} \right\}$		

Table 4.2: Classification of problems using the optimization of a characteristic of an alignment. $\mathcal{M}_j = f_j(T_j)G_j(d_j) + g_j(F_j)$ and $\bar{j}$ is the conjugate position of j. Notice that the the cost function for the $\text{EBFC}_{\min,\max}$ and $\text{EBFC}_{\max,\min}$ problems is different from equation (4.4) and is shown in the table; however the "signature" functions are linear.

4.3 Analysis of a Statistical Approach

A statistical model for the restriction map problem is presented in [3]: we will analyze this model in the context of our framework and show the following: (1) the signature functions $f()$ and $g()$ take transcendental forms in this model (problem A-8 in Table 4.1), and (2) make some observations about the model and suggest possible modifications.

Signature functions $f()$ and $g()$. We will not give a complete definition and description of the proposed model here. The reader is advised to look at [3] for notation and other details.

We will use the same notation as in [3]. $Pr[D_j^{(k)}|\mathcal{H}, good]$ denotes the probability of the molecule j aligned (alignment index k) with the hypothesis $\mathcal{H}$ and molecule j being a good (not spurious) molecule. $Pr\left[D_j|\mathcal{H}, A_{jk}\right]$ denotes the probability of the molecule j aligned (alignment index k) by A_{jk} with the hypothesis $\mathcal{H}$. $c_1, c_2, \ldots, c_N$ are the consensus cuts and $s_1, s_2, \ldots, s_N$ are their locations in the hypothesis $\mathcal{H}$. $h_1, h_2, \ldots, h_N$ denote the distances of each of these cuts, if they exist, in the molecule j from the exact locations under the alignment A_{jk}. M is the number of molecules and N is the number of consensus sites in the molecule. λ_f is the expected number of false cuts. F_{jk} is the number of false cuts in the data D_j for the alignment A_{jk} that do not match any hypothesis $\mathcal{H}$. Using this notation, at the heart of the model is the following definition (equation (1) in [3], which we reproduce here):

$$Pr[D_j^{(k)}|\mathcal{H}, good] = \left[\prod_{i=1}^{N}\left(p_{c_i}\frac{e^{-(s_{ijk}-h_i)^2/2\sigma_i^2}}{\sqrt{2\pi}\sigma_i}\right)^{m_{ijk}}(1-p_{c_i})^{1-m_{ijk}}\right]\times\lambda_f^{F_{jk}}e^{-\lambda_f} \quad (4.12)$$

However, for the sake of ease of understanding, we use the example that the authors use to explain the above formula (which we again produce here) shown below:

$$Pr\left[D_j|\mathcal{H}, A_{jk}\right] = p_{c_1}\frac{e^{-(s_1-h_1)^2/2\sigma_1^2}}{\sqrt{2\pi}\sigma_1}\times(1-p_{c_2})\times\lambda_f e^{-\lambda_f}\times\ldots\times p_{c_N}\frac{e^{-(s_N-h_N)^2/2\sigma_N^2}}{\sqrt{2\pi}\sigma_N} \quad (4.13)$$

This defines the underlying cost function for the statistical algorithm where $Pr_{jk} = Pr\left[D_j|\mathcal{H}, A_{jk}\right]$. To understand this function better, consider the special case where

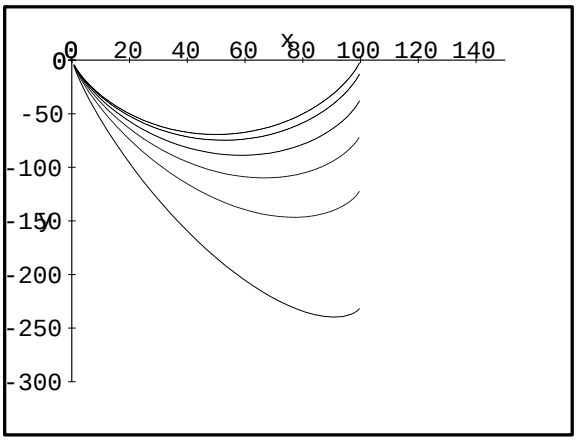


Figure 4.2: The plot of $f(x) = x\ln(\frac{a*x}{M}) + (M-x)\ln(1-\frac{x}{M})$ with $M = 100$ and $a = 0.1, 0.3, 0.5, 0.7, 0.9, 1.0$. The "perfect" cup shape is obtained when $a = 1.0$; the curve with the lowest dip corresponds to $a = 0.1$.

every molecule is *good*. Then, the cost function that equation (4.13) suggests can be viewed as form (4.4). In [3] the log-likelihood, $\mathcal{L}$, which is to be maximized, is computed as (we again reproduce here):

$$\mathcal{L} \equiv \sum_j \ln Pr[D_j|\mathcal{H}]. \tag{4.14}$$

Note that in the special case where every molecule is *good*, by the definition of $Pr[D_j|\mathcal{H}]$ in [3], the following holds:

$$Pr[D_j|\mathcal{H}] = \frac{1}{2}\sum_k Pr[D_j|\mathcal{H}, A_{jk}]. \tag{4.15}$$

Under the assumption that, when a molecule is matched against a hypothesis, the alignments that do not match give a very low probability, that is $P_{jk} \approx 0$, $\mathcal{L}$ corresponds to $\mathcal{M}$ of equation (4.4). Now, to understand how this function behaves, consider the special case where G_l has a constant σ_l, for all l; for different values of the constant we get plots of the function as shown in Figure 4.2 (using the

equation (4.16) discussed below). For further discussion, we choose the curve which corresponds to $G_l() = 1$ [3]. Let $x = \sum_{j=1}^{M} m_{ijk}$ for a configuration. Under alignment A_k, m_{ijk} is the indicator variable for molecule j and cut i of the hypothesis $\mathcal{H}$. Since p_{c_i} is the probability of cut i, $p_{c_i} = x/M$. Let $y = \sum_{j=1}^{M} F_{jk}$, where F_{jk} is the number of false cuts in data D_j. By definition, $\lambda_f = y/M$,

Notice that equation (4.12) uses indicator variables s_{ijk} and m_{ijk} and a count variable F_{jk} that gives the number of false cuts per molecule j for the alignment A_{jk} (although s_{ijk} does not appear explicitly in the equation it is nevertheless used, see [3] for details). The number (and the values) of the variables s_{ijk} and m_{ijk} depend on the number of cuts in the hypothesis $\mathcal{H}$ and the value of F_{jk} depends on decisions whether cuts in a molecule correspond to cuts in the hypothesis $\mathcal{H}$ or not. Hence we can effectively discretize the model depending on the number and values of these variables.

As $p_{c_j} = e^{\ln p_{c_j}}$, if there are x molecules showing a cut at location j (thus $M - x$ that do not) for an alignment A_k, we have (using equation (4.14)),

$$\begin{aligned} f(x) &= \ln\left(\left(e^{\ln p_{c_j}}\right)^x \left(e^{\ln(1-p_{c_j})}\right)^{M-x}\right) \\ &= x \ln p_{c_j} + (M-x)\ln(1-p_{c_j}) \\ &= x\ln(\tfrac{x}{M}) + (M-x)\ln(1-\tfrac{x}{M}) \end{aligned} \tag{4.16}$$

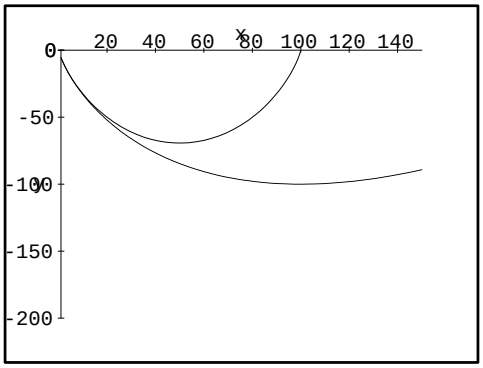
assuming $G_j() = a = 1.0$ (see Figure 4.2). Similarly, if there are y false cuts, then,

$$\begin{aligned} \tilde{g}(y) &= M\left(\ln e^{-\lambda_f}\right) + y\ln\lambda_f \\ &= -M\lambda_f + y\ln\lambda_f \\ &= y\left(\ln\tfrac{y}{M} - 1\right) \end{aligned} \tag{4.17}$$

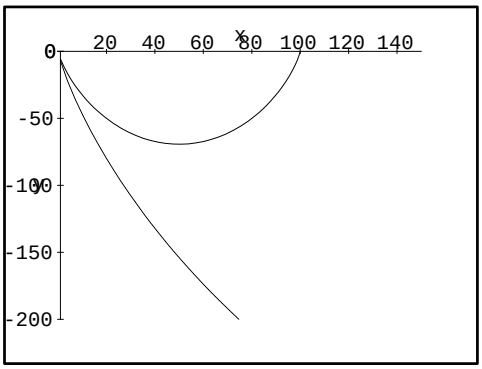
Observations. Having the optimizing function in sharp focus helps us observe some properties of the model.

- $f(x)$ is such that $f(a+b) < f(a) + f(b)$, for $a + b < M/2$ and $f(a+b) > f(a) + f(b)$ for $a + b > M/2$. Thus for $a + b > M/2$, the model prefers to put

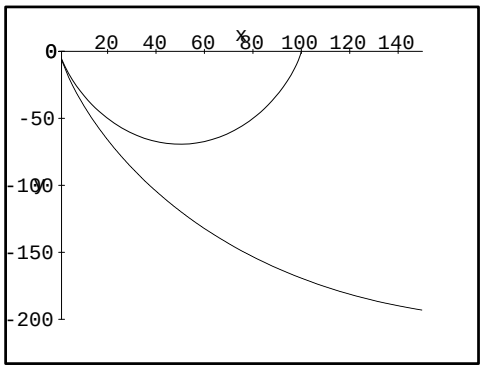
[3] Notice that if $G_l() = a = 0.1$ is used, the $\tilde{g}()$ curve in Figure 4.3 is such that $f(x) < \tilde{g}(x)$, for all $x > \delta > 0$, where δ is a very small number, which is not desired.



$(a) k = 1.0$



(b) $k = 0.25$



(c) $k = 0.5$

Figure 4.3: The plots of the $f(x)$ and $g(x)$ functions for the statistical model ($f(x)$ is the cup-shaped curve and the other is the $g(x)$ curve). (a) shows the $f()$ and $\tilde{g}()$ functions for equation (4.12) and (b) and (c) show the $f(x)$ and $g(x)$ functions for equation (4.18).

the 1's into one consensus site and for $a + b < M/2$ it tries to distribute the cuts between j and its conjugate $\bar{j}$.

Thus if the digestion rate is $< 50\%$, then the model will attempt to maximize the number of cuts (spread the true positives to its conjugates) and for a digestion rate $> 50\%$ will minimize the number of cuts (concentrate the cuts along one column). This dichotomy is due to the cup-shaped $f()$ as shown in Figure 4.3 (which also violates property 1 of the $f()$ function described in Section 4.2.1, for digestion rates $< 50\%$).

- For digestion rates lower than approximately 20%, it is conceivable that the model does not effectively distinguish between true positives and false positives. Using some straightforward calculations, we can show that $f()$ and $g()$ functions are such that for $x < v$, $f(x) \approx g(x)$, for $v = 0.2$. (Also see Figure 4.3.) This problem can be circumvented by modeling the false cut at different locations separately, and, using a term $k_j \leq 1$ that ensures that the digestion rate is at least $p_j, 0 < p_j \leq 1$.

  For clarity, we skip the subscript j of k_j, p_j in the rest of the discussion. Now equation (4.13) can be modified as:

$$Pr_{jk} = p_{c_1} \frac{e^{-(s_1 - h_1)^2/2\sigma_1^2}}{\sqrt{2\pi}\sigma_1} \times (1 - p_{c_2}) \times k\lambda_{f_l} e^{-\lambda_{f_l}} \times \ldots \times p_{c_N} \frac{e^{-(s_N - h_N)^2/2\sigma_N^2}}{\sqrt{2\pi}\sigma_N} \quad (4.18)$$

  k can be viewed as a factor that "discourages" λ_{f_l} to be large and the extent of this "discouragement" is governed by equation (4.20) as shown below. Thus equation (4.17) modifies as follows [4]:

$$g(y) = y\left(\ln \frac{ky}{M} - 1\right). \quad (4.19)$$

  k is such that

$$g(x) < f(x), \text{ for } x > pM. \quad (4.20)$$

  Figure 4.3 shows a plot of the functions $f()$ and $g()$ for different values of k.

This concludes the analysis of the statistical model in our framework.

[4] Note that equation (4.17) defines $\tilde{g}()$, but this defines $g_j()$ due to modification (1).

Chapter 5

Computational Complexity

We begin by giving a brief overview of computational complexity of optimization problems. Then we formulate the problems using the two approaches of the last chapter and discuss their computational complexity. We postpone the discussion on the algorithms for the problems, both theoretical and practical, to Chapters 6 and 7.

5.1 On Complexity of Optimization Problems

The theory of NP-Completeness developed by [11, 27, 33] show that many decision problems of interest are NP-complete. A natural decision version of optimization problems such as the traveling salesperson problem, maximum clique and others are also NP-complete, implying that the optimization versions are NP-hard. Thus if P $\neq$ NP, no polynomial time algorithm can give the optimal solution to these problems.

In this context, it is meaningful to ask the question whether there exists a polynomial-time approximation algorithm for an NP-hard problem. An algorithm is said to approximate a solution within a factor of ϵ, where $\epsilon \geq 1$, if the algorithm guarantees a solution whose cost is within a factor of ϵ of the optimum, for every instance of the problem. It has been shown that NP-hard optimization problems can be classified further depending on the extent of approximability guaranteed by

the algorithm [26, 52, 5]. Some NP-hard problems such as knapsack problem [17] have a *Fully Polynomial Time Approximation Scheme (FPTAS)*, i.e., for any $\epsilon > 0$, the algorithm guarantees a solution that is within a factor of $1 + \epsilon$ in time that is polynomial in the input size and $1/\epsilon$. If the algorithm is polynomial time in the input size only (but depends arbitrarily on $1/\epsilon$), then the algorithm belongs to a scheme called the *Polynomial Time Approximation Scheme (PTAS)*. Most problems of interest are not known to admit a PTAS. However, many of these problems, such as maximum cut, vertex cover, Eucledian Steiner Tree, metric traveling salesperson have a constant factor approximation algorithm. The class of these problems is called APX. The unrestricted traveling salesperson problem is not in APX [52] and neither is the clique problem [7].

In [40], the authors use second order logic to define a class of NP optimization problems called MAX SNP as well as a notion of completeness for this class. See [40] for a rigorous definition of this class of optimization problems. In [6], the authors show the following result: a problem is MAX SNP hard if it is NP-hard and there exists a constant $\epsilon > 0$ such that approximating this problem within a factor of $1 - \epsilon$ is NP-hard. In the rest of the chapter, we will use this result to show the MAX SNP hardness of the different problems.

In the reductions presented in the following sections and in later chapters, we use the maximum cut problem. This problem is known to be MAX SNP hard, hence does not admit a PTAS unless P = NP [5, 6]. However, a dense instance of the problem admits a PTAS [4]. Let $1 - \Upsilon$ denote the upper bound on the polynomial time approximation factor for the maximum cut problem. Then $1 - \Upsilon$ can be no more than 0.94, unless P=NP [23]. The best known lower bound for $1 - \Upsilon$ is 0.878 [20].

The approaches to the ordered restriction map problem, as seen in the last chapter, can be broadly categorized as (1) using mutual agreement amongst the data in the population, and, (2) using an explicit map. We discuss the computational complexity of these approaches in the next two sections.

5.2 Using an explicit map (The EBFC Problem)

Recall that in this approach, in some sense we guess a map that best "fits" the the input data. In 4 this notion has been formalized to give rise to appropriate optimization problems. We discuss one such problem below (the BFC problem).

We formalize the problem as follows. Given m molecules with n sites each, and, p_j as the digestion rate for column j, obtain an alignment of the molecules such that the total number of 1's in the consensus cut columns, J, which is at least mp_J in each, is maximized (p_J is the digestion rate for site J). This is called the Binary Flip Cut (BFC) problem [18, 38]. We show that BFC is MAX SNP hard and give an upper bound on the polynomial time approximation factor of the problem. [3] showed that the problem is NP-hard but the cost function was different, we call this altered cost function $\text{BFC}_{\max}$, for uniformity of notation. In $\text{BFC}_{\max}$, the total number of consensus cut columns, K, which is at least mp_J in each consensus cut column, is maximized. We show at the end of this section that even $\text{BFC}_{\max}$ is MAX SNP-hard and give an upper bound on the polynomial time approximation factor of the problem.

We associate indicator variables X_i, $i = 1, 2, \ldots, m$, with every row which takes a value 1 if the molecule is flipped and 0 otherwise. Let Y_j, $j = 1, 2, \ldots, n$, be an indicator variable associated with every column that takes on a value of 1 if it is a consensus cut and 0 otherwise. Define *conjugate* of column j to be $\bar{j} = n - j + 1$. BFC can be modeled as the following optimization problem:

$$\max \left\{ \sum_{j=1}^{n} Y_j \left(\sum_{i=1}^{m} \left(M_{ij}(1 - X_i) + M_{i\bar{j}} X_i \right) - mp_j \right) \right\}. \tag{5.1}$$

Note that the term mp_j is used to ensure that the number of 1's along a consensus cut site j (with the rows flipped, if required) is at least mp_j. In other words, for a given alignment (which is an assignment of boolean values to X_i, $i = 1, 2, \ldots, m$, and Y_j, $j = 1, 2, \ldots, n$) we count the number of 1's in every column j, that has $Y_j = 1$, less mp_j.

Assume n is even[1]. Suppose, for every pair of columns, j and $\bar{j}$, we know

[1] If n is odd, we simply remove the middle site, that is, the site $(n+1)/2$, and the problem is unchanged.

whether both are consensus cut or neither are consensus cuts, then the remaining columns are such that exactly one of j and $\bar{j}$ is a consensus cut. This problem is called the exclusive BFC (EBFC) problem ([18, 38]). This problem was shown to be NP-hard in [14] using a similar reduction as in [3] to show the hardness of the $\text{BFC}_{\max}$ problem. In the practical algorithm presented in Section 7.2 information about the digestion rates is used to extract the EBFC problem. Later in this section we show that EBFC problem can also be viewed as a special case of the the BFC problem defined by cost function in (5.1).

Formally, the EBFC problem ([18, 38]) is as follows. Given m binary molecules of length n each, determine the flip for each molecule and an assignment of either j or $\bar{j}$ as a cut (but not both) for j, $1, \leq j \leq n/2$, such that the total number of 1's in the cut sites is maximized.

We prove the following lemma about the EBFC problem.

Lemma 3 *EBFC is a special case of the BFC problem.*

Proof: Let

$$S_j = |\{i | M_{ij} = 1 \text{ AND } M_{i\overline{j}} = 1\}|,$$
$$\bar{S}_j = |\{i | M_{ij} = 1 \text{ XOR } M_{i\overline{j}} = 1\}|,$$

where $\overline{j} = n - j + 1$. Further, let

$$p_j = p_{\overline{j}} = \frac{\bar{S}_j + 2S_j}{2m}. \tag{5.2}$$

Note that S_j is the count of the number of symmetric cuts and $\bar{S}_j$ is the total number of non-symmetric cuts in columns j and $\overline{j}$. Irrespective of the assignment of orientations to the molecules/rows, j and $\overline{j}$ always has at least S_j many 1's. The 1's corresponding to $\bar{S}_j$ get distributed between j and $\overline{j}$ depending on the alignment. We claim that under this definition of p_j the BFC problem is the same as the EBFC problem. It can be verified that under these conditions that $Y_j + Y_{\overline{j}} = 1$ holds for all j, since the definition of p_j ensures that only one of j or $\overline{j}$ is a consensus cut in the optimal alignment (and that is the one with the larger

number of 1's). If the number of 1's is equal in both, we can arbitrarily pick only one without changing the cost. QED

For the sake of completeness we give the following definitions.

Max Cut (MC) problem: Given a graph, find a partition of the vertices into disjoint sets, S_1 and S_2, that maximizes the number of edges with one vertex in S_1 and the other in S_2.

Bipartite Max Cut (BMC) problem: Given a bi-partite weighted graph with edge weights $\in \{+1, -1\}$, find a partition of the vertices into disjoint sets, S_1 and S_2, such that the sum of the weights of edges with one vertex in S_1 and the other in S_2 is maximized.

Theorem 1 *EBFC is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating EBFC within a factor of $1 - \epsilon$ is NP-hard.*

Proof: We reduce an instance of an MC problem to an instance of an EBFC problem: we show this reduction in two steps (steps 1 and 2) for the sake of clarity. Showing merely the relationship between the optimal solutions for the two problems would show that the EBFC problem is NP-hard; however, we also show how a solution to the Max Cut problem can be constructed given *any* solution (not necessarily optimal) to the EBFC problem. Further, we show that if the cost of the former is close to the optimal, so is the cost of the latter.

The proof proceeds in the following three steps. Let C^*_X denote the cost of an optimal solution and C_X denote the cost of an arbitrary solution to X.

Step 1. We show the reduction of an instance of the MC problem with e edges to an instance of the BMC problem with

(**1.1**) ono-to-one correspondence between the two solutions,

(**1.2**) $C^*_{MC} = C^*_{BMC}/2$, $C_{MC} \geq C_{BMC}/2$, and,

(**1.3**) the number of negative edges in the BMC is $2e$.

Step 2. We show the reduction of an instance of the BMC problem to an instance of the EBFC problem with

(**2.1**) ono-to-one correspondence between the two solutions, and,

(**2.2**) $C_{EBFC} - e^- = C_{BMC}$,

where e^- is the number of edges with negative weights in BMC.

Step 3. Finally, we show that the reduction is *gap-preserving*.

For some $\epsilon > 0$, let C^* denote the optimal solution and $\tilde{C}$ denote an approximate solution with $\tilde{C}_{EBFC} \geq (1-\epsilon)C^*_{EBFC}$.

$$\begin{array}{rcll} \tilde{C}_{MC} & \geq & \frac{\tilde{C}_{BMC}}{2} & \text{(using Step 1.2)} \\ & = & \frac{\tilde{C}_{EBFC}-2e}{2} & \text{(using steps 1.3 \& 2.2)} \\ & \geq & \frac{(1-\epsilon)C^*_{EBFC}-2e}{2} & \text{(by definition of } \tilde{C}_{EBFC}) \\ & = & \frac{(1-\epsilon)(C^*_{BMC}+2e)-2e}{2} & \text{(using Step 2.2)} \\ & = & \frac{(1-\epsilon)C^*_{BMC}-2e\epsilon}{2} & \\ & = & (1-\epsilon)\frac{C^*_{BMC}}{2} - e\epsilon & \\ & \geq & (1-\epsilon)C^*_{MC} - (\epsilon)2C^*_{MC} & \text{(since } C^*_{MC} \geq e/2) \\ & = & (1-3\epsilon)C^*_{MC} & \end{array} \tag{5.3}$$

This shows that given a PTAS for EBFC, we can construct a PTAS for MC, which is a contradiction, hence EBFC does not have a PTAS.

Now, we show each of the steps from 1 to 2. We outline the construction here and the details appear in Appendix A for the interested reader.

Step 1. MC to BMC reduction (see Figure 5.1).
Consider an MC problem with vertices and edges $(V, E), n = |V|, e = |E|$. Let a solution be of size K, and, the partition of the vertices induced by this solution be S_1 and S_2.

Reduction: Construct an instance of BMC with $(\tilde{V}, \tilde{E})$ as follows: For each $v_i \in V$, with degree d_i, construct $2(d_i + 1)$ vertices,

$$V_{gadget_i} = \{v'_{i0}, v'_{i1}, \ldots, v'_{id_i}, v''_{i0}, v''_{i1}, \ldots, v''_{id_i}\}.$$

Further, $wt(v'_{ij}, v''_{ij}) = wt(v'_{i0}, v''_{ij}) = wt(v'_{ij}, v''_{i0}) = -1, j = 1, 2, \ldots, d_i$. Thus, v_i gives rise to $3d_i$ edges with negative weight. Also if $v_1v_2 \in E$ then $wt(v'_{10}v''_{20}) = wt(v'_{20}v''_{10}) = +1$. It can be seen that this construction gives a bipartite graph with $\tilde{V} = V' \cup V''$ where $v'_x \in V', v''_x \in V''$.

Thus the BMC has $2n + 2e$ vertices, and, $2e$ edges with weights $+1$, and, $2e$ edges with negative weights. Recall for any graph $\sum_i d_i = 2e$.

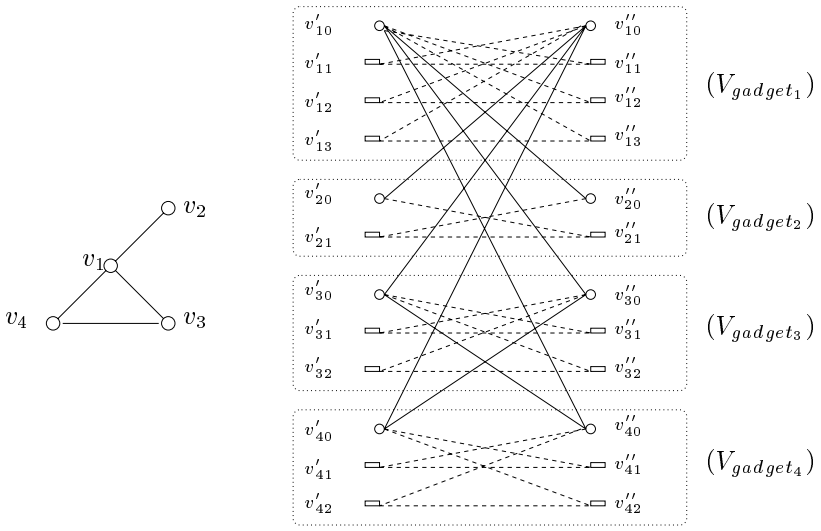


Figure 5.1: An example to show the reduction of an MC instance to a BMC instance. The bipartite graph is shown on the right: the solid edges have +1 weight and the dashed edges have weight -1. The "gadgets" corresponding to each vertex v_i, $i = 1 \ldots 4$ of the original graph, is shown enclosed in a dotted box.

	v''_{10}	v''_{11}	v''_{12}	v''_{13}	v''_{20}	v''_{21}	v''_{30}	v''_{31}	v''_{32}	v''_{40}	v''_{41}	v''_{42}	v''_{42}	v''_{41}	v''_{40}	v''_{32}	v''_{31}	v''_{30}	v''_{21}	v''_{20}	v''_{13}	v''_{12}	v''_{11}	v''_{10}	
v'_{10}	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	$\surd$
v'_{11}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	$\surd$
v'_{12}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	$\surd$
v'_{13}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	$\surd$
v'_{20}	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	$\times$
v'_{21}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	$\times$
v'_{30}	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	$\times$
v'_{31}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	$\times$
v'_{32}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	$\times$
v'_{40}	1	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	$\times$
v'_{41}	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	$\times$
v'_{42}	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	$\times$
	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\times$	$\times$	$\times$	$\times$	

Figure 5.2: The EBFC matrix corresponding to the BMC problem of Figure 5.1. Note that the 1's in the left half of the matrix correspond to the positive edge weights of the BMC and the 1's on the right correspond to the negative edge weights. [In the solution, the rows/molecules marked with $\surd$ are flipped, and, the columns marked with $\surd$ are the consensus cut columns.]

Step 2. BMC to EBFC reduction (see Figure 5.2).
Consider a BMC $((V_1, V_2), E)$, $V_1 = \{v_1^1, v_2^1, \ldots, v_m^1\}$, $V_2 = \{v_1^2, v_2^2, \ldots, v_n^2\}$, and, number of edges with negative weights be e^-. Let a solution be of size K and partition of vertices, $V_1 \cup V_2$, induced by this solution be S_1 and S_2.

Reduction: Construct an instance of EBFC $[M_{ij}]$ with m rows and $2n$ columns as follows. If $wt(v_i^1 v_j^2) = 1$, then $M_{ij} = 1, M_{i\bar{j}} = 0$. If $wt(v_i^1 v_j^2) = -1$, then $M_{ij} = 0, M_{i\bar{j}} = 1$. If $v_i^1 v_j^2$ is not an edge in the BMC, then $M_{ij} = M_{i\bar{j}} = 0$.

This concludes the proof of the inapproximability of the EBFC problem. QED

Corollary 2 *Achieving an approximation ratio* $1 - \Upsilon/3$ *for EBFC is NP-hard.*

Theorem 2 *BFC is NP-hard. Further, there exists a constant* $\epsilon > 0$ *such that approximating BFC within a factor of* $1-\epsilon$ *is NP-hard. Also achieving an approximation ratio* $1 - \Upsilon/3$ *for BFC is NP-hard.*

We also show the following results for the $\text{BFC}_{\max}$ problem.

Theorem 3 *$BFC_{\max}$ is NP-hard. Further, there exists a constant* $\epsilon > 0$ *such that approximating $BFC_{\max}$ within a factor of* $1 - \epsilon$ *is NP-hard. Further, achieving an approximation ratio* $(1 - \Upsilon/3)\frac{p_{\max}}{p_{\min}}$ *for $BFC_{\max}$ is NP-hard.*

Proof. Under the definition of p_j's as in equation (5.2) the $\text{BFC}_{\max}$ is the same as the EBFC problem (the number of consensus cuts is always $n/2$, when the molecules have n sites), hence $\text{BFC}_{\max}$ is NP-hard.

Next, we show that if we have a PTAS for $\text{BFC}_{\max}$, we have a PTAS for BFC, which would be a contradiction. Given a BFC let $p_{\min} = \min_j p_j$, and $p_{\max} = \max_j p_j$. Let $\tilde{X}$ denote an approximate solution and X^* denote the optimal solution. Recall that $\text{BFC}_{\max}$ optimizes the number of consensus columns. Let N^* denote the number of consensus cut columns when the solution is optimal, with C^* as the BFC cost, and let $\tilde{N}$ denote the number of consensus columns in a solution that is not necessarily optimal, with $\tilde{C}$ as the corresponding BFC cost. Thus if $\text{BFC}_{\max}$ has a PTAS let $\frac{\tilde{N}}{N^*} \geq \epsilon$ for some $0 < \epsilon \leq 1$. Note that N^* is the number of consensus cuts. Since $\tilde{C} \geq \tilde{N} p_{\min}$ and $C^* \leq N^* p_{\max}$, we have the following:

$$\frac{\tilde{C}}{C^*} \geq \frac{\tilde{N} p_{\min}}{N^* p_{\max}} \geq \epsilon \frac{p_{\min}}{p_{\max}}. \tag{5.4}$$

QED

Corollary 3 *There does not exist a polynomial time algorithm, unless $P = NP$, that guarantees the estimation of $(1 - \Upsilon/3)$ of the total number of consensus cuts when digestion rates at each site is the same under the $BFC_{\max}$ model.*

5.3 Using mutual agreement of data (The CG, WCG Problems)

We show the hardness of the CG and WCG problems described in Section 4.2.1, below.

Theorem 4 *The CG problem is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating CG within a factor of $1 + \epsilon$ is NP-hard.*

Proof: We give the proof in two steps. In step 1 we show a reduction of an instance of a maximum cut (MC) problem to an instance of the CG problem and show that $C_{MC} = 2C_{CG} - e$ where C_X is a solution to the problem X and e is the number of edges in the MC problem. In step 2 we show that the reduction is *gap-preserving*.

Step 1. Given an instance of the MC problem with n vertices and e edges, we construct an instance of CG by simply labeling every edge as *Opposite*. A consistent graph is such that the vertices can be partitioned into two sets S_1 and S_2 such that $\forall v_i, v_j \in S_1$ (or S_2) $, L(e_{ij}) = Same$, and, $\forall v_i \in S_1, v_j \in S_2, L(e_{ij}) = Opposite$. There are only two kinds of consistent triangles: (1) all labels are *Same* or (2) exactly one label is *Same*. It can be verified that only these two kinds of triangles (and no other) exist for the consistent graph whose vertices are given by $S_1 \cup S_2$.

Given a solution of size e' to the CG, which is the number of edges with label *Opposite* (since there was no edge with label *Same*), we can show that the solution to the MC problem is of size $2e' - e$. It can also be verified that increasing the solution to the CG problem by $x > 0$, increases the solution to the MC problem by x.

Step 2. Let $\tilde{C}_{CG}$ denote an approximate solution and C^*_{CG} denote the optimal solution. Then $\tilde{C}_{CG} \leq (1+\epsilon)C^*_{CG}$.

$$\begin{aligned}
\tilde{C}_{MC} &= 2\tilde{C}_{CG} - e && \text{(using step 1)} \\
&\geq 2(1-\epsilon)C^*_{CG} - e && \text{(by defn of } \tilde{C}_{MC}) \\
&\geq (1-\epsilon)(C^*_{MC} + e) - e && \text{(by step 1)} \\
&= (1-\epsilon)C^*_{MC} - e\epsilon \\
&\geq (1-3\epsilon)C^*_{MC} && \text{(since } C^*_{MC} \geq e/2).
\end{aligned} \tag{5.5}$$

This shows that given a polynomial time approximation scheme (PTAS) for the CG problem, we can construct a PTAS for the MC problem (using step 1 gives the correspondence between the two solutions), which is a contradiction; hence the CG problem does not have a PTAS. This concludes the proof. QED

Corollary 4 *Achieving an approximation ratio* $1 - \Upsilon/3$ *for the CG problem is NP-hard.*

This directly follows the theorem and the fact that the approximation factor can be no more than $1 - \Upsilon$, unless P=NP [23].

Corollary 5 *The WCG and dM problems are MAX SNP hard. Further, achieving an approximation ratio* $1 - \Upsilon/3$ *for the WCG and dM problems is NP-hard.*

The same proof goes through for the WCG problem as well, by simply assigning weights along with the labels to the edges. Since WCG is a special case of the d-wise match problem (with $d = 2$), all the results for the WCG problem also hold for the dM problem.

All the acronyms used in this chapter also have been listed in Appendix C.

Chapter 6

Theoretical Algorithms

In this chapter, we give theoretical algorithms for the idealized version of both the approaches described in the previous chapter. (The practical algorithms for these problems are discussed in Chapter 7.)

Roadmap. In Section 6.1 we discuss the Exclusive Binary Flip Cut (EBFC) problem, defined earlier: in Section 6.1.1 we describe the basic constructions that are later used in Section 6.1.2 to give a guaranteed polynomial time 0.878 approximation algorithm and in Section 6.1.3 to give a polynomial time approximation scheme (PTAS) for the EBFC problem. In Section 6.2 we give a guaranteed polynomial time 0.817-approximation algorithm for the Consistency Graph (CG) and the Weighted Consistency Graph (WCG) problems. The polynomial time approximate algorithms are based on the Goemans and Williamson algorithm for the Maximum Cut (MC) problem [20] and use semidefinite programming to achieve these approximation bounds.

6.1 The EBFC Problem

We first give a reduction of an instance of the EBFC problem to an instance of the MC problem. This reduction is best described in two steps: first reducing an instance of the EBFC problem to an instance of the BMC problem and then

reducing this instance of the BMC problem to that of an MC problem. Using this reduction we give a PTAS for a dense instance of the EBFC problem and a polynomial time 0.878 approximation algorithm for a general instance of the EBFC problem.

6.1.1 Basic constructions

EBFC to BMC reduction. Given an $m \times 2n$ binary matrix $[a_{ij}]$ for the EBFC problem, we first pre-process the matrix as follows: if $a_{ij} = a_{i\overline{j}} = 1$, then we make the assignment $a_{ij} = a_{i\overline{j}} = 0$. This does not affect the algorithm since one of j or $\overline{j}$ is a cut and in all the configurations there will be contribution of 1 towards the solution due to these two values. Notice that this can only *improve* the approximation factor of the solution.

After the pre-processing, we generate a bipartite graph with $m + n$ vertices with the first partition having m vertices and the second n vertices. The weights are assigned as follows: if $a_{ij} = 1$, then there is an edge between vertex v_i of the first partition and vertex v_j, in the second partition with a weight of 1; if $a_{i\overline{j}} = 1$, then there is an edge between vertex v_i of the first partition and vertex v_j, in the second partition with a weight of -1; if $a_{ij} = a_{i\overline{j}} = 0$, then there no edge between the vertices v_i of partition 1 and v_j of partition 2.

There exists a correspondence between a solution to the BMC problem and a solution to the EBFC problem with the corresponding costs defined as $\tilde{C}_{BMC}$ and $\tilde{C}_{EBFC}$ respectively. If the solution to the BMC problem is optimal, its cost is denoted by C^*_{BMC} and the corresponding solution to the EBFC problem is also optimal with the cost denoted as C^*_{EBFC}. Then, the following hold:

$$C^*_{EBFC} = C^*_{BMC} + L, \quad \tilde{C}_{EBFC} \geq \tilde{C}_{BMC} + L. \tag{6.1}$$

L is the number of 1's in the right half of the input matrix ($L = \sum_{i=1}^{m} \sum_{j=n+1}^{2n} a_{ij}$). Notice that all the rows of the matrix can be flipped to interchange the left and the right half of the matrix.

BMC to MC reduction. Consider an instance of BMC $(V, E^+ \cup E^-)$ where the edges with weight 1 are in E^+ and the ones with negative weight are in E^-. Further, without loss of generality, the number of vertices on the left partition of the BMC problem is m and on the right partition is n.

Construct an instance of the MC problem with $|V| + m'$ vertices and $|E^+| + m' + |E^-|$ edges as follows: if a vertex v has an edge of weight -1 incident on it, then replace it by a pair of vertices u, w connected by an edge $[u, w]$; all thed edges of weight $+1$ incident on v are now incident on u. All the edges of weight -1 incident on v are now incident on w with the weight changed to $+1$. See Figure 6.1 for an illustration.

Let the solution to the MC problem include l_1 edges which correspond to the original edges with weight -1, m_1 edges that correspond to the new single edges introduced and and p of the original edges (which had a weight of 1 in the BMC problem). Then

$$\tilde{C}_{MC} = p + m_1 + l_2, \tag{6.2}$$

and the cost of the BMC problem by the construction is,

$$\tilde{C}_{BMC} = p - l_1. \tag{6.3}$$

Let $L = |E^-|$. Since $l_1 + l_2 = L$, we have from equations (6.2) and (6.3)

$$m_1 + L \leq C^*_{BMC}. \tag{6.4}$$

$m_1 + L$ is the trivial solution obtained by having all the new vertices (with edges having weight -1 incident on them) in one partition and the rest of the vertices in the other partition.

Assuming we can obtain a solution for the MC problem with cost $\tilde{C}_{MC} \geq \epsilon C^*_{MC}$,

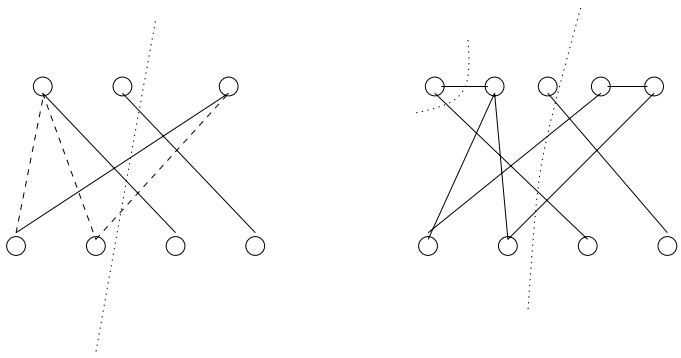


Figure 6.1: The instance of the BMC problem shown on the left where an edge with negative weight is shown as a dashed line. The optimal partition of the vertices is shown in the dotted curve. The instance of the MC problem is shown on the right that is constructed from this BMC. Every vertex on the "top" partition of the BMC instance, that has at least one edge with weight -1 associated with it, has a corresponding new vertex with an edge between the original and the new vertex. All the edges with weight $+1$ are incident on the first and all the edges with weight -1 are incident on the second vertex as shown. The corresponding optimal solution for the MC problem is shown by the dotted curve.

---

for some $0 < \epsilon \leq 1$, we obtain the following.

$$\begin{aligned}
\tilde{C}_{BMC} &= \tilde{C}_{MC} - (m_1 + L) && \text{(from eqn (6.4))} \\
&\geq \epsilon C^*_{MC} - (m_1 + L) \\
&\geq \epsilon(C^*_{BMC} + (m_1 + L)) - (m_1 + L) && \text{(from eqn (6.4))} \\
&= \epsilon C^*_{BMC} - (1 - \epsilon)(m_1 + L) \\
&\geq \epsilon C^*_{BMC} - (1 - \epsilon)(C^*_{BMC}) && \text{(from eqn (6.4))} \\
&\geq (2\epsilon - 1)C^*_{BMC}.
\end{aligned} \tag{6.5}$$

6.1.2 A 0.878 approximation algorithm

In this section, using the constructions presented in the last section, we present an algorithm that achieves an approximation factor of 0.878 for the EBFC problem.

Let $[a_{ij}]$ be the matrix for the EBFC problem and L be the total number of 1's in the right half of this matrix, then the following holds:

$$2L \geq \sum_i \sum_j a_{ij} \geq C^*_{EBFC}, \tag{6.6}$$

Recall that all the rows (molecules) can be flipped without altering the problem so that the above holds. Further, let $\tilde{C}_{BMC} \geq \delta C^*_{BMC}$.

$$\begin{aligned} \tilde{C}_{EBFC} &= \tilde{C}_{BMC} + L && \text{(using equation (6.1))} \\ &\geq \delta C^*_{BMC} + L \\ &\geq \delta(C^*_{EBFC} - L) + L && \text{(using equation (6.1))} \\ &= \delta C^*_{EBFC} + (1-\delta)L \\ &\geq \tfrac{(\delta+1)}{2} C^*_{EBFC} && \text{(using equation (6.6))} \end{aligned} \tag{6.7}$$

When $\delta = 2\epsilon - 1$, from derivation (6.5), then we obtain $\tilde{C}_{EBFC} \geq \epsilon C^*_{EBFC}$, if we have $\tilde{C}_{MC} \geq \epsilon C^*_{MC}$. Using the algorithm presented in [20] for the MC problem, we obtain a 0.878 approximation algorithm for the EBFC problem.

6.1.3 A PTAS for a dense instance of EBFC

In [38], the following result was shown: a dense version of the EBFC problem has a polynomial time approximation scheme (PTAS). The proof used techniques from randomized rounding and integer linear programming along the lines of one of the results in [4]. An instance of the EBFC problem is dense if the number of 1's in a column and its conjugate is at least some fraction f, $0 < f \leq 1$, of the number of molecules.

Using the basic reductions described in the earlier section, we can show that a dense EBFC admits a PTAS: this vastly simplifies the earlier proof in [38] in the sense that we merely argue that the reduction from a dense EBFC results in a dense instance of the MC problem and the result from [4] directly applies here.

It is straightforward to see that if the number of 1's in the dense EBFC problem is at least mnf where m is the number of molecules (rows), n the number of sites (columns) and f is the fixed fraction (away from zero), then the instance of the

MC problem has $m+n+m_1$ vertices with at least $mnf+m_1$ edges where $m_1 \leq m$. Thus the instance of the MC problem is dense and from [4] we have the result that dense instances of MC problems admit a PTAS.

6.2 The CG and WCG Problems

Recall that Consistency Graph (CG) and Weighted Consistency Graph (WCG) problems arise in the approach that uses mutual agreement of data: this takes into account the false positive and false negative errors along with orientation uncertainties.

6.2.1 A 1.183 approximation algorithm

Let the maximum cut problem on a graph with weights $+1$ be called the Positive Max Cut (PMC) problem and the one with weights $+1$ or -1 be called Negative Max Cut (NMC). We show that given an arbitrary instance of the CG problem, we can construct an instance of the NMC problem, and then construct an instance of the PMC problem. As all the weights in the PMC problem are positive, we can use the Goemans and Williamsons' semi-definite programming based algorithm to get a 0.878-approximation of the PMC problem. Next we use this solution to obtain an approximate solution for the NMC problem, and using that solution we obtain a solution for the CG problem.

We describe this in three steps. In Step 1, we describe the reduction of an instance of the CG problem to an instance of an NMC problem and also describe the correspondence between a solution to the NMC problem and that of the CG problem. In Step 2, we describe the reduction of an instance of an NMC problem to an instance of the PMC problem and describe the correspondence between a solution to the PMC problem and that of the NMC problem. These two steps give the algorithm and finally in Step 3, we argue that the algorithm gives an approximation factor of 0.817.

Step 1. Given an instance of the CG problem given by graph $\mathcal{G}_{CG}$, with m vertices and n edges labeled as *Same* or *Opposite*, we construct an instance of

NMC on the graph $\mathcal{G}_{NMC}$, by assigning a weight of -1 to all those edges labeled *Same* and assigning a weight of $+1$ to the *Opposite* labeled edges. Thus the number of vertices of $\mathcal{G}_{NMC}$ is m and the number of edges is n.

Next, we claim that an optimal solution to the NMC instance gives an optimal solution to the CG problem, and an approximate solution to the CG problem can be constructed from the approximate solution to the NMC instance.

Let L be the total number of edges labeled *Same* in the CG problem or labeled -1 in the NMC problem. Given a solution of the form $p_1 - l_2$ of the NMC instance, where p_1 is the number of edges with weight $+1$ and l_1 is the number of edges with weight -1 in the cut, the solution C_{CG} to the CG problem is

$$C_{CG} = p_1 + l_1, \tag{6.8}$$

where l_1 is the number of edges with weight -1 not in the cut. The p_1 edges corresponding to the $+1$ labels are the ones in the CG instance which are labeled *Opposite*, and, the l_2 edges corresponding to the -1 labels are the ones in the CG instance which are labeled *Same*, and these edges *do not switch labels* in the solution. Thus $L = l_1 + l_2$. Also, it can be verified that an optimal solution in the NMC instance gives an optimal solution in the CG problem.

Step 2. Given an instance of the NMC problem, we construct an instance of the PMC problem (with weight on the edges as $+1$) by replacing every edge with a negative weight by two edges and a vertex, each edge having a weight of 1. If L is the number of edges with weight -1, then the PMC instance has $m + n/2 + L$ vertices.

Next, we claim that that an optimal solution to the PMC instance, gives an optimal solution to the NMC problem, and, an approximate solution to the NMC problem can be constructed from the approximate solution to the PMC instance.

Now, we give the correspondence between the solutions in each of the problem. Notice that the edges introduced in the reduction come in pairs. Let the solution to the PMC problem include l_1 edges which are *not* paired, $2l_2$ paired edges and p of the original edges (which had a weight of 1 in the NMC problem). Then

$$C_{PMC} = p + 2l_1 + l_1. \tag{6.9}$$

The edges that come in pairs correspond to the l_2 edges which are not in the cut in the NMC instance and the edges corresponding to l_1 which are not in pairs correspond to the edges in the cut of the NMC instance. Thus the cost of the NMC problem by this construction is $C_{NMC} = p_1 - l_1$. It can be verified that the optimal solution in one corresponds to the optimal one in the other.

Step 3. Thus from equations (6.8) and (6.9), we have

$$C_{PMC} = L + C_{CG}. \tag{6.10}$$

Now, we make the following observation:

$$2L \leq p_1 + l_1. \tag{6.11}$$

This holds since $2L$ is the cost of the trivial solution to the constructed PMC problem which corresponds to the zero solution in the NMC instance, (where all the vertices belong to just one partition!), hence this must be smaller than the cost of any other non-trivial solution (*viz.*, $p_1 + l_2$). Also, if C^*_{CG} is the optimal solution then,

$$p_1 + l_1 \leq C^*_{CG}. \tag{6.12}$$

Finally, we use the algorithm presented in [20] to obtain an algorithm for the PMC problem which has an approximation factor of 0.878. Note that we could not directly use it on the NMC instance due to the -1 weights. Let $\tilde{C}_X$ denote an approximate solution and C^*_X denote the optimal solution to problem X.

$$\begin{aligned}
\tilde{C}_{CG} &= \tilde{C}_{PMC} - L && \text{(from eqn (6.10))} \\
&\geq 0.878 C^*_{PMC} - L && \text{(from [20])} \\
&\geq 0.878(C^*_{CG} + L) - L && \text{(from eqn (6.10))} \\
&= 0.878 C^*_{CG} - 0.122L \\
&\geq 0.878 C^*_{CG} - 0.122(C^*_{CG}/2) && \text{(from eqns (6.11) and (6.12))} \\
&\geq 0.817 C^*_{CG}.
\end{aligned}$$

This concludes the argument.

0.817-approximation algorithm for the WCG problem. Assigning the appropriate weights to the graph, *i.e.*, positive weights to edges labeled *Opposite* and negative

weights to edges labeled *Same*, and, using the same steps as in the CG problem, we get similar results for the WCG problem. Also note that in the NMC to PMC reduction if an edge has weight $-w$ it is replaced by two edges with weight w each incident on a vertex as for the CG problem.

Chapter 7

Practical Algorithms

The theoretical algorithms of the last section use semidefinite programming and are high polynomial time algorithms. In this section we present (low polynomial) practical algorithms that have been tested on simulated as well as real data. For the real data, we evaluate our results by comparing them with the maps provided by the laboratory.

Modeling the false positive, false negative errors along with orientation uncertainties, we tested both the approaches of using mutual agreement and using an explicit map on simulated and real data. For the former, we start out with an initial hypothesis for a map, which is defined by one or more of the data molecules randomly chosen. This initial hypothesis is refined by comparing with different molecules. Our experience with this approach has been that (1) different starting hypotheses alter the accuracy of the final results and (2) the threshold that bounds the distance between the same cut in different molecules also alters the final result. Further, our experience has been that the second approach of using an explicit map works better (See Table 7.1 for a comparison of this approach with the others). Hence we present the details of the second approach in the following sections.

Various approaches have been suggested to deal with the ordered restriction map problem [3, 19, 29, 32, 38, 49]. In [3, 32] statistical methods to solve the problem has been presented: in [3] a MAP (Maximum A Posteriori) method has been presented along with heuristics for branch and bound techniques, and, in [32]

a hierarchical Bayes model has been used. (The MAP model has been analyzed in Section 4.3.) In [19] a simulated annealing technique and in [29] various combinatorial algorithms, along with a probabilistic analysis of the algorithms, have been presented.

Roadmap. In Section 7.1 we develop heuristics to solve the EBFC problem (the problem is shown to be MAX SNP hard in Section 5.2). We design the algorithm with two sub-tasks: (1) obtaining an order of the columns and (2) given the order, assigning orientations to the molecules. We experimented with three different heuristics for the first sub-task, the column order heuristic. However, the real data also has positional variations of the cut sites in the molecules (also called sizing error). Section 7.2 deals with handling this error and then extracting an EBFC problem from this using appropriate heuristics. In Section 7.3, we introduce two measures of evaluating the result (when the true map is not known) and compare the different column order heuristics on the real data using the two measures. Our experience has been that the sorted column-order heuristic, which results in a linear time ($2mn$) algorithm, works well and we present the results on real data from the laboratory using this heuristic. Since the "true" maps for this data were available we compare the computed maps with these maps and observe less than 3% error.

7.1 Solving the EBFC problem

In this section, we describe an extremely simple approach for the EBFC problem. The EBFC problem is obtained from the problem after appropriate pruning as discussed in Section 7.2. The approximation factor for this algorithm is 0.5. However as the experimental results show, this algorithm is remarkably accurate in predicting the consensus cuts on both synthetic data and real data [1].

We deal with the sites j, $1 \leq j \leq n/2$, in some sequential order, one at a time. At every stage, given j, we have to decide whether j or $\bar{j}$ can be the designated

[1] A preliminary version of this also appeared in [38].

cut. We use indicator variables X_i associated with each row (molecule) and Y_j associated with every column (cut), $1 \leq i \leq m, 1 \leq j \leq n/2$. We adhere to the following convention in this section:

$$X_i = \begin{cases} 1 & \text{row } i \text{ is flipped,} \\ 0 & \text{otherwise.} \end{cases} \qquad Y_j = \begin{cases} 1 & \text{site } j \text{ is a consensus cut,} \\ 0 & \text{otherwise.} \end{cases}$$

The algorithm is summarized as follows.

Step 1. Determine an ordering of the sites.

Step 2. For a given ordering, obtain the values of X_i and Y_j.

Ordering the sites (Step 1). In this section we discuss some of the different ways of ordering the sites. For the EBFC problem, the potential or score S_j, of every site is defined as follows:

$$S_j = |\{j | M_{ij} = 1 \, OR \;\; M_{i\bar{j}} = 1\}|, 1 \leq j \leq n/2.$$

Thus, if the only site in the data is j, then the total number of 1's in a consensus cut site is exactly S_j; when other sites are present, this is bounded above by S_j.

We experimented extensively with three different schemes that we describe below.

1. Random Order: This uses a random order of the sites.
2. Sorted Order: This uses the sites in decreasing order of the S_j scores.
3. MaST Order: This uses a Maximum Spanning Tree on a graph constructed from the data.

   This scheme is a little more sophisticated, and, involves three major steps: first, a pruning of the input matrix M, second, construction of a labeled and weighted graph, $\mathcal{G}$, and third, extraction of a maximum spanning tree (MaST) of $\mathcal{G}$. The labels on the edges of the MaST define the resulting map as shown in lemma 4. We give the details of each step below.

   (a) Pruning: Given an EBFC problem with n cuts and m molecules, the input can be pruned so that the following holds. First, every molecule or

row must have at least two 1's, otherwise that molecule can be removed from consideration without affecting the problem. This implies that we drop all those molecules which have exactly one cut, since this does not affect the optimal configuration. Second, if a 1 appears in a site and its conjugate, we just drop the cut from that molecule, that is replace both the 1's by 0, since no matter what flip is assigned to the molecule, it will contribute 1 to the optimization function.

(b) Graph Construction: Given a pruned EBFC, we define mutual information of the i^{th} molecule, regarding two cuts j, k as whether both the cuts appear in the same half (sites from $1, 2, \ldots, n/2$ or sites from $n/2 + 1, n/2 + 2, \ldots, n$) of the molecule or on different halves of the molecule.

$$I^i_{jk} = \begin{cases} 1 & 1\text{'s are on same half of molecule } i, \\ -1 & 1\text{'s are on different halves of molecule } i. \end{cases} \tag{7.1}$$

For a pruned EBFC problem with n cuts and m molecules, define a weighted-labeled graph $\mathcal{G}(V, E)$ as follows:

- Vertices: Every $v \in V$ corresponds to a cut, thus $n = |V|$.
- Edges with weights and labels: Let U_j be the set of molecules that have a cut at j or $\bar{j}$ and $U_{jk} = U_j \cap U_k$, and, whenever $|U_{jk}| > 0$, let $U_{jk} = G_{jk} \cup L_{jk}$ such that $G_{jk} = \left\{i | I^i_{jk} = c\right\}$, $L_{jk} = \left\{i | I^i_{jk} = -c\right\}$, and further, $|G_{jk}| \geq |L_{jk}|$. If $|U_{jk}| > 0$, define an edge between vertices v_j and v_k with weight as G_{jk}. and label $l_{jk} = c$. When the label $l_{jk} = 1$, it indicates that the cut k is on the same half of the map as is j, and, when $l_{jk} = -1$, it indicates that the cut k is on the opposite half from the cut j.

  Note that the pruning of the first step ensures that these sets are well-defined. A molecule belongs to U_{jk}, if it has both the cuts j and k. Further, we partition U_{jk} into sets such that in each partition, every pair of cuts j and k has the same mutual information value. Since there are only two such values, the number of partitions is 2. The set with the higher cardinality is termed G_{jk} and the one with

lower cardinality is L_{jk}.

(c) MaST Computation: Construct the MaST using any standard algorithm.

Lemma 4 *A spanning tree T of $\mathcal{G}$ defines a unique map.*

Proof: Recall that Y_j is an indicator variable for a site: if $Y_j = 1$, then j is a cut and is 0 otherwise. Define a *map* to be a set of assignments (0 or 1) to Y_j, $j = 1, 2, \ldots, n$. For the sake of convenience, let us define

$$\tilde{Y}_j = \begin{cases} 1 & \text{when } Y_j = 1, \\ -1 & \text{when } Y_j = 0. \end{cases}$$

It is easy to see that if l_{jk} is a label on the edge $v_j v_k$, then an assignment such that $\tilde{Y}_j \tilde{Y}_k = l_{jk}$, $\forall j, k$, gives a map. Such an assignment to $\tilde{Y}_j$, $\forall j$, is possible and we give a construction: We carry out a BFS traversal of the tree, and, as each node, v_j, is traversed, we give an appropriate assignment to the indicator variable $\tilde{Y}_j$. Without loss of generality, let the root of the tree be v_0, say. Let $\tilde{Y}_0 = 1$. Consider edge $v_j v_k$ of the tree where node v_j has been traversed first, and thus $\tilde{Y}_j$ already has an assignment. Then $\tilde{Y}_k = l_{jk} \tilde{Y}_j$, which gives a consistent assignment. QED

It is easy to see that this works in $O(mn)$ space and takes $O(nm + n^2 \log n)$ time.

Figure 7.1 shows a simple example of constructing a graph from a binary matrix and obtaining the consensus map from the MaST of the graph.

Figures 7.3 to 7.5 show the results of using different site orderings on a sample set of data.

Solving EBFC, given an ordering of sites (Step 2). Our algorithm is greedy in two ways which are mutually orthogonal. We try to attain as many 1's as possible in candidate cut sites by greedily flipping the molecules appropriately to the extent we can. However, as we accumulate cut sites, existing molecule orders

	A	B	C	$\overline{C}$	$\overline{B}$	$\overline{A}$
1	1	0	0	1	1	0
2	1	0	0	0	1	0
3	1	0	0	0	1	0
4	1	1	1	0	0	0
5	1	1	1	0	0	0
6	0	1	1	0	0	0
7	0	1	1	0	0	0

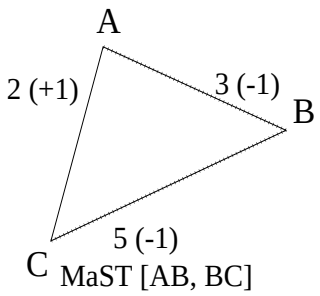


Figure 7.1: The EBFC binary matrix and the corresponding graph $\mathcal{G}$. The sets of molecules (denoted by the row number) are as follows: $G_{AB} = \{1, 2, 3\}$, $L_{AB} = \{4, 5\}$, $G_{AC} = \{4, 5\}$, $L_{AC} = \{1\}$, $G_{BC} = \{1, 4, 5, 6, 7\}$, $L_{BC} = \{\}$. Thus $wt_{AB} = |G_{AB}| = 3$, $wt_{AC} = |G_{AC}| = 2$, $wt_{BC} = |G_{BC}| = 5$. The labels are $l_{AB} = -1$, $l_{AC} = +1$ and $l_{BC} = +1$. The Maximum Spanning Tree is given by the edges AB and BC. Hence the consensus map is [101|010] (or [010|101]).

Input EBFC matrix

	A	B	C	D	E	F	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$
1	1	0	0	0	0	0	0	0	1	1	1	0
2	1	0	0	0	0	0	1	1	0	0	1	0
3	1	0	1	0	0	0	0	0	0	0	0	0
4	1	0	0	1	0	0	0	0	0	0	0	0
5	1	0	0	0	1	0	0	0	0	0	0	0
6	1	0	0	0	0	1	0	0	0	0	0	0

True Optimal

	1	1	1	1	1	1						
	A	B	C	D	E	F	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$
1	0	1	1	1	0	0	0	0	0	0	0	**1**
2	0	1	0	0	1	1	0	0	0	0	0	**1**
3	1	0	1	0	0	0	0	0	0	0	0	0
4	1	0	0	1	0	0	0	0	0	0	0	0
5	1	0	0	0	1	0	0	0	0	0	0	0
6	1	0	0	0	0	1	0	0	0	0	0	0

Figure 7.2: A small example of the input matrix with the true optimal configuration shown on the right. The results due to the different orders are shown in Figures 7.3 to 7.5.

Random order:

Step 1 (site C)												Steps 2-6 (sites D, E, F, B, A)												
.	.	0	.	.	.							1	0	0	0	1	1							
A	B	C	D	E	F	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	A	B	C	D	E	F	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	
.	.	0	.	.	.	.	.	.	1	.	.	1	0	0	0	0	0	0	0	1	1	1	0	
.	.	0	.	.	.	.	.	.	0	.	.	1	0	0	0	0	0	**1**	**1**	0	0	1	0	
.	.	0	.	.	.	.	.	.	1	.	.	0	0	0	0	0	0	0	0	0	1	0	**1**	*
.	.	0	.	.	.	.	.	.	0	.	.	0	0	0	0	0	0	0	0	1	0	0	**1**	*
.	.	0	.	.	.	.	.	.	0	.	.	1	0	0	0	1	0	0	0	0	0	0	0	
.	.	0	.	.	.	.	.	.	0	.	.	1	0	0	0	0	1	0	0	0	0	0	0	

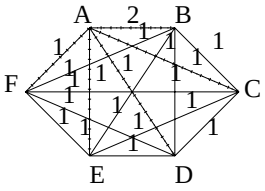
Figure 7.3: Figure 7.2 shows the input matrix and the true optimal. Here random order of sites is used which gives a cost of 12. (The flipped rows are marked by asterisk and the 1's in bold denote the cuts that do *not* contribute to the cost.)

Sorted order:

Step 1 (site A)												Steps 2-6 (sites B, C, D, E, F)												
1	.	.	.	.	.							1	0	1	1	1	1							
A	B	C	D	E	F	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	A	B	C	D	E	F	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$	
1	.	.	.	.	.	.	.	.	.	.	0	1	0	0	0	0	0	0	0	**1**	**1**	1	0	
1	.	.	.	.	.	.	.	.	.	.	0	1	0	0	0	0	0	**1**	**1**	0	0	1	0	
1	.	.	.	.	.	.	.	.	.	.	0	1	0	1	0	0	0	0	0	0	0	0	0	
1	.	.	.	.	.	.	.	.	.	.	0	1	0	0	1	0	0	0	0	0	0	0	0	
1	.	.	.	.	.	.	.	.	.	.	0	1	0	0	0	1	0	0	0	0	0	0	0	
1	.	.	.	.	.	.	.	.	.	.	0	1	0	0	0	0	1	0	0	0	0	0	0	

Figure 7.4: Figure 7.2 shows the input matrix and the true optimal. Here sorted order of sites is used which gives a cost of 12. (The flipped rows are marked by asterisk and the 1's in bold denote the cuts that do *not* contribute to the cost.)

MaST order:



MaST - (AB, AC, AD, AE, AF)

MaST labels - (-1,-1,-1,-1,-1)

Map [100000:111110]

1	0	0	0	0	0						
A	B	C	D	E	F	$\overline{F}$	$\overline{E}$	$\overline{D}$	$\overline{C}$	$\overline{B}$	$\overline{A}$
1	0	0	0	0	0	0	0	1	1	1	0
1	0	0	0	0	0	1	1	0	0	1	0
1	0	**1**	0	0	0	0	0	0	0	0	0
1	0	0	**1**	0	0	0	0	0	0	0	0
1	0	0	0	**1**	0	0	0	0	0	0	0
1	0	0	0	0	**1**	0	0	0	0	0	0

Figure 7.5: Figure 7.2 shows the input matrix and the true optimal. Here MaST order of sites is used which gives a cost of 12. Notice that the different orders (see also Figures 7.3 and 7.4 to give rise to very different consensus maps, although for this example the cost in the three cases are the same (cost = 12). The true optimal has been obtained by inspection. (The flipped rows are marked by asterisk and the 1's in bold denote the cuts that do *not* contribute to the cost.)

hinder procuring additional potential 1's. In that case, we reverse the flips of the molecules selectively before proceeding further. This may be thought of as limited backtracking on the choice of molecule flips. Similarly, we do a limited backtracking on the choice of the consensus cuts seen so far. In this phase, we ensure that the number of 1's in site j, where $Y_j = 1$, is more than the number of 1's in $\bar{j}$ and, when $Y_j = 0$, the number of 1's in the site $\bar{j}$ is more than that in j. This is carried out in the following steps.

1. Backtracking molecule flips.

   Given site j, compute the two costs, Δ_j, the cost of assigning j as the consensus cut, and $\Delta_{\bar{j}}$, the cost of assigning $\bar{j}$ as the consensus cut. To compute this in an efficient manner, we use local variables, d_i, for every molecule i to store the current status. d_i keeps track of the difference between the number of number of 1's in cut sites and the number of 1's in the non-cut sites, per molecule. If $d_i < 0$, we count that as $-d_i$ (since molecule i can be flipped to get $d_i \geq 0$). This can be considered as limited backtracking on the flip assignments to the molecules. Summing over all the molecules, $\sum_i d_i$, gives the total cost.

   We compare these sums for the two cases: the consensus cut being j and then $\bar{j}$. We choose the one that gives a higher cost. We also go ahead and flip the molecules so that all the molecules are such that $d_i > 0$.

2. Backtracking consensus cut assignments (roll-back).

   We also carry out a limited backtracking on the consensus cut assignments to the columns as follows. For every site j, $j = 1, 2, \ldots, n/2$, define D_j as the difference between the number of 1's in the consensus cut site and the number of 1's in its conjugate. Notice that, at this stage, $\sum_{i=1}^{m} d_i = \sum_{j=1}^{n/2} D_j$. If $D_j < 0$ for some j, then switch the assignments of Y_j and $Y_{\bar{j}}$. Next, update d_i, $\forall i$. Repeat this phase until $D_j \geq 0$, $\forall j$. We note that this process terminates in time polynomial in the input size since at every step, the cost function increases by at least 1 and as there is an upper bound, $O(mn)$, on the cost, the process must terminate in polynomial time.

Input

1	0	0	0	0	1	1	1	1	0
0	1	1	1	1	0	0	0	0	1
1	0	0	0	0	1	1	1	1	0
1	0	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	0	1
0	0	0	0	0	1	1	1	0	1
1	0	1	1	1	0	0	0	0	0

Figure 7.6: The input matrix for the example shown in Figures 7.7 to 7.9.

We illustrate the algorithm with a simple example in Figures 7.7 to 7.9 that uses 7 molecules with 10 sites.

Notice that of the two backtracking steps, the roll-back is the more expensive one. It is easy to see that this takes $O(m^2n^2)$ time when both the backtracking steps are used and $O(mn)$ time when the roll-back is not used. Our experimental experience suggests that the limited backtracking without roll-back suffices and our strong experimental intuition is that it is not crucial.

We experimented extensively with this algorithm, using different ways of ordering the sites, and, with and without roll-back, on simulated data. We obtained extremely positive results on predicting the restriction sites without using the roll-back in most cases. Only on very carefully hand constructed input data, does the roll-back have any effect.

7.2 Handling real data (with sizing errors)

In the last sections, we assumed that the cuts in the different molecules are perfectly aligned. However, in real data the actual positions of the cuts vary in each molecule (called sizing error). The handling of the real data is described below. In the first and second steps we discretize the data and compute an "approximate" position of the potential cuts which is used to orient the molecules in step

	Step 1, $j = 1$										
	1	2	3	4	5	$\bar{5}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	
	1	·	·	·	·						d_i
	1	·	·	·	·	·	·	·	·	0	1
*	1	·	·	·	·	·	·	·	·	0	1
	1	·	·	·	·	·	·	·	·	0	1
	1	·	·	·	·	·	·	·	·	0	1
*	1	·	·	·	·	·	·	·	·	0	1
*	1	·	·	·	·	·	·	·	·	0	1
	1	·	·	·	·	·	·	·	·	0	1
D_j	7										7

	Step 2, $j = 2$										
	1	2	3	4	5	$\bar{5}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	
	1	0	·	·	·						d_i
	1	0	·	·	·	·	·	·	1	0	2
*	1	0	·	·	·	·	·	·	1	0	2
	1	0	·	·	·	·	·	·	1	0	2
	1	0	·	·	·	·	·	·	0	0	1
*	1	0	·	·	·	·	·	·	0	0	1
*	1	0	·	·	·	·	·	·	0	0	1
	1	0	·	·	·	·	·	·	0	0	1
D_j	7	3									10

Figure 7.7: Illustration of the algorithm for the EBFC problem: the input consists of 7 molecules with 10 sites each. The sites are considered in the order $1, 2, \ldots, 5$. The $\cdot$ denotes the sites that have not been examined till that point. A row (or a molecule) that has been flipped, that is the corresponding indicator variable X_i is assigned 1, is marked with an asterisk. Note that $\sum_i d_i = \sum_j D_j$ at all the stages. Only step 5 necessitates the roll-back phase (where the site $j = 2$ assignment was switched from 0 to 1). The 1's in bold denote the cuts that do *not* contribute to the cost. See Figures 7.8 and 7.9 for the remaining steps.

Step 3, $j = 3$

	1	2	3	4	5	$\bar{5}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	
	1	0	1	·	·						d_i
	1	0	0	·	·	·	·	**1**	1	0	1
*	1	0	0	·	·	·	·	**1**	1	0	1
	1	0	0	·	·	·	·	**1**	1	0	1
	1	0	1	·	·	·	·	0	0	0	2
*	1	0	1	·	·	·	·	0	0	0	2
*	1	0	1	·	·	·	·	0	0	0	2
	1	0	1	·	·	·	·	0	0	0	2
D_j	7	3	1								11

Step 4, $j = 4$

	1	2	3	4	5	$\bar{5}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	
	1	0	1	1	·						d_i
	1	0	0	0	·	·	**1**	**1**	1	0	0
*	1	0	0	0	·	·	**1**	**1**	1	0	0
	1	0	0	0	·	·	**1**	**1**	1	0	0
	1	0	1	1	·	·	0	0	0	0	3
*	1	0	1	1	·	·	0	0	0	0	3
*	1	0	1	1	·	·	0	0	0	0	3
	1	0	1	1	·	·	0	0	0	0	3
D_j	7	3	1	1							12

Figure 7.8: Steps 3 and 4 of the example of Figure 7.6.

Step 5, $j = 5$

	1	2	3	4	5	$\bar{5}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	
	1	0	1	1	1						d_i
*	0	**1**	1	1	1	0	0	0	0	**1**	1
	0	**1**	1	1	1	0	0	0	0	**1**	1
*	0	**1**	1	1	1	0	0	0	0	**1**	1
	1	0	1	1	1	0	0	0	0	0	4
*	1	0	1	1	1	0	0	0	0	0	4
*	1	0	1	1	1	0	0	0	0	0	4
	1	0	1	1	1	0	0	0	0	0	4
D_j	1	-3	7	7	7						19

Roll-back (since $D_2 = -3$)

	1	2	3	4	5	$\bar{5}$	$\bar{4}$	$\bar{3}$	$\bar{2}$	$\bar{1}$	
	1	1	1	1	1						d_i
*	0	1	1	1	1	0	0	0	0	**1**	3
	0	1	1	1	1	0	0	0	0	**1**	3
*	0	1	1	1	1	0	0	0	0	**1**	3
	1	0	1	1	1	0	0	0	0	0	4
*	1	0	1	1	1	0	0	0	0	0	4
*	1	0	1	1	1	0	0	0	0	0	4
	1	0	1	1	1	0	0	0	0	0	4
D_j	1	3	7	7	7						25

Figure 7.9: The last steps of the example of Figure 7.6.

	Cuts	*Positions*
1	3	$0.1, 0.5, 0.9$
2	2	$0.3, 0.7$
3	3	$0.2, 0.6, 0.8$

(a) The input.

1	0	1	0	0	0	1	0	0	0	1	0
2	0	0	0	1	0	0	0	1	0	0	0
3	0	0	1	0	0	0	1	0	1	0	0

(b) M_{ij} from (a) using equation (7.2), $n = 11$.

1	0.78	1.0	0.78	0.0	0.78	1.0	0.78	0.0	0.78	1.0	0.78
2	0.0	0.0	0.78	1.0	0.78	0.0	0.78	1.0	0.78	0.0	0.0
3	0.0	0.78	1.0	0.78	0.0	0.78	1.0	0.78	1.0	0.78	0.0

(c) W_{ij} from M_{ij} using equation (7.3), $w = 1$.

Figure 7.10: A simple example showing 3 molecules with 3, 2 and 3 cuts respectively (a) with the corresponding binary matrix M_{ij} (b) and the weighted matrix W_{ij} (c).

---

3. In [37] the following observation has been made: there exists no discretization of the moleucles, in the absence of orientation information about the molecules, that will yield the correct result with high probability. However, in the presence of orientation information about the molecules, other straightforward approaches could be used. In the post-processing step (Step 4), we refine the positions of the cuts using simple heuristics to obtain the consensus cut sites. It is quite possible that for data sets that have more severe sizing errors, the heuristics of Step 4 may not suffice and a more elaborate scheme such as a dynamic programming mechanism or a maximum likelihood approach could be used on the raw correctly oriented molecules (the input vectors of Step 1) in Step 4, to obtain the correct result. A similar approach could also be used to obtain higher accuracy in the position of the consensus cut sites.

Input: m vectors where each vector represents a molecule. Each vector is

$$[K_i, l_1^i, l_2^i, \ldots, l_{K_i}^i]$$

where K_i is the number of cuts detected in molecule i and each cut is at a location

l_j^i from the *left end* of the molecule with $j = 1, 2, \ldots, K_i$. Further all the (implicit) molecule lengths are normalized, that is it is assumed that all the molecules are of 1 unit length. This is the form in which the data is available from the laboratory.
Output: A *map*, a vector, $[c, a_1, a_2, \ldots, a_c]$, where c is the number of consensus cuts in the map and $a_1, a_2, \ldots, a_c$ are the location of the cuts from the *left end* of the map.
Algorithm: The algorithm proceeds in four phases which we elaborate below.

(1) Handling positional variations: This consists of the following two steps.

1. Converting input to binary data, M_{ij}.
   Using a pre-defined number of discretization units, n, each molecule is converted to a vector of 0's and 1's. m such vectors form the $m \times n$ matrix M_{ij} as follows:

$$M_{ij} = \begin{cases} 1 & \text{if } \lceil (n-1) l_k^i \rceil + 1 = j, \text{ for some } k, \\ 0 & \text{otherwise.} \end{cases} \tag{7.2}$$

2. Converting binary data M_{ij}, to rational data, W_{ij}.
   The input data, in the real situation, does not depict the location of restriction sites accurately because of the error inherent in measuring the lengths of fragments that remain after digestion by the restriction enzyme. Thus a 1 at some site in the molecule might in fact signal a restriction site in one of its neighbors. This fuzziness is the result of coarse resolution and discretization, other experimental errors, or errors in preprocessing the data prior to constructing physical maps such as in the image processing phase.

   We tackle this problem as follows: we fix a window width w as a threshold parameter. If $M_{ij} = 0$, we locate the closest k to j where $\delta = |k - j| \leq w$, such that $M_{ik} = 1$ (if it exists), and make the assignment

$$W_{ij} = e^{-\frac{\delta^2}{(w/2)^2}}. \tag{7.3}$$

   If such a k does not exist, $W_{ij} = M_{ij}$. The underlying assumption we make is that the position of a cut is normally distributed around its true site with

a standard deviation of $w/2$. In other words, if there is a 1 at site j of a molecule, with 0 in locations $j-w$ to $j+w$, except j, we change the values from to a non-zero value that decreases as the position moves away from j (governed by the normal distribution). This results in entries that are not necessarily binary; they have values between 0 and 1 at each position.

This leaves us with the problem of finding the flip-cut as before except that now the entries of W_{ij} are rationals between 0 and 1.

Figure 7.10 shows a simple example.

(2) Computing potential sites: We first prune the sites and then filter them to give rise the exclusive version. Compute the scores for every site as $S_j = \sum_i \max(W_{ij}, W_{i\bar{j}})$, $\bar{S}_j = \sum_i \min(W_{ij}, W_{i\bar{j}})$, $1 \leq j \leq n/2$.

1. Pruning the sites: Notice that in the BFC, the cut sites never "interfered" with one another, since the locations were exact. However, cut sites that are closely located will interfere with one another when we compute S_j.

   Given $w \geq 0$, the ordered sequence $S_{j-w}, S_{j-w+1}, \ldots, S_j, \ldots, S_{j+w-1}, S_{j+w}$, is called the *distribution* of j. Let $j < k \leq n/2$, then the *joint distribution* of two sites j and k is given by the ordered sequence,

   $$S_{j-w}, S_{j-w+1}, \ldots, S_j, \ldots, S_k, \ldots, S_{k+w-1}, S_{k+w}.$$

   Figure 7.11 shows the joint distribution of two cut sites j and k in an ideal situation where the cuts j and k have the same number of missing cuts (false negatives). The case of more than two cuts is similar and we do not discuss it separately. In this step, our task is to locate the cut sites, given these distributions. One needs heuristics to predict the cut sites and we describe one such heuristic which is simple and has been very effective in practice. We simply locate the local maxima in the joint distribution.

   Local-peak Heuristic: Filter the columns as follows: Select S_j if if $S_{j-1} \leq S_j$ *and* $S_j \geq S_{j+1}$ and discard the sites S_{j-1} and S_{j+1}. In other words, we locate

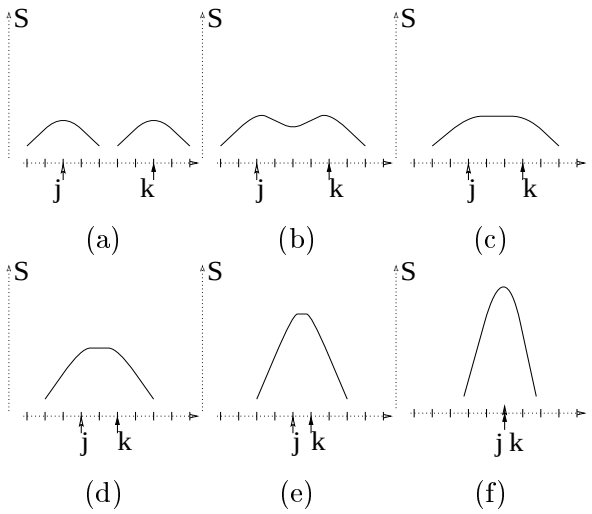


Figure 7.11: This shows the (approximate) joint distribution of two consensus cut sites at j and k, as the cuts move closer to each other from 5 units down to 0. (a)-(e): the "true" cut sites can be any one of the following - j & k, or, j & $\bar{k}$, or, $\bar{j}$ & k, or, $\bar{j}$ & $\bar{k}$. (f): the "true" cut sites are any one of j & k, or, $\bar{j}$ & $\bar{k}$. The shape of these distributions are the basis for the *local-peak* heuristic.

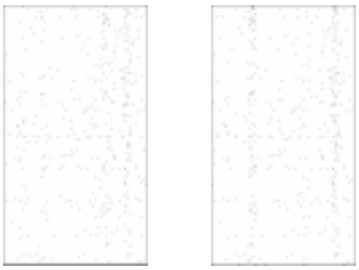
---

all the local peaks. Repeat this process until no more sites can be selected. Discard the left-over sites.

2. Filtering the pruned sites: Next, we remove two types of sites (in conjugate pairs) from consideration.

   (a) No-cut Heuristic. We remove those sites j and $\bar{j}$ that have fewer than $n\tau_p$ of $1'$s each; here τ_p (say, $1/50$) is a parameter we set from the knowledge of the error parameters in the experimental set up. We look upon these as sites where there is no underlying cut, but some molecules display the cut owing to false positive errors.

   (b) Symmetric-cut Heuristic. We remove all those sites j and $\bar{j}$ where the number of molecules with 1's at both these sites exceeds $m\tau_s$, for a parameter τ_s (say $1/10$), again set from the knowledge of the error parameters in the experimental set up. We look upon these as sites where



(a) (b)

Figure 7.12: An example illustrating the various steps in handling the real data. (a) An image corresponding to the (input) 361×200 binary matrix with a black dot for a 0 and a white dot for a 1 in the matrix (a dot corresponds to an element in the matrix). (b) The result of aligning the molecules/rows showing 2 consensus cut sites after applying steps (c) to (e) shown in Figure 7.13.

---

there are *symmetric* cuts at j and $\bar{j}$.

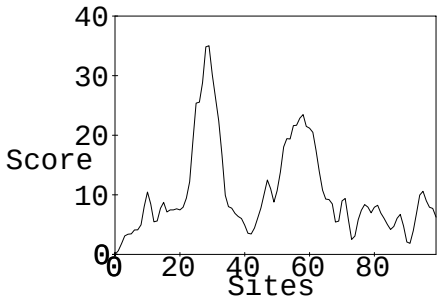
Now the remaining sites have the property that precisely one of j or its conjugate $\bar{j}$ will be a cut, and that reduces the problem to the exclusive version.

(3) Orienting the molecules: For the practical solution to the exclusive version, we follow the algorithm of the last section with the following modification: a molecule i has a cut at location j if there is a 1 in the w-neighborhood of location j in the binary matrix M_{ij}.

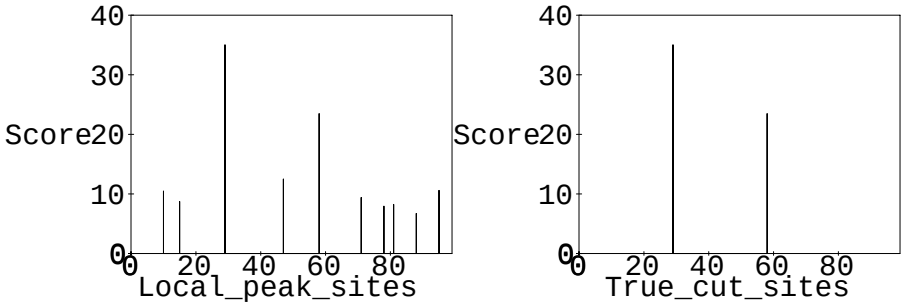
(4) Post-processing: We carry out the following postprocessing on the results obtained in the last step.

First, the molecules are aligned using the orientation computed by the algorithm.

1. Refining the location of the consensus cut sites.
   We recompute $S_j = \sum_i a_{ij}$ for each j in the neighborhood of the computed cut site. and choose the the j with the maximum value of S_j.



(c)



(d) (e)

Figure 7.13: See Figure 7.12 for the input. (c) The plot of the scores of sites S_j versus the sites j, $j = 0 \ldots 99$. (d) The sites selected using the *local-peak heuristic*. (e) The true sites detected using the *No-cut heuristic*. This is the input to the exclusive version. In this simple example the exclusive version has to deal with just 4 (2×2) sites! The possible sites are 29, 170, 57 and 141. (Note that site 170 is the conjugate of site 29, and site 141 is that of 57.) The algorithm gives sites 57 and 170 as the consensus cut sites as shown in (b).

2. Detecting multiple consensus cut sites.

   For each cut site j, we count the number of molecules (rows) that show multiple cuts in the small neighborhood of j. If the number of such molecules $> m\tau_m$, for some pre-defined threshold, τ_m, multiple cuts are declared in that location.

The solution sites are the ones discussed in the last step along with the ones that were declared to be symmetric in the symmetric-cut heuristic of step 2.

An alternate way is to simple use the orientation information, obtained in the last step, and do a straightforward dynamic programming to obtain the locations of each of the cut sites.

This concludes the description of the algorithm. We present the results on real data.

7.3 Experimental results

All the real data we used for our experiments were obtained from the W. M. Keck Laboratory for Biomolecular Imaging of the Department of Chemistry, New York University [2]. The input to our problem from the laboratory is a set of molecules, each having unit length, with the positions of their restriction cuts given as lengths from the left end. We discretize this to 200 units[3]. All the DNA molecules in our experiment shown here are lambda vectors having $48,000$ base pairs. Thus, each discretization unit here represents about 240 base pairs. The restriction enzymes used were *AvaI* and *EcoRI*. In all the experiments carried out, including the one with about 3000 molecules, the program takes less than thirty seconds of elapsed

[2] I am grateful to the team lead by Dr. David Schwartz at the W. M. Keck Laboratory for Biomolecular Imaging of the Department of Chemistry, NYU, for providing the data. Thanks in particular to Joanne Edington and Junping Jing whose data has been used in this chapter (Figures 7.14 to 7.19). I would also like to thank Estarose Wolfson, Ernest Lee, Alex Shenker, Brett Porter, Ed Huff, Thomas Anantharaman, Davi Geiger and Bud Mishra for providing the data in the form used by the algorithm.

[3] A larger discretization does not give higher accuracy, as the precision of the imaging and the pre-processing stages is limited. This number of 200 was chosen after discussion with Thomas Anantharaman and other researchers from the laboratory.

time on a SPARCstation to compute the restriction map. Here we show a sample of four experimental results on real data.

For most of the real data available for experimentation, best results were obtained for these parameter values: $n = 200$, $w = 2$, $\tau_s = 0.1$, $\tau_p = 0.5$. For the data set illustrated in Figure 7.17, $\tau_p = 0.4$ gave the best results.

Comparison of the different schemes. The results of the different approaches are compared using "similarity measure" on the resulting outputs. To keep the measure independent of what the program computes, we do not use the computed map for the measure, but instead use the oriented molecules, whose orientations were computed by the algorithm.

We define two such measures. The first measures the number of matching cuts in pairs of molecules, called *Sum-of-pairs*, Sim_1.

$$Sim_1 = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}}{\binom{m}{2}},$$

where for a pair of molecules, i and j, c_{ij} denotes the number of matching 1's (that appear within, say $\delta > 0$, of each other).

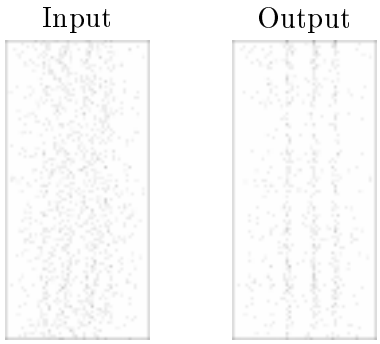
The second measure Sim_2, also looks for cuts in a pair of molecules, but prefers higher number of cut matches per pair of molecules: we call this *Sum-square-of-pairs*, Sim_2. We achieve this by simply squaring the number of common cuts per molecule pair. Thus,

$$Sim_2 = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij}^2}{\binom{m}{2}}.$$

We tested the different schemes on a wide variety of synthetic and real data. Although in most cases the cut sites were picked accurately, some molecules were assigned wrong or no orientation. Table 7.1 shows the scores of the similarity measures on a set of real data available from the laboratory. However the errors in assigning the correct orientation to each molecule improved from random to sorted to MaST site orders which is reflected in the similarity measure scores.

Data (c cuts : m mols)	Using Mutual Agreement		Using Explicit Map					
			Random Order		Sorted Order		MaST Order	
	Sim_1	Sim_2	Sim_1	Sim_2	Sim_1	Sim_2	Sim_1	Sim_2
Synthetic data: Fig 7.14 (3: 415)	0.112	0.251	0.611	0.916	0.749	1.032	1.19	1.513
λ DNA, *AvaI*: Fig 7.15 (8: 800)	0.133	0.432	0.876	1.274	0.882	1.283	1.879	2.195
λ DNA, *AvaI*: Fig 7.15 (8: 120)	0.121	0.501	0.823	1.165	0.914	1.391	1.636	2.294
λ DNA, *EcoRI*: Fig7.17 (5: 333)	0.032	0.362	0.511	0.801	0.513	0.861	1.092	1.315
λ DNA, *EcoRI*: Fig 7.17 (5: 403)	0.111	0.239	0.501	0.785	0.692	0.798	1.008	1.116
λ DNA, *ScaRI*: Fig 7.18 (5: 281)	0.235	0.356	0.596	0.879	0.612	0.914	1.036	1.336
λ DNA, *BamHI*: Fig 7.19 (5: 196)	0.229	0.318	0.616	0.910	0.858	0.993	1.087	1.46

Table 7.1: Comparison of the various schemes: The real data sets shown in this table correspond to the ones illustrated in Figures 7.14 to 7.19. Using the approach of mutual agreement, the correct map was not obtained only for the case using *AvaI* with λ DNA. All the rest of the schemes gave the correct map but varied in the similarity measure as shown, with the third scheme displaying best results. Sim_1 refers to the sum-of-pairs measure and Sim_2 refers to the sum-square-of pairs measure. $\delta = 2$ was used to compute Sim_1 and Sim_2. The small value of the measure indicates that the average number of cuts that agree in a pair of molecules is small (for the fixed w). The unexpectedly small difference between Sim_1 and Sim_2 indicates that multiple matches between molecules (two molecules agreeing on multiple cut sites) is not very common in practice (for fixed w).



Known map vs computed map

Positions (1-200)			
True Sites	59	85	120
Computed Sites	60	85	120

Cuts	Standard Deviation	Expression of cuts (%)
1	0.816	4.1
2	0.370	3.7
3	0.413	12.0

Figure 7.14: Synthetic data (derived from anonymous experimental data). Number of molecules is 415. The MAX ABS percentage error is 0.5%.

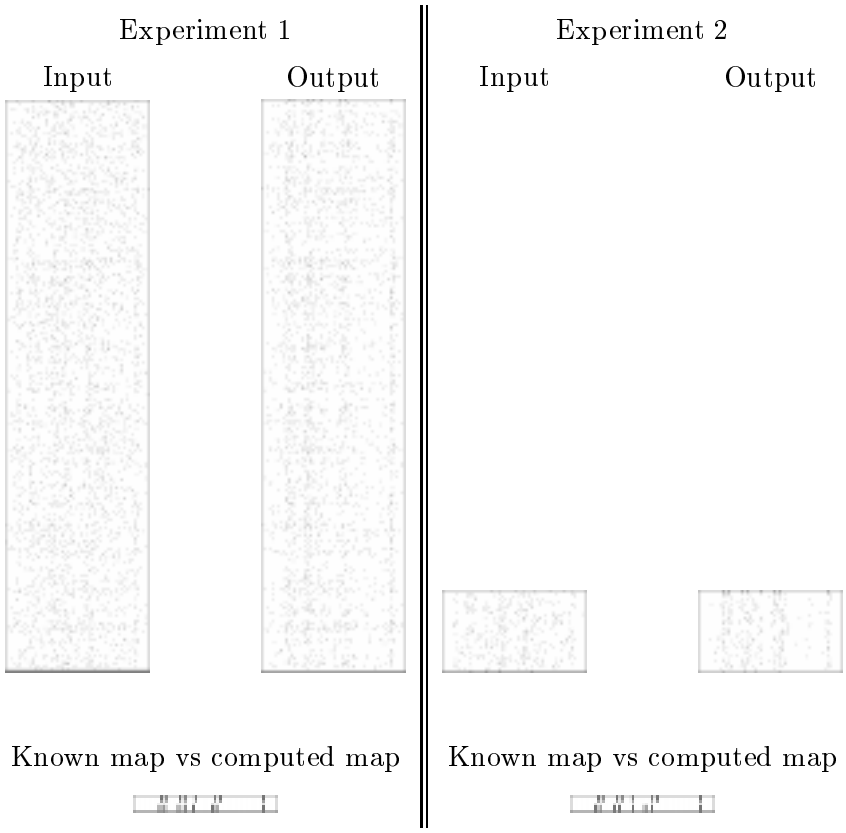
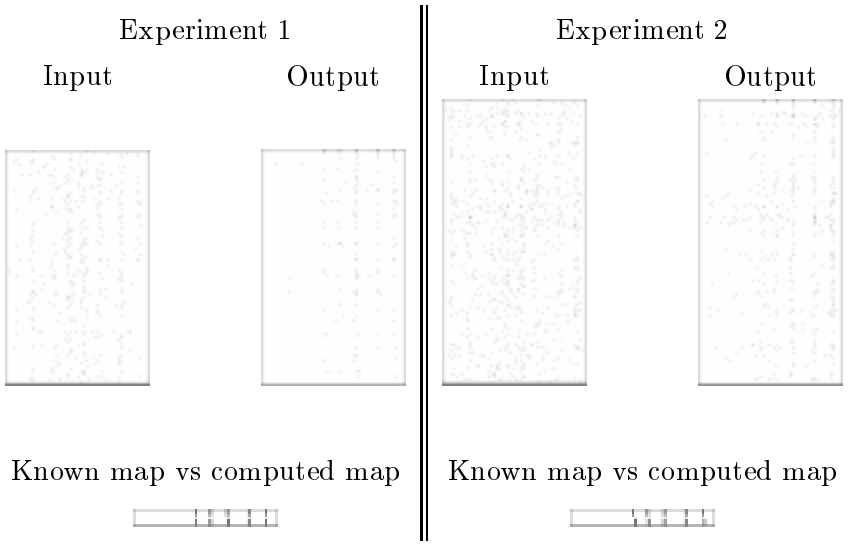


Figure 7.15: Clone: λ DNA, Enzyme: *AvaI*. Number of molecules in the first experiment is 800 and 120 in the second experiment. See Figure 7.16 for a statistical evaluation of the results.

Positions (1-200)								
True Sites	38	45	64	71	84	114	120	180
Computed Sites (Experiment 1)	37	44	63	71	83	109	115	180
Computed Sites (Experiment 2)	36	43	62	70	85	110	114	180

Cuts	Experiment 1		Experiment 2	
	Standard Deviation	Expression of cuts (%)	Standard Deviation	Expression of cuts (%)
1	0.140	15.8	0.110	23.2
2	0.164	11.6	0.133	19.3
3	0.177	14.4	0.145	18.1
4	0.215	8.5	0.165	10.1
5	0.225	7.4	0.183	12.3
6	0.240	7.6	0.179	13.5
7	0.198	10.6	0.171	19.1
8	0.128	15.5	0.106	18.2

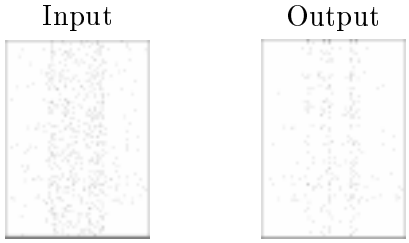
Figure 7.16: The data is shown in Figure 7.15. The MAX ABS percentage error is 2.5% in the first experiment and 3.0% in the second experiment. The first experiment uses 800 molecules and the second only 120, yet the algorithm extracts the correct maps as shown.



Positions (1-200)					
True Sites	87	107	130	160	183
Computed Sites (Experiment 1)	88	107	131	160	184
Computed Sites (Experiment 2)	89	108	130	160	184

Cuts	Experiment 1		Experiment 2	
	Standard Deviation	Expression of cuts (%)	Standard Deviation	Expression of cuts (%)
1	0.606	2.1	0.436	2.7
2	0.490	3.0	0.309	4.5
3	0.219	9.0	0.243	8.4
4	0.335	4.2	0.243	7.4
5	0.286	6.3	0.193	11.9

Figure 7.17: Clone: λ DNA, Enzyme: *EcoRI*. Number of molecules in the first experiment is 333 and 403 in the second experiment. The MAX ABS percentage error is 0.5% in the first experiment and 1.0% in the second experiment. However, the data in the second experiment is noisier than the first, yet the algorithm extracts the correct map.

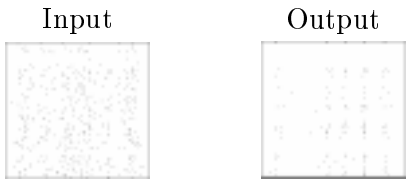


Known map vs computed map

Positions (scale of 1-200)					
True Sites	66	88	95	124	132
Computed Sites	65	86	94	125	132

Cuts	Standard Deviation	Expression of cuts (%)
1	0.329	6.4
2	0.328	6.0
3	0.242	11.7
4	0.215	14.2
5	0.353	5.3

Figure 7.18: Clone: λ DNA, Enzyme : *ScaI*. Number of molecules = 281. The MAX ABS percentage error is 1.0%.



Known map vs computed map

Positions (scale of 1-200)					
True Sites	23	92	115	142	172
Computed Sites	23	91	116	143	172

Cuts	Standard Deviation	Expression of cuts (%)
1	0.427	3.4
2	0.378	4.2
3	0.441	8.3
4	0.317	7.1
5	0.452	3.3

Figure 7.19: Clone: λ DNA, Enzyme : *BamHI*. Number of molecules = 196. The MAX ABS percentage error is 0.5%. The output looks much sparser than the input, since the orientation of a large number of molecules (not displayed) could not be ascertained.

Experimental results using linear time ($2mn$) algorithm. Although we studied three different orderings of the sites for the heuristic algorithm, we found that the simple scheme of using a sorted order of sites without roll-back was very effective in practice. The real data appears to be very well behaved and if we are willing to accept some wrong assignment of orientations (or unknown orientation of the molecules), the correct maps, with small positional errors, are obtained very quickly.

For the data shown in Figures 7.14 to 7.19, the sorted order, without roll-back, of the potential cuts sites was used. We now take a closer look at one of the data sets, shown as experiment 1 in Figures 7.15 and 7.16. At the pruning stage, all the sites except [4] 20/180, 37/163, 44/156, 63/137, 71/129, 83/117, 91/109 were detected as no consensus cut sites. Further, 83/117 was declared to be a symmetric cut. The remaining potential sites were considered in the decreasing order of potential, that is, 20/180, 71/129, 91/109, 37/163, 63/137, and, 44/156. After the alignment of the molecules, the symmetric cut at 83/117 was refined to 83 and 115 at the post-processing stage.

For exposition, consider Figure 7.14.

1. The image on the left is the input data and the one on the right is the output of our algorithm. Each row of the image is a molecule with a black dot indicating a restriction site. The output image shows the molecules flipped as per the solution computed by the algorithm, with the computed cuts marked on the top of the image by tiny bars. In some cases, the output image is sparser than the input image since we do not display the molecules, i, with $d_i = 0$.

2. At the bottom of the two images, we display a smaller image which has two rows of bars: the top row shows the true position of cuts as provided by the laboratory and the second gives the positions computed by our algorithm.

3. The table below (2) shows the position of the true and the computed cuts in $1 - 200$ scale.

4. The table below (3) displays statistics on the distribution of the cuts in the input relative to the computed ones. The standard deviation of the displacement

[4] We represent a site j by $j/\bar{j}$.

of a restriction site of a molecule from the computed cut site, as computed by the program, is shown, as is the expression of a cut site in the molecules (the number of molecules that have a 1 in that site) as a percentage. QED

We compare the map determined by our algorithm with the true map as follows. We define a one-to-one correspondence between the restriction sites in both, maintaining a left-to-right order. The number of the restriction sites in both must match. The MAX ABS error is the maximum of the absolute distance of a cut site in the true map from its corresponding cut in the computed map. In our experiments, the MAX ABS percentage error is $0 - 3\%$, and we never missed a cut or found an extra cut. Also, there have been input instances (eg., one with a sample of 2910 molecules) with 0% error!

Figure 7.15 shows two experiments where one uses a sample of 800 molecules and the other uses only 120 molecules. Figure 7.17 shows two experiments where the data in the second experiment is much noisier than the first. In all these cases the algorithm performs robustly enough to extract the right maps.

Naksha[5] is an implementation of this algorithm and a demonstration is available on the author's homepage (`http://www.cs.nyu.edu/parida`). Interested readers can also obtain the test results on simulated data from the homepage.

[5]The word *naksha* in Hindi means a map.

Chapter 8

Generalizations of the Problem

This chapter consists of two sections. In the first section we discuss the modeling of other error sources such as missing fragments in the data, bad or spurious molecules and sizing errors (though this error has been handled in the chapter on practical algorithms). In the second section we deal with the K-populations problem.

8.1 Modeling Other Errors

In this section we study the computational complexity of the problems that model the different error sources. We show that all the problems, except the one incorporating the spurious molecules, are MAX SNP hard and give approximation bounds for each.

Continuing the Ann and John game of Section 4.1, John can also make the following changes (including the ones discussed earlier):

1. Spurious Molecules: John can throw out some, say k, molecules from this data set and throw in k random strings of 0's and 1's in its place.

   In practice, some "bad" molecules get into the sample population; these need to be invalidated and not used in the map computation.

2. Missing Fragments: John can remove some fragments of the string (the substring between two 1's).

This corresponds to fragments that get *desorbed* (washed away) during the experiment, which poses problems for BAC DNA, although not for cosmids and λDNA [3].

Now, the alignment of the rows/molecules refers to the following additional assignments:

5) Labeling a molecule as *spurious* or not.

6) Assigning a *left-flushed* or *right-flushed* or any other positioning of each molecule.

8.1.1 Modeling spurious molecules

The Binary Partition Cut (BPC) problem takes into account the presence of spurious or bad molecules (along with false positive and negative errors). We can define two kinds of cost functions for the problem: one that maximizes the number of cuts in each molecule corresponding to a cut of the consensus physical map (BPC) and the other that maximizes the number of cuts in the consensus map ($\text{BPC}_{\max}$). Recall that each consensus cut site must satisfy the digestion rate criterion. A variation of the latter where the information that *the total number of bad molecules is known exactly*, is shown to be NP-complete in [3]. However, we show that the two original problems have efficient polynomial time solutions.

Let the input binary $m \times n$ matrix be $[M_{ij}]$. Associate indicator variables X_i, $i = 1, 2, \ldots, m$, with every row which takes a value 1 if the molecule is good and 0 if it is spurious. Let Y_j, $j = 1, 2, \ldots, n$, be an indicator variable associated with every column that takes on a value of 1 if it is a consensus cut and 0 otherwise. Let the digestion rate be p_j for column j, $j = 1, 2, \ldots, n$. Formulating the problem along the lines of formulating the BFC problem in equation (5.1) we obtain the the following optimization problem:

$$\max \left\{ \sum_{j=1}^{n} Y_j \left(\sum_{i=1}^{m} X_i M_{ij} - p_j m \right) \right\}. \tag{8.1}$$

Notice that the term mp_j is used to ensure that the number of 1's along a consensus cut site j (ignoring the spurious molecules) is at least mp_j.

Consider the corresponding minimization problem

$$\min \left\{ \sum_{i=1}^{m} \sum_{j=1}^{n} -M_{ij} X_i Y_j + \sum_{j=1}^{n} m p_j Y_j \right\} \tag{8.2}$$

which is a submodular function [39], hence BPC has a polynomial time solution.

Corollary 6 *The BPC$_{\max}$ problem has a polynomial time solution.*

Proof: If n_o is the number of consensus cuts in the optimal solution then there exists no sub-optimal solution with $n > n_o$. This is because the new optimal alignment can be obtained from this sub-optimal giving a larger n_o, which is a contradiction. Hence a solution to the BPC problem gives a solution to the BPC$_{\max}$ problem. QED

8.1.2 Modeling missing fragments

The Binary Shift Cut (BSC) problem takes into account missing fragments along with false positive and negative errors [1]. As in the BPC problem formulation, we can define two kinds of cost functions for the problem: one that maximizes the number of cuts in each molecule corresponding to a cut of the consensus physical map (BSC) and the other that maximizes the number of cuts in the consensus map (BSC$_{\max}$). Recall that each consensus cut site must satisfy the digestion rate criterion. The BSC$_{\max}$ is shown to be NP-complete in [3]. We show that both the approaches to the problem are MAX SNP hard and also give bounds on the approximation factors achievable.

We show that BSC is MAX SNP-hard in the following theorem.

Theorem 5 *The BSC problem is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating this problem within a factor of $1 - \epsilon$ is NP-hard.*

Proof: We prove the result for a special case of the BSC problem where every molecule is such that either the left or the right fragment (not both) is missing;

[1] The authors in [3] introduced the missing fragments error, and the same problem was called "Binary Shift Cut" in [14] in keeping with the earlier naming convention.

the missing fragment is exactly one unit in all the molecules. In the aligned configuration, a column j is in a cut only if the number of cuts is at least $p_j m$, which is defined in the proof of step 2.

The proof is along the lines of that of theorem 1. We give an outline of the proof here with the full detail in Appendix B.

The proof has three steps. Let C_X^* denote the cost of the optimal solution and C_X denote the cost corresponding to a particular solution for problem X.

Step 1 . We show a reduction of an instance of the maximum cut (MC) problem with n vertices and e edges to an instance of the bipartite maximum cut (BMC) problem (which is the maximum cut problem on a bipartite graph with weights $+1$ or -1 on the edges) such that

(**1.1**) there exists a correspondence between the two solutions,

(**1.2**) $4C_{MC}^* = C_{BMC}^* - 4e - 3n$, $4C_{MC} \geq C_{BMC} - 4e - 3n$, and,

(**1.3**) the number of edges with positive weights in the BMC is $8e + 2n$.

Step 2 . We show the reduction of an instance of the BMC problem to an instance of the BSC problem such that

(**2.1**) there exists a correspondence between the two solutions, and,

(**2.2**) $2C_{BSC} - c = C_{BMC}$, where c is the number of 1's in the BSC matrix.

Step 3 . For some $\epsilon > 0$, let C^* denote the optimal solution and $\tilde{C}$ denote an approximate solution with $\tilde{C}_{BSC} \geq (1-\epsilon)C_{BSC}^*$.

$$\begin{array}{rcll}
\tilde{C}_{MC} & \geq & \frac{\tilde{C}_{BMC}-4e-3n}{4} & \text{(using Step 1.2)} \\
& = & \frac{2\tilde{C}_{BSC}-12e-5n}{4} & \text{(using Steps 1.3 \& 2.2)} \\
& \geq & \frac{(1-\epsilon)2C_{BSC}^*-12e-5n}{4} & \text{(by definition of } \tilde{C}_{BSC}) \\
& = & \frac{(1-\epsilon)(C_{BMC}^*+8e+2n)-12e-5n}{4} & \text{(using Step 2.2)} \\
& = & \frac{(1-\epsilon)C_{BMC}^*-4e-3n}{4} - \frac{(8e+2n)\epsilon}{4} & \\
& \geq & (1-\epsilon)C_{MC}^* - 2.5e\epsilon & \text{(using Step 1.2 \& } e \geq n) \\
& \geq & (1-\epsilon)C_{MC}^* - (2.5\epsilon)2C_{MC}^* & \text{(since } C_{MC}^* \geq e/2) \\
& = & (1-6\epsilon)C_{MC}^*. &
\end{array} \tag{8.3}$$

This shows that given a polynomial time approximation scheme (PTAS) for BSC, we can construct a PTAS for MC (using steps 1 and 2 that give the correspondence between the two solutions), which is a contradiction; hence BSC does not have a PTAS.

We skip the details of steps 1 and 2 here and present it in Appendix B for the interested reader. QED

Corollary 7 *Achieving an approximation ratio* $1 - \Upsilon/6$ *for BSC is NP-hard.*

Theorem 6 *The* $BSC_{\max}$ *problem is NP-hard. Further, there exists a constant* $\epsilon > 0$ *such that approximating this problem within a factor of* $1-\epsilon$ *is NP-hard and achieving an approximation ratio* $(1 - \Upsilon/6)\frac{p_{max}}{p_{min}}$ *for* $BSC_{\max}$ *is NP-hard.*

The proof is identical to that of theorem 3.

Corollary 8 *There does not exist a polynomial time algorithm, unless* $P = NP$*, that guarantees the estimation of* $(1 - \Upsilon/6)$ *of the total number of consensus cuts when digestion rates at each site is the same taking missing fragments into account.*

8.1.3 Modeling sizing errors of the fragments

The Binary Sizing error Cut (BSeC) problem takes into account varying size of fragments in the molecules (along with false positive and negative errors). Although this error has been handled in the chapter dealing with the practical algorithms, we discuss its MAX SNP-hardness here as the proof is similar to that of the BSC problem. As in the BPC problem formulation, we can define two kinds of cost functions for the problem: one that maximizes the number of cuts in each molecule corresponding to a cut of the consensus physical map (BSeC) and the other that maximizes the number of cuts in the consensus map ($\text{BSeC}_{\max}$). Recall that each consensus cut site must satisfy the digestion rate criterion. The $\text{BSeC}_{\max}$ is shown to be NP-complete in [3]. We show that both the approaches to the problem are MAX SNP hard and also give bounds on the approximation factors achievable.

Theorem 7 *The BSeC problem is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating it within a factor of $1 - \epsilon$ is NP-hard and achieving an approximation ratio $1 - \Upsilon/6$ for BSeC is NP-hard.*

Proof: Consider the special case of the BSeC problem where only the end fragments of each molecule have differing sizes and the sizes vary by at most one unit [2] along with any apparent size differences of fragments due to false negative or false positive errors. The proof now proceeds along similar lines as in the proof of theorem 5. QED

Theorem 8 *$BSeC_{\max}$ problem is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating this problem within a factor of $1 - \epsilon$ is NP-hard and achieving an approximation ratio $(1 - \Upsilon/6)\frac{p_{max}}{p_{min}}$ for $BSeC_{\max}$ is NP-hard.*

Again, the proof is along the lines of the proof of theorem 6.

Corollary 9 *There does not exist a polynomial time algorithm, unless $P = NP$, that guarantees the estimation of $(1 - \Upsilon/6)$ of the total number of consensus cuts when digestion rates at each site is the same, taking sizing errors into account.*

8.1.4 Summary

The computational complexity of the problems modeling the different errors are summarized in table 8.1. Some of these problems were open and the other known results have been improved here (also in [41]).

8.2 The K-Populations Problem[3]

8.2.1 Introduction

Here, we explore some algorithmic and complexity questions related to the use of optical mapping to study a population of homologous DNA fragments [45]. In

[2] Also, the acceptable deviation from the exact location of the cut is zero.

[3] Portions of this section also appear as "Partitioning K Clones: Hardness Results and Practical Algorithms for the *K-Populations Problem*", coauthored with Bud Mishra, Proceedings of the ACM Conference on Computational Molecular Biology (RECOMB), New York, March 1998.

Problem	Complexity class	Approx factor (upper bound)	Errors
Exclusive Binary Flip Cut† (EBFC)	MAX SNP-hard	$1 - \Upsilon/3$	Orien. uncert.
Binary Flip Cut* (BFC)		$1 - \Upsilon/3$	
$\mathrm{BFC}_{\max K}{}^\dagger$		$(1 - \Upsilon/3)\frac{p_{max}}{p_{min}}$	
Consistency Graph* (CG), Weighted Consistency Graph* (WCG), d-wise Match* (dM)		$1 - \Upsilon/3$	
Binary Partition Cut* (BPC) $\mathrm{BPC}_{\max K}{}^\dagger$	$\mathcal{P}$	–	Bad mols
Binary Shift Cut* (BSC)	MAX SNP-hard	$1 - \Upsilon/6$	Missing frags
$\mathrm{BSC}_{\max K}{}^\dagger$		$(1 - \Upsilon/6)\frac{p_{max}}{p_{min}}$	
Binary Sizing-error Cut (BSeC)	MAX SNP-hard	$1 - \Upsilon/6$	Sizing errors
$\mathrm{BSeC}_{\max K}{}^\dagger$		$(1 - \Upsilon/6)\frac{p_{max}}{p_{min}}$	

Table 8.1: Computational Problems from Optical Mapping: All the problems model false positive and false negative errors as well. Problem* denotes unknown complexity until this work and Problem† denotes the best known result for the hardness of this problem was that it was NP-complete. The Binary Partition Cut (BPC) problem has been modified (slightly) to admit a polynomial time solution. $p_{min} = \min_j p_j$, and $p_{max} = \max_j p_j$ are defined by the given problem. (Υ denotes the upper bound on the polynomial time approximation factor of the well-known max cut problem.)

practice, the most interesting case involves just two populations ($K = 2$) where one examines populations related to diploid DNA. However, there are other situations, e.g., dealing with PCR products, several strains of a microorganisms, etc., where K could be arbitrarily large. Computationally, it is also interesting to study how this generality (when K is unconstrained) affects the problem.

We focus on a somewhat idealized model. We assume that we are given m molecules, each one derived from one of K different clones. We wish to partition the m molecules into K different disjoint classes (K "populations"), each class corresponding to a distinct clone. We then wish to compute and output the K distinct ordered restriction maps (one for each population) so that the dissimilarities among the ordered restriction maps can be quickly distinguished. The output is given in a novel data structure that can quickly identify the dissimilar regions in the maps.

In an ideal setting, if the correct ordered restriction map for each molecule can be made available, then the resulting computational problem is rather trivial. In practice, however, the single molecule restriction map that can be computed by the image processing algorithm will be governed by several error processes: missing restriction cut sites due to partial digestion (false negatives), spurious optical cut sites (false positive), sizing error, missing fragments, error in assigning the orientation of the molecule, presence of other spurious molecules etc.

Even though we ignore all but the first two error processes, the resulting computational problem still poses many challenges from a purely algorithmic point of view. We characterize these structural difficulties, propose an extension of an earlier heuristic and explore its power empirically.

In our simplified model, each molecule (m in total) is represented as a binary vector of length n:

$$A_i = (a_{i1}, a_{i2}, \ldots, a_{in}) \in \{0,1\}^n,$$

and the set of m molecules is represented by an $m \times n$ 0-1 matrix. A 1 in location a_{ij} is meant to indicate a cut site at the location j in the i^{th} molecule and may be a true restriction cut or a false optical cut. Thus, if $a_{ij} = 1$, then either it is a true restriction site and the correct map for the corresponding clone has a restriction

site at j or it is a false optical cut and the correct map has no restriction site at j. Conversely, if $a_{i,j} = 0$, then either it is a missing restriction site and the correct map has a restriction site at j or it is simply not a restriction site and the correct map does not have a restriction site at j. Our goal is to devise an algorithm to find a partition of the molecules into K populations so that each restriction site in the proposed map for a population is supported by "enough" restriction sites at the corresponding location in the same population. We shall define a cost function that formalizes this notion and examine the complexity of the resulting combinatorial optimization problem.

Roadmap. In the next section, we reformulate the problem in a purely combinatorial setting and show that the resulting problem as well as computing an arbitrarily good approximation is computationally infeasible (NP-hard). Note that our results strengthen the negative results shown earlier [3] for the 2-population problem, by first getting rid of the constraints on the size of the populations (Problem 5 in [3]) and secondly by demonstrating the inapproximability of the problem. (Also, see [41] and Chapters 5 and 8.1.4.) In Section 8.2.3, we give a 0.756-approximation algorithm for a 2-populations problem. In Section 8.2.4, we propose a simple heuristic extending our earlier heuristic for a single population problem [38] and demonstrate experimentally that the resulting simple polynomial time algorithm finds the maps correctly for reasonable values of the parameters (partial digestion rate of 50%, negligibly small optical false cut rate, upto 4 populations). This suggests that it may be possible to extend our heuristics to other approaches (e.g., Bayesian schemes) that can handle other sources of error (most notably, sizing errors) without a severe penalty in computational complexity, as long as partial digestion rate is significantly high compared to the false cut rate and for a reasonable number of populations. Finally, we describe our empirical results based on the synthesized data and explore its limitations. In a concluding section, we interpret the significance of our results.

8.2.2 Complexity

For the sake of complexity analysis, we may assume that the only errors to be handled are false negative and false positive errors (due to partial digestion and optical cuts, respectively[4]). Given a set of molecules from K different populations, the task is to identify the different populations and the map of each of the population. The problem can be formally stated as follows. We are given an $m \times n$ matrix $[a_{ij}]$ denoting m molecules and n sites with p_j, $j = 1, 2, \ldots, n$ defined for each site j, which is a lower bound on the fraction of the size of the population where the site j is a consensus cut[5]. The task is to maximize the number of 1's in the consensus cut columns and the number of 0's in the non-consensus cut columns in each population.

Let Y_{11}, Y_{12}, ..., Y_{1n}, Y_{21}, Y_{22}, ..., Y_{2n}, ..., Y_{K1}, Y_{K2}, ..., Y_{Kn} be the (nK) indicator variables associated with each site (column) and population with the following connotation:

$$Y_{kj} = \begin{cases} 1, & \text{if } j \text{ is a consensus cut site in population } k; \\ 0, & \text{if } j \text{ is not a consensus cut site in population } k. \end{cases}$$

$\text{Pop}(A_i) = k$ where $1 \leq k \leq K$, denotes that molecule i belongs to population k. Let

$$X_{ik} = \begin{cases} 1, & \text{if } \text{Pop}(A_i) = k; \\ 0, & \text{otherwise.} \end{cases}$$

and $m_k = \sum_{i=1}^{m} X_{ik}$. The K-populations problem then can be formulated as a

[4]Recall that we may also model sizing errors or uncertainty in orientation or errors due to missing fragments or spurious molecules, etc. However, these errors only result in higher worst-case complexity of the problem and complicate the constructions used in the proof.

[5]In general, p_j depends on the digestion rate, which cannot be always expected to be known a priori, and may have to be estimated from the data (for instance, by an MLE method). A simple heuristic estimator can be formulated by assuming that all the p_j's are equal and is given by

$$p = (1 - \lambda)(\tilde{p}/m)\frac{\sum \#\text{cuts in} A_i}{\text{length}(A_i)},$$

where $\tilde{p}$ denotes the cutting efficiency (e.g., $\approx 1/4,000$ for a 6-cutter) and $(1 - \lambda)$ is a shrinkage factor that compensates for the bias due to false optical cuts.

combinatorial optimization problem; *maximize the following function*:

$$C(p_j, [a_{ij}], X_{ik}, Y_{kj}) = \sum_{k=1}^{K} \left\{ \sum_{j=1}^{n} \sum_{i=1}^{m} Y_{kj}[X_{ik} a_{ij} - p_j m_k] \right\}. \tag{8.4}$$

and the *cost* of the configuration (given by optimizing the above function) is the total number of 1's in the consensus cut columns of each population. Note that the function given by equation (8.4) is simply

$$\sum_{k=1}^{K} \sum_{j=1}^{n} \sum_{i=1}^{m} X_{ik} Y_{kj} [a_{ij} - p_j].$$

The cost function used here is a straightforward generalization of the "consensus with data" cost function used for the 1-population problem described in [43] and Chapter 5. This is somewhat simplified by the fact that we include only those terms that model the false positive and false negative errors. This cost function can also be heuristically justified by considering its expected value over random $m \times n$ 0-1 matrices, where the entries of the matrix are Bernoulli random variables with negligibly small false positive probability and with a false negative probability determined by p_j's.

Theorem 9 *The K-populations problem is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating the problem within a factor of $1-\epsilon$ is NP-hard.*

Proof Sketch. We will prove the result for a special case of the K-populations problem with the following characteristics:

1. $K = 2$. (A two-populations problem.)
2. $\sum_{k=1}^{K} Y_{kj} = 1$ for all j. (Every cut belongs to exactly one population.)
3. The molecules come in pairs such that no two elements of the pair belong to the same population. (This also ensures that the two populations are of the same size.)

4. $p_j = \sum_i a_{ij}/2m$ for all j, where m is the total number of molecules.

To prove the inapproximability of this special case of the K-populations (KP) problem [6], we use the recent technique of giving a *gap-preserving reduction* of a Max SNP-hard problem, the Max-cut (MC) problem, to our problem [5].

The proof has three steps. In step 1 we show the reduction of an instance of the MC problem to an instance of a bipartite Max-cut problem, with weights on the edges as 1 or -1 (BMC). In step 2 we show the reduction of an instance of the BMC problem to an instance of the KP problem and in the final step we show that the reduction is *gap-preserving*.

Let C^*_X denote the cost of the optimal solution and $\tilde{C}_X$ denote the cost of an approximate solution of the problem X.

Step 1. (MC to BMC reduction.)
Consider an MC problem on a graph with vertices and edges $(V, E), n = |V|, e = |E|$. Construct an instance of BMC with $(\tilde{V}, \tilde{E})$ as follows: For each $v_i \in V$, with degree d_i, construct $2(d_i + 1)$ vertices, $V_{gadget_i} = \{v'_{i0}, v'_{i1}, \ldots, v'_{id_i}, v''_{i0}, v''_{i1}, \ldots, v''_{id_i}\}$. Further, $wt(v'_{ij}, v''_{ij}) = wt(v'_{i0}, v''_{ij}) = wt(v'_{ij}, v''_{i0}) = -1, j = 1, 2, \ldots, d_i$. Thus, v_i gives rise to $3d_i$ edges with negative weight. Also if $v_1v_2 \in E$ then $wt(v'_{10}v''_{20}) = wt(v'_{20}v''_{10}) = +1$. It can be seen that this construction gives a bipartite graph with $\tilde{V} = V' \cup V''$ where $v'_x \in V', v''_x \in V''$.

Thus the BMC has $2n + 6e$ vertices, and, $2e$ edges with weights $+1$, and, $6e$ edges with negative weights. Recall for any graph $\sum_i d_i = 2e$.

Let a solution be of size K, and, the partition of the vertices induced by this solution be S_1 and S_2 in the MC problem. We make the following observations.

1.1 In a solution of the BMC, the two sets $\tilde{S}_1, \tilde{S}_2$, are such that $V_{gadget_i} \subset \tilde{S}_1$ or $\tilde{S}_2$, $\forall i$. If this does not hold, that is $V_{gadget_i} \not\subset \tilde{S}_1$, then the solution can be modified that only improves the solution. In a solution to the BMC, if $v'_1v''_2$ is in the cut, so must $v''_1v'_2$ (called the *image* of $v'_1v''_2$). This follows from observation 1.1, as V_{gadget_1} and V_{gadget_2} are in the sets $\tilde{S}_1$ and $\tilde{S}_2$ respectively

[6] The argument given here is similar to the Max SNP-hardness proof of the Exclusive Binary Flip-Cut (EBFC) problem discussed in [41].

(without loss of generality). Hence, given a solution to the BMC, the solution to the corresponding MC is constructed as follows: if $v_1'v_2''$ (and its image) is in the solution to the BMC, then v_1v_2 is in the solution to the MC.

1.2 We make the following claim:

$$C^*_{MC} = C^*_{BMC}/2$$
$$\tilde{C}_{MC} \geq \tilde{C}_{BMC}/2 \tag{8.5}$$

Step 2. (BMC to KP reduction.)
Consider a BMC $((V_1, V_2), E)$, $V_1 = \{v_1^1, v_2^1, \ldots, v_m^1\}$, $V_2 = \{v_1^2, v_2^2, \ldots, v_n^2\}$. Define $\bar{i} = m + i$. Construct an instance of KP $[M_{ij}]$ with $2m$ rows and n columns as follows. If $wt(v_i^1v_j^2) = 1$, then $M_{ij} = 1, M_{\bar{i}j} = 0$. If $wt(v_i^1v_j^2) = -1$, then $M_{ij} = 0, M_{\bar{i}j} = 1$. If $v_i^1v_j^2$ is not an edge in the BMC, then $M_{ij} = M_{\bar{i}j} = 0$. See Figure 8.1 for an example. This construction ensures that molecules/rows corresponding to i and $\bar{i}$ belong to different populations (this also explains the third characteristic of the special KP problem). Also notice that, assuming that the assignment of the rows/molecules to the populations have been made, a consensus cut column is the one with the larger number of 1's (this explains the fourth characteristic of the special KP problem).

We make the following observations.

2.1 Given a a configuration (assignments of populations one/two to rows and cuts/no-cuts to columns) for the KP problem, it can be shown that we can obtain a solution for the BMC problem. We can further show that given an approximate solution for the KP problem, we can construct an approximate solution for the BMC problem and any solution is $\geq 6e$. Also the solution to the KP problem is such that the molecules/rows and sites/columns corresponding to the vertices of a gadget of a vertex from the MC belong to the same population. Thus the corresponding solution for the BMC (and then the MC) can be obtained.

2.2 We make the following claim:

$$C^*_{BMC} = C^*_{KP} - 6e$$

$$\tilde{C}_{BMC} \geq \tilde{C}_{KP} - 6e \tag{8.6}$$

Step 3. (*Gap-preserving* reduction.)
Finally, we show that the reduction is *gap-preserving*.

For some $\epsilon > 0$, let $\tilde{C}_{KP} \geq (1-\epsilon)C^*_{KP}$.

$$\begin{aligned}
\tilde{C}_{MC} &\geq \frac{\tilde{C}_{BMC}}{2} && \text{(eqn (8.5))} \\
&\geq \frac{\tilde{C}_{KP}-6e}{2} && \text{(eqn (8.6))} \\
&\geq \frac{(1-\epsilon)C^*_{KP}-6e}{2} \\
&\geq \frac{(1-\epsilon)(C^*_{BMC}+6e)-6e}{2} && \text{(eqn (8.6))} \\
&\geq \frac{(1-\epsilon)C^*_{BMC}-6e\epsilon}{2} \\
&\geq (1-\epsilon)\frac{C^*_{BMC}}{2} - 3e\epsilon \\
&\geq (1-\epsilon)C^*_{MC} - (3\epsilon)2C^*_{MC} \quad (C^*_{MC} \geq e/2) \\
&\geq (1-7\epsilon)C^*_{MC}
\end{aligned}$$

This shows that given a PTAS for KP, we can construct a PTAS for MC, which is a contradiction; hence KP does not have a PTAS. This concludes the proof of the inapproximability of the KP problem. QED

Let $1-\Upsilon$ denote the upper bound on the polynomial time approximation factor of the well-known max cut problem.

Corollary 10 *Achieving an approximation ratio* $1 - \Upsilon/7$ *for the* K*-populations problem is NP-hard.*

Theorem 10 *There does not exist a polynomial time algorithm (assuming* $P \neq NP$*) that guarantees the estimation of* $(1 - \Upsilon/7)p^k_{\max}/p^k_{\min}$ *of the total number of consensus cuts in each population* (k) *where* $p^k_{\min}$ *and* $p^k_{\max}$ *are the minimum and maximum of the digestion rates in the population* k *of the given problem.*

Proof Sketch: For convenience of notation, let us name the problem of maximizing the number of consensus cuts in each population as the $\mathrm{KP}_{\max}$ problem (where a consensus cut, j, in each population k is such that it has at least $p^k_j m_k$ cuts in the position j and m_k is the size of the population k). We will show that if we have a PTAS for $\mathrm{KP}_{\max}$, we will have a PTAS for the k-populations problem, KP, which

	v0A"	v1A"	v0B"	v1B"	v2B"	v0C"	v1C"	
v0A'	0	0	1	0	0	0	0	2
v1A'	0	0	0	0	0	0	0	2
v0B'	1	0	0	0	0	1	0	1
v1B'	0	0	0	0	0	0	0	1
v2B'	0	0	0	0	0	0	0	1
v0C'	0	0	1	0	0	0	0	2
v1C'	0	0	0	0	0	0	0	2
v0A'	0	1	0	0	0	0	0	1
v1A'	1	1	0	0	0	0	0	1
v0B'	0	0	0	1	1	0	0	2
v1B'	0	0	1	1	0	0	0	2
v2B'	0	0	1	0	1	0	0	2
v0C'	0	0	0	0	0	0	1	1
v1C'	0	0	0	0	0	1	1	1
	1	1	2	2	2	1	1	Pop

Figure 8.1: The input matrix corresponding to the bipartite graph (BMC problem) shown in Figure 1. The population (1 or 2) the molecule belongs to is shown in the rightmost column, and the bottom most row shows the consensus cut in population 1 or 2 in the optimal configuration. Also notice that the solution to the KP problem is such that the molecules/rows and sites/columns corresponding to the vertices of a *gadget* of a vertex of the graph with the MC problem, belong to the same population. The partition suggested by this solution for the MC problem (of Figure 1) is $\{B\}$ and $\{A, C\}$.

would be a contradiction. Given a KP, let $p^k_{\min} = \min_j p^k_j$, and $p^k_{\max} = \max_j p^k_j$ for each population k. Let $\tilde{X}$ denote an approximate solution and X^* denote the optimal solution. Then if $\mathrm{KP}_{\max}$ has a PTAS let $\frac{\tilde{N}_k}{N_k} \geq \epsilon$ for some $0 < \epsilon \leq 1$ where N^*_k is the number of consensus cuts in population k. Let $\tilde{C}_k \geq \tilde{N}_k p_{\min}$, then $C^*_k \leq N^*_k p_{\max}$. Hence we have

$$\frac{\tilde{C}_k}{C^*_k} \geq \frac{\tilde{N}_k p_{\min}}{N^*_k p_{\max}} \geq \epsilon \frac{p^k_{\min}}{p^k_{\max}},$$

for each population k. Summing over all the populations, we get a PTAS for the KP problem, which is a contradiction. Thus using corollary 10 we get the required result. QED

Corollary 11 *There does not exist a polynomial time algorithm (assuming $P \neq NP$) that guarantees the estimation of $(1 - \Upsilon/7)$ of the total number of consensus cuts in each population when the digestion rate at each site is the same.* QED

8.2.3 A 0.756-approximation algorithm for a 2-populations problem

We give a 0.756-approximation algorithm for a 2-populations problem [7] using the semi-definite programming based algorithm for the Max-cut problem [20], on appropriately pruned data.

Given the input, an $m \times n$ binary matrix, we assume we can trim the columns of the input using thresholds λ_1 and λ_2. Every column that has a number of 1's larger than $m\lambda_1$ is a cut in both the populations and is removed. Similarly, every column that has a number of 1's smaller than $m\lambda_2$ is not a cut in either of the populations, and is removed. After the trimming, we are left with a version of the problem where every column is a consensus cut either in population 1 or in population 2. We further assume that the digestion rate for every column is 50%.

We first show that, under these conditions, the 2-populations (2P) problem can be reduced to a *complete* BMC problem and *vice-versa*.

[7] The reduction here is similar to the reduction in Section 6.2.1.

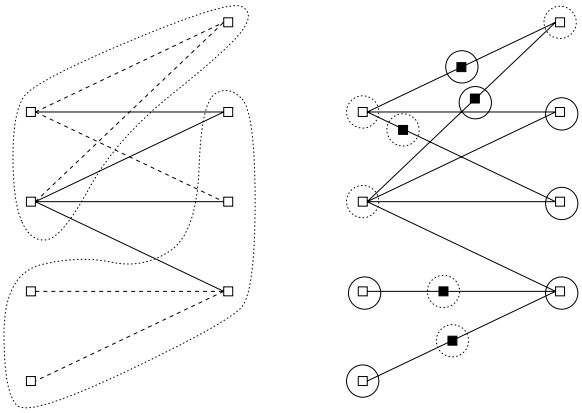


Figure 8.2: The graph for the BMC is shown on the left where an edge with negative weight is shown as a dashed line. The optimal partition of the vertices is shown in the dotted closed curves enclosing the vertices. The graph for the instance of the MC problem shown on the right is constructed from the one on the left. Every edge with weight -1 has been replaced with two edges and a vertex (shown as solid black rectangle to distinguish it from the other vertices shown as hollow rectangles). The corresponding optimal solution for the MC problem is shown by enclosing each vertex either in a solid circle or a dotted circle; the solid circle enclosed vertices belong to one partition in the solution and the dotted circle enclosed vertices belong to the other.

Given an instance of the 2-populations problem given by an $m \times n$ binary matrix $[a_{ij}]$, we construct a *complete* BMC for a bipartite graph isomorphic to $K_{m,n}$ with vertices v_1^1, v_2^1, ..., v_m^1 in the first partition and v_1^2, v_2^2, ..., v_n^2 in the second partition with the weights, $wt(\cdot,\cdot)$, defined as follows:

$$wt(v_i^1, v_j^2) = \begin{cases} 1, & \text{if } a_{ij} = 1; \\ -1, & \text{if } a_{ij} = 0. \end{cases}$$

Notice that given an instance of a complete BMC, we can similarly construct an instance of the 2-populations problem. As a result the following identity must hold:

$$a_{ij} - p_j = wt(v_i^1, v_j^2)/2,$$

since $p_j = 1/2$.

The correspondence between the solutions is as follows. Without loss of generality, let rows (molecules) 1, 2, ..., m_1 belong to the first population and the remaining rows $m_1 + 1$, $m_1 + 2$, ..., m belong to the second. Thus $X_{i1} = 1$ for $i \in [1..m_1]$ and $X_{i2} = 1$ for $i \in [m_1 + 1..m]$. Again, without loss of generality, let columns (sites) 1, 2, ..., n_1 be the consensus cut sites in the first population and the remaining columns $n_1 + 1$, $n_1 + 2$, ..., n in the second. Thus $Y_{1j} = 1$ for $j \in [1..n_1]$ and $Y_{2j} = 1$ for $j \in [n_1 + 1..n]$. Thus the cost of this partition is simply:

$$\begin{aligned} &\sum_{k=1}^{2} \sum_{j=1}^{n} \sum_{i=1}^{m} X_{ik} Y_{kj} [a_{ij} - p_j] \\ &= \frac{1}{2} \left(\sum_{j=1}^{n_1} \sum_{i=1}^{m_1} wt(v_i^1, v_j^2) + \sum_{j=n_1+1}^{n} \sum_{i=m_1+1}^{m} wt(v_i^1, v_j^2) \right). \end{aligned}$$

The corresponding partition of the BMC problem is then as follows: the vertices of the first partition are $\{v_1^1, v_2^1, \ldots, v_{m_1}^1, v_{n_1+1}^2, v_{n_1+2}^2, \ldots, v_n^2\}$, and the vertices in the second partition are $\{v_{m_1+1}^1, v_{m_1+2}^1, \ldots, v_m^1, v_1^2, v_2^2, \ldots, v_{n_1}^2\}$. Thus, it immediately follows that 2P has a solution of size x iff BMC has a solution of size $2x$. Notice that

$$C_{BMC} = p_1 - l_1, \tag{8.7}$$

where p_1 is the number of edges with weight 1 and l_1 is the number of edges with weight -1 in the cut. Let l_2 be the remaining number of edges with weight -1

and L be the total number of edges with weight -1. Thus

$$L = l_1 + l_2. \tag{8.8}$$

Instead of counting only the 1's in the consensus cut columns, let the cost function measure all the "correct decisions" in the configuration. Notice that the optimal value is obtained at the same configuration for both the cost functions. Thus, let the cost function of the 2P problem be the number of 1's in the consensus cut columns and the number of -1's in the columns that are not consensus cuts in each of the two populations $(p_1 + l_2)$. Thus using equations (8.7) and (8.8), we have

$$C_{2P} = L + \frac{C_{BMC}}{2}. \tag{8.9}$$

Given a BMC, we construct an instance of the MC problem (with weight on the edges as 1) by replacing every edge with a negative weight by two edges and a vertex, each edge having a weight of 1. See Figure 8.2 for an illustration. If L is the number of edges with weight -1, then the MC instance has $m + n/2 + L$ vertices.

Now, we give the correspondence between the solutions in each of the problem. Notice that the edges introduced in the reduction come in pairs. Let the solution to the MC problem include l_1 edges which are *not* paired, $2l_2$ paired edges and p of the original edges (which had a weight of 1 in the BMC problem). Then

$$C_{MC} = p + l_1 + 2l_2, \tag{8.10}$$

and the cost of the BMC problem by the construction is,

$$\frac{C_{BMC}}{2} = p - l_1. \tag{8.11}$$

Since $l_1 + l_2 = L$, we have from equations (8.10) and (8.11),

$$\frac{C_{BMC}}{2} = C_{MC} - 2L.$$

From equation (8.9) we have

$$C_{2P} = C_{MC} - L \tag{8.12}$$

Finally, we use the algorithm presented in [20] to obtain a an algorithm for the MC problem with an approximation factor of 0.878. Let $\tilde{C}_X$ denote an approximate solution and C^*_X denote the optimal solution to problem X.

$$\begin{aligned}
\tilde{C}_{2P} &= \tilde{C}_{MC} - L && \text{(from eqn (8.12))} \\
&\geq 0.878 C^*_{MC} - L && \text{(from [20])} \\
&\geq 0.878(C^*_{2P} + L) - L && \text{(from eqn (8.12))} \\
&= 0.878 C^*_{2P} - 0.122L \\
&\geq 0.756 C^*_{2P} && \text{(from eqn (8.9))}
\end{aligned}$$

This concludes the argument.

8.2.4 An algorithm for the K-populations problem

Here we present a set of heuristics to detect the different populations in a sample of molecules. In our experiments, we have modeled the false positive, false negative, orientation and sizing errors in the input. Let

p be the digestion rate (true positive),

q be the false positive rate,

m be the sample size, and

$m_{\min}$ be the smallest sample size for any population.

Thus

$$m_{\min} K \leq m.$$

The population detection is carried out in three major steps: First we detect the physical map which is the union of the map of all the populations. Next, we merge identical consensus cuts (characterized by their presence or absence in a population, defined by the *population map*). At this step we also carry out a certain amount of error correction which makes the next step more robust. In the final step, we use the merged consensus cut columns of the last step to separate the populations. We give the details of each step below.

(Step 1) Common physical map detection. Obtain the physical map, which is the union of the physical maps of all the populations. This step eliminates the orientation, sizing and false positive errors. Here we must assume that the false

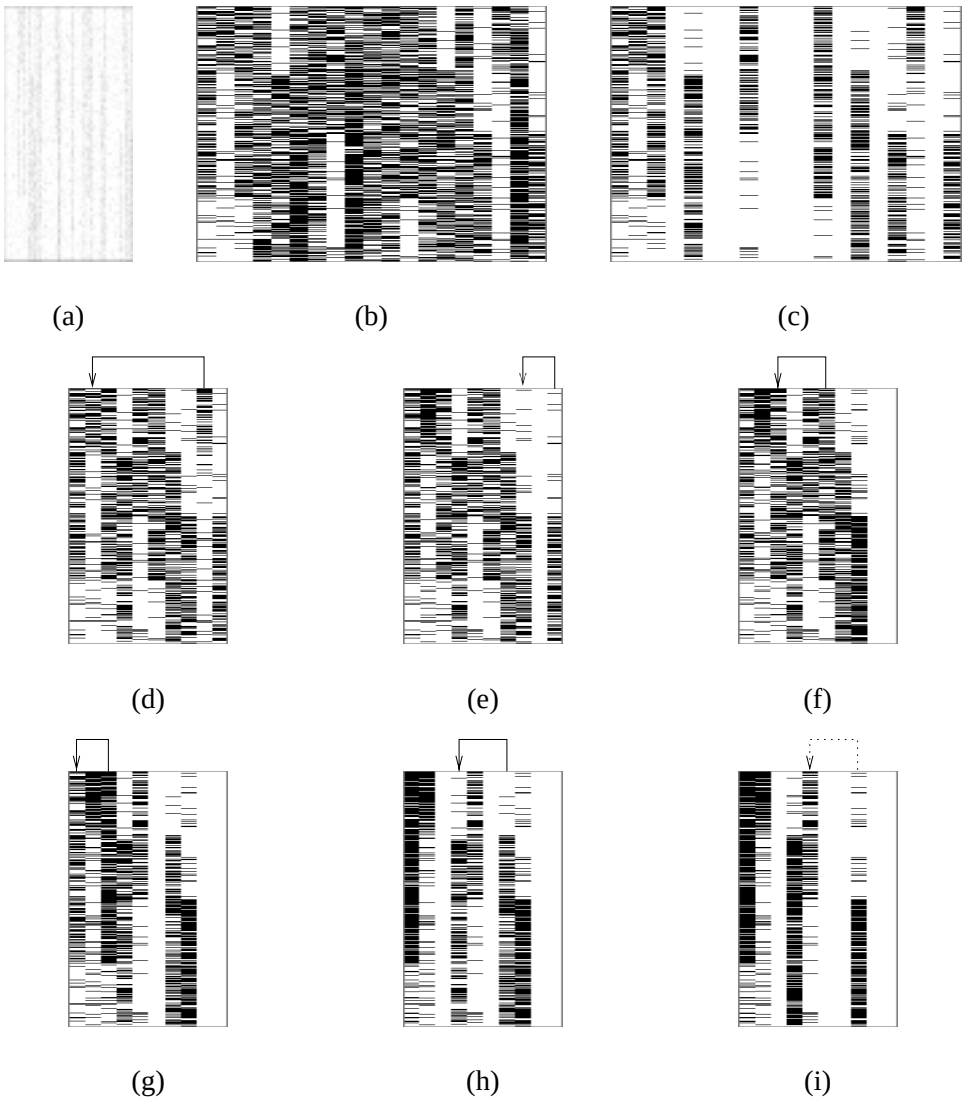


Figure 8.3: Illustration of step 2 (merging and error correction): (a) The correctly aligned molecules with the computed consensus cuts. In (b) to (i), each cut is shown by a small horizontal bar. (b) This shows only the consensus cuts in the population aligned in (a). (c) For the very first merging, all those cuts that appear in *all* the populations have been filtered. (d) to (i) show each step of the merge; the polyline on top denotes the two columns with identical *population map* being merged. Notice the "darkening" of the merged column due to the error correction. In (i) the two columns to be merged actually have population maps that are complements of each other.

positive rate is low enough not to be confused as a true positive of a "small" population; thus the following must hold:

$$mq \ll m_{\min}p \quad \Rightarrow \quad Kq \leq \frac{m}{m_{\min}}q \ll p. \tag{8.13}$$

In our experiments, we use the EBFC-based algorithm (Exclusive Binary Flip Cut) presented in [38].

(Step 2) Merging consensus cuts and error correction. Given a consensus cut j, define a *population map* for j as follows: let the number of populations be K, then the population map is a K length vector of 0's and 1's, where a 1 at position i denotes j is a consensus cut in population i and a 0 denotes j is *not* a consensus cut in population i.

Let $j_{pm_1}, j_{pm_2}, \ldots, j_{pm_l}$ be l cuts that have the same population map. We replace all these l consensus cut columns by a single *representative* column. In addition, we also carry out false negative error correction in the merged (representative) column as described below.

(Step 2.1) Identifying the consensus cuts for merging. Recall that at this stage of the algorithm we do not know what the populations are (nor how many of them). We will identify the consensus cuts with identical population maps by simultaneously looking at c consensus cuts for the presence of at least t pairwise true positives. The reader may verify that in the worst case,

$$c \geq \lceil \tfrac{1}{p} \rceil \quad \text{and} \quad t = m_{\min}p.$$

Thus for $p \geq 0.5$, it suffices to consider *pairwise* cuts (i.e., $c = 2$). The merging is carried out iteratively until no more columns can be merged. A weight wt is associated with every merged column which is the number of consensus cut columns that were merged.

(Step 2.2) Error (false negative) correction. We give the details for the case where $c = 2$, the other cases are similar. Once we recognize that two consensus cuts j_1 and j_2 have the same population map, it is easy to make the error correction: if j_1 and j_2 do not agree in molecule i, we place a consensus cut for molecule i in the merged column. Assuming that the population map of j_1 and j_2 is correct, this

consensus cut in molecule i is correct with probability p and wrong with probability q (recall that $q << p$). See Figure 8.3 for an example.

Routine Separate-Populations

Set $Tol = pm_{\min}$.
Initialize every molecule, i, to be in population p_0.
For each column, j, in the descending order of wt do {
  For each population, p, with size $> Tol$, do {
    Compute N_0^{jp}, the number of molecules with 0 at j.
    Compute N_1^{jp}, the number of molecules with 1 at j.
    If ($N_0^{jp} > Tol$ and $N_1^{jp} > Tol$) {
      Split the population by the 0's and 1's.
      Renumber the new populations as p' and p''.
    }
  }
}

A *similarity* tree.

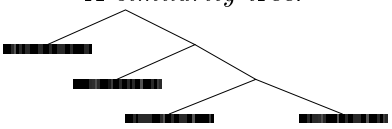


Figure 8.4: The algorithm to separate the populations using the representative columns, and an example of a tree based on the decisions taken at the "if" statement. The physical map of each population is at the leaf node of this tree.

(Step 3) Separating the populations. Let the number of representative columns be n_r. The reader may verify that

$$\log K \leq n_r \leq n.$$

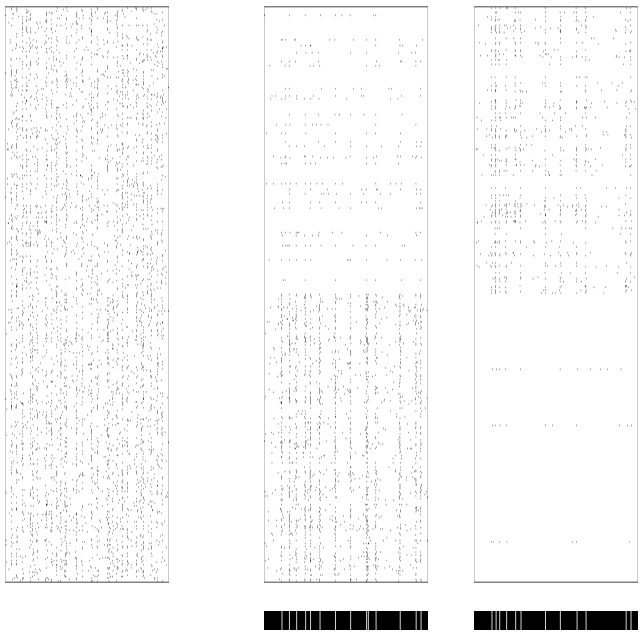
The pseudocode for the algorithm to detect the different populations is presented in Figure 8.4. This algorithm splits the molecules into different populations and also gives a tree, based on the iterative splitting of the sample, to give rise to a *similarity tree*. The common structure of the maps of the different populations is captured in this tree.

8.2.5 Experimental results

We carried out several experiments using simulated data on upto six populations data. We present the results of a two, a four and a six population sample in Figures 8.5 through 8.8. We observed in our experiments that as the number of populations increases, we need to increase the digestion rate to detect the different populations. In the figure each molecule is represented by a row with a black dot denoting a cut in the molecule. For ease of visualization of the output, in the synthesized data, the molecules from each population are placed close to one another (the algorithm neither knows nor uses this information). As can be seen, the algorithm separates the populations with about 20% error (picking a wrong molecule or missing out a molecule), but gathers sufficient information to infer the right ordered restriction map.

8.2.6 Summary

We have presented our initial complexity analysis and some heuristics for identifying K ($K \geq 2$) distinct populations from a set of corrupted ordered restriction map data. Our main results are the following: (1) The problem (even under a forgiving idealized model) is computationally infeasible. (2) We give a guaranteed 0.756-approximation algorithm for a special class of the 2-populations problem. (3) However, for the general K-populations problem, when certain assumptions are made regarding the errors in the input data (e.g., false positive error probability is negligible compared to false negative error probability), we have been able to empirically obtain very promising results. It remains open whether using better prior models of the data as well as variations on the heuristics presented here we can devise algorithms with better performance for this problem while accounting for many other error sources.



$2-populations\ data.\quad Population\ 1.\quad Population\ 2.$

Figure 8.5: An example of a 2-populations problem. The false negative rate is 50%; the false positive rate is 5%; the orientation is incorrect 50% of the time; small sizing error; the number of consensus cuts is around 15 for each population and the the maps of each population are very similar. The second row shows the aligned molecules of each population (as detected by the algorithm) and the computed physical maps. The algorithm makes an error of about 17% in assigning population to each molecule, but picks up the right map for each population as shown.

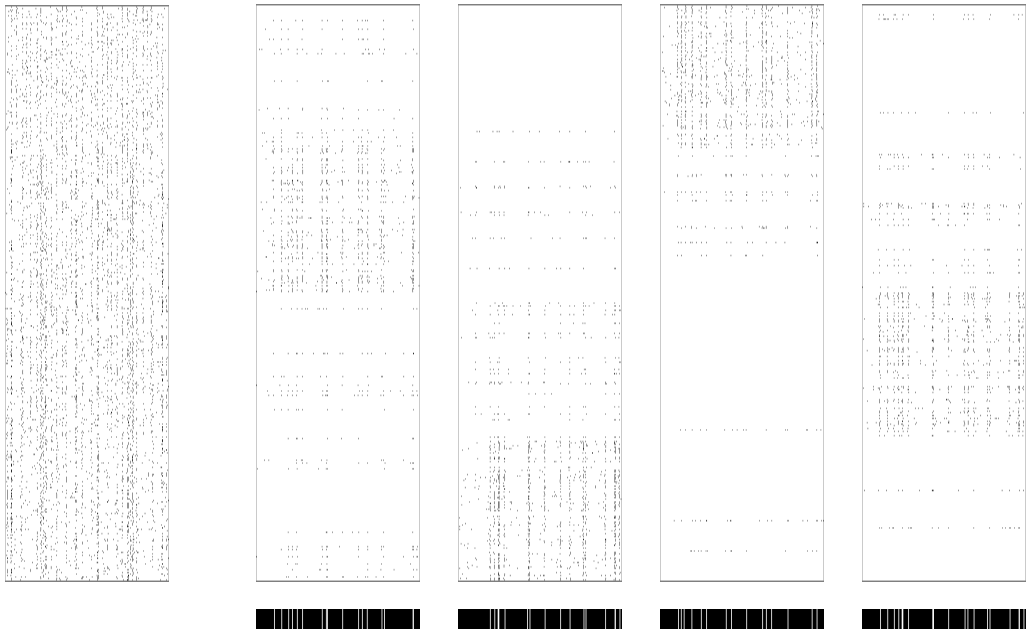
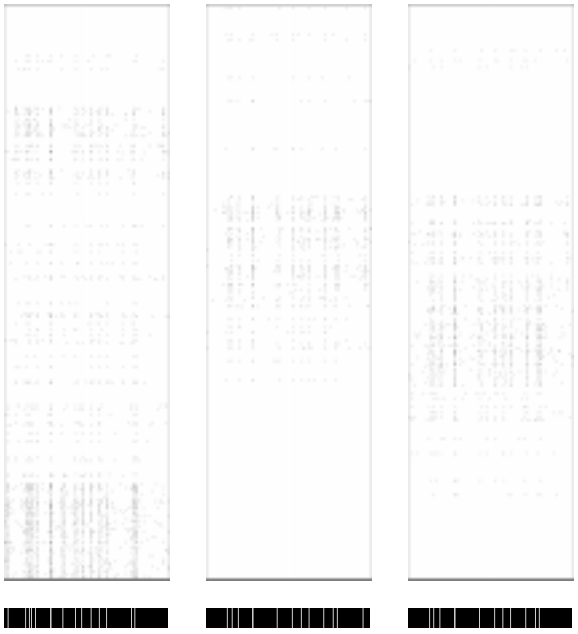


Figure 8.6: An example of a 4-populations problem. The false negative rate is 50%; the false positive rate is 5%; the orientation is incorrect 50% of the time; small sizing error; the number of consensus cuts is around 10-15 for each population and the the maps of each population are very similar. The algorithm makes an error of about 20% in assigning population to each molecule, but picks up the right map for each population as shown.

Figure 8.7: An example of a 6-populations problem. The false negative rate is 30%; the false positive rate is 5%; the orientation is incorrect 50% of the time; small sizing error; the number of consensus cuts is around 10-15 for each population and the the maps of each population are very similar. The algorithm makes an error of about 20% in assigning population to each molecule, but picks up the right map for each population as shown. See Figure 8.8 for the remaining three populations.



Population 4 *Population* 5 *Population* 6

Figure 8.8: An example of a 6-populations problem. See Figure 8.7 for the remaining three populations and the data.

Part II

Sequence Analysis

Chapter 9

Pattern Discovery

9.1 Introduction

Given an input sequence of data, a motif is a repeating pattern (possibly interspersed with dont-care characters) that occurs in the sequence. The number of motifs could potentially be exponential in the size (number of characters) of the input (see Examples 1 and 2 in the following section). Typically, the higher the self similarity in the sequence, greater is the number of motifs in the data. Motif discovery on such data, such as repeating DNA or protein sequences, is a source of concern since these exhibit a very high degree of self-similarity (repeating patterns). Usually, this problem of an exploding number of motifs is tackled by pre-processing the input, using heuristics, to remove the repeating or self-similar portions of the input. Another way of trimming down the number of motifs is to use a "statistical significance" measure. However, due to the absence of a good understanding of the domain, there is no consensus over the right model to use. Thus there is a trend towards model-less motif discovery in different fields. To keep the problem manageable, it is useful to identify a small number of motifs that capture important information about the family of motifs.

In this chapter, we show that, for any sequence, there exists only a polynomial number of special motifs and every other motif can be simply *generated* from them. We name these special motifs *irredundant* motifs. The result is meaningful

also from an algorithmic viewpoint, since the ideas from the proof can be used to design polynomial time algorithms to detect these irredundant motifs. This bound on the number of useful motifs gives validation to motif-based approaches, since the total number of irredundant motifs does not explode. This result is of significance to most applications that use pattern discovery as the basic engine such as data mining, clustering and matching. This family of irredundant motifs is also very characteristic of the family of all the motifs: in applications such as multiple sequence alignment, we have shown that the irredundant motifs suffice to obtain the alignment [44]. However, in applications that use the motifs to extract signature motifs of sets of sequences, all the motifs, including the redundant ones, may be of relevance.

Roadmap. In Section 9.2 we define motifs and some basic related concepts (such as maximality) and in Section 9.3 we introduce the notion of irredundancy and show that the number of such motifs is only quadratic in the input length.

9.2 Basic Concepts

Let s be a sequence on an alphabet Σ, '.' $\notin \Sigma$. A character from Σ, say σ, is called a solid character and '.' is called a "dont care" character. For brevity of notation, if x is a sequence, then $|x|$ denotes the length of the sequence and if x is a set of elements then $|x|$ denotes the cardinality of the set. Thus the size of the sequence s is denoted by $|s|$ and the j^{th} $(0 \leq j \leq |s|-1)$ character of the sequence is given by $s[j]$.

Definition 1 *($\sigma_1 <, =, \leq \sigma_2$) If σ_1 is a "dont care" character and $\sigma_2 \in \Sigma$, then $\sigma_1 < \sigma_2$. If both σ_1 and σ_2 are identical characters in Σ, then $\sigma_1 = \sigma_2$.*
If either $\sigma_1 < \sigma_2$ or $\sigma_1 = \sigma_2$ holds, then $\sigma_1 \leq \sigma_2$.

Definition 2 *(p occurs at l, cover) A string, p, on $\Sigma \cup$ '.', occurs at position l in s if $p[j] \leq s[l+j]$ holds for $0 \leq j \leq |p|-1$. p is said to cover the interval $[l, l+|p|-1]$ on s.*

Definition 3 *(k-motif m, location list $\mathcal{L}_m$) Given a string s on alphabet Σ and an integer k, $2 \leq k < |s|$, a string m on $\Sigma \cup$ '.' is a k-motif with location list $\mathcal{L}_m = (l_1, l_2, \ldots, l_p)$, if all of the following hold:*

1. $m[0], m[|m|-1] \in \Sigma$.
   (The first and last characters of the motif are solid characters; if "dont care" characters are allowed at the ends, the motifs can be made arbitrarily long in size without conveying any extra information.)

2. $p \geq k$.

3. *there does not exist a location l, $l \neq l_i$, $1 \leq i \leq p$ such that m occurs at l on s (the location list is of maximal size).*
   (This ensures that any two distinct location lists must correspond to distinct motifs.)

4. *for every "dont care" character at position j in m, there exist at least two distinct occurrences l_{i_1} and l_{i_2}, $1 \leq i_1, i_2 \leq p$, such that $s[l_{i_1} + j] \neq s[l_{i_2} + j]$.*
   (No "dont care" character can be replaced by a solid character, otherwise an arbitrary number of "dont care" characters could be introduced without conveying any extra information.)

If m is a string on Σ, m is called a *simple motif*. If m is a string of $\Sigma \cup \{\text{'.'}\}$, m is called a *rigid motif*. In the rest of the discussion a k-motif is referred to as a motif.

Consider $s = ABCDABCD$. Using the definition of motifs, the different motifs are as follows:

1. $m_1 = AB$ with $\mathcal{L}_{m_1} = \{0, 4\}$,
2. $m_2 = BC$ with $\mathcal{L}_{m_2} = \{1, 5\}$,
3. $m_3 = CD$ with $\mathcal{L}_{m_3} = \{2, 6\}$,
4. $m_4 = ABC$ with $\mathcal{L}_{m_4} = \{0, 4\}$,
5. $m_5 = BCD$ with $\mathcal{L}_{m_5} = \{1, 5\}$ and

6. $m_6 = ABCD$ with $\mathcal{L}_{m_6} = \{0, 4\}$.

Notice that $\mathcal{L}_{m_1} = \mathcal{L}_{m_4} = \mathcal{L}_{m_6}$ and $\mathcal{L}_{m_2} = \mathcal{L}_{m_5}$. Using the notation $\mathcal{L} + i = \{x + i | x \in \mathcal{L}\}$, $\mathcal{L}_{m_5} = \mathcal{L}_{m_6} + 1$ and $\mathcal{L}_{m_3} = \mathcal{L}_{m_6} + 2$ hold. We call the motif m_6 *maximal* as $|m_6| > |m_1|, |m_4|$ and $|m_5| > |m_2|$. Motifs m_1, m_2, m_3, m_4 and m_5 are non-maximal motifs.

We give the formal definition of maximality below.

Definition 4 *(Maximal Motif) Let p_1, p_2, ..., p_k be all the motifs in the sequence s. A motif p_i is maximal if and only if there is no p_j $(j \neq i)$ such that p_i is a substring of p_j, or, if p_i is a substring of p_j, then there exists at least one occurrence of p_i in s that is not covered by p_j in s.*

However, the notion of maximality alone does not suffice to bound the number of motifs. We now motivate the need for defining irredundant maximal motifs by giving two simple examples of strings that have an unusually large number of maximal motifs without conveying extra information about the input.

Example 1 Let the input string s have the following form:

$$ac_1c_2c_3baXc_2c_3bYac_1Xc_3bYYac_1c_2Xb$$

Then the maximal motifs (which are $2^{\Omega\sqrt{n}}$ in number) are as follows.

motif	location list			
	$[ac_1c_2c_3b$	aXc_2c_3bY	ac_1Xc_3bYY	$ac_1c_2Xb]$
$a...b$	+	+	+	+
$a..c_3b$	+	+	+	
$a.c_2.b$	+	+		+
$ac_1..b$	+		+	+
$\mathbf{a.c_2c_3b}$	+	+		
$\mathbf{ac_1.c_3b}$	+		+	
$\mathbf{ac_1.c_3b}$	+			+

Example 2 Let $s = aaaaaaaaaaa$ and $k = 2$. By the definition, the motifs with the location lists shown as positions on the input string are as follows.

size	motif	location list										
		[a	a	a	a	a	a	a	a	a	a	a]
2	aa	+	+	+	+	+	+	+	+	+	+	
3	aaa	+	+	+	+	+	+	+	+	+		
4	aaaa	+	+	+	+	+	+	+	+			
5	aaaaa	+	+	+	+	+	+	+				
6	aaaaaa	+	+	+	+	+	+					
7	aaaaaaa	+	+	+	+	+						
8	aaaaaaaa	+	+	+	+							
9	aaaaaaaaa	+	+	+								
10	aaaaaaaaaa	+	+									

In other words, a sequence of n identical characters has $n-2$ maximal motifs.

Consider a minor variation of the original string $s' = aaaaaXaaaaa$. Note that the number of motifs increase drastically. The motifs, in increasing order of size, along with the locations list are as follows. We also give the non-maximal motifs (for instance all the motifs of size 6) for the sake of completeness.

Motif size = 2

	[a	a	a	a	a	X	a	a	a	a	a]
aa	+	+	+	+			+	+	+	+	

Motif size = 3

	[a	a	a	a	a	X	a	a	a	a	a]
aaa	+	+	+				+	+	+		
a.a	+	+	+		+		+	+	+		

Motif size = 4

	[a	a	a	a	a	X	a	a	a	a	a]
aaaa	+	+					+	+			
a.aa	+	+			+		+	+			
aa.a	+	+		+			+	+			
a..a	+	+		+	+		+	+			

Motif size = 5

	[a	a	a	a	a	X	a	a	a	a	a]
aaaaa	+						+				
a.aaa	+				+		+				
aa.aa	+			+			+				
aaa.a	+		+				+				
a..aa	+			+	+		+				
a.a.a	+		+		+		+				
aa..a	+		+	+			+				
a...a	+		+	+	+		+				

Motif size = 6

	[a	a	a	a	a	X	a	a	a	a	a]
a..aaa				+	+						
aa..aa			+	+							
aaa..a		+	+								
a.aa.a		+			+						
a.a.aa			+		+						
aa.a.a		+		+							
aa...a		+	+	+							
a...aa			+	+	+						
a.a..a		+	+		+						
a..a.a		+		+	+						
a....a		+	+	+	+						

Motif size = 7

	[a	a	a	a	a	X	a	a	a	a	a]
a..aaaa				+	+						
a.a.aaa			+		+						
a.aa.aa		+			+						
a.aaa.a	+				+						
aa..aaa			+	+							
aa.a.aa		+		+							
aa.aa.a	+		+								
aaaa.aa		+	+								
aaa.a.a	+		+								
aaaa..a	+	+									
a...aaa			+	+	+						
a.a..aa		+	+		+						
a.aa..a	+	+			+						
a.a.a.a	+		+		+						
a..a.aa		+		+	+						
a..aa.a	+			+	+						
a....aa		+	+	+	+						
aa....a	+	+	+	+							
a.....a	+	+	+	+	+						

Motif size = 8

	[a	a	a	a	a	X	a	a	a	a	a]
aa..aaaa			+	+							
aa.aa.aa	+			+							
aaa.a.aa	+		+								
aaaa..aa	+	+									
aaa..aaa		+	+								
aa.a.aaa		+		+							
aa.a..aa	+	+		+							
aa..a.aa	+		+	+							
aaa...aa	+	+	+								
aa...aaa		+	+	+							
aa....aa	+	+	+	+							

Motif size = 9

	[a	a	a	a	a	X	a	a	a	a	a]
aaa.a.aaa	+		+								
aaaa..aaa	+	+									
aaa..aaaa		+	+								
aaa...aaa	+	+	+								

Motif size = 10

	[a	a	a	a	a	X	a	a	a	a	a]
aaaa..aaaa	+	+									

9.3 Notion of Redundancy

We saw in the last section an example where a small change in the input string (replacing just one character by another) increases the number of maximal motifs from linear to exponential. This suggests that using a notion stronger than maximality will be useful. We introduce such a notion below.

Definition 5 *($m_1 \leq m_2$) Given two motifs m_1 and m_2 with $|m_1| \leq |m_2|$, $m_1 \leq m_2$ holds if $m_1[j] \leq m_2[j]$, $0 \leq j < |m_1|$.*

For example, let $m_1 = AB..E$, $m_2 = AK..E$ and $m_3 = ABC.E.G$. Then $m_1 \leq m_3$, and $m_2 \not\leq m_3$.

Definition 6 *(Redundant motif) A maximal motif m, with location list $\mathcal{L}_m$, is redundant if there exist maximal motifs m_i, $1 \leq i \leq p$, such that $\mathcal{L}_m = \mathcal{L}_{m_1} \cup \mathcal{L}_{m_2} \ldots \cup \mathcal{L}_{m_p}$, (i.e., every occurrence of m on s is already covered by one of the motifs $m_1, m_2, \ldots, m_p$).*
If $|m| = |m_i|$, $\forall i$, motif m is called fixed-size *redundant* [1].
A motif that is not redundant is called an irredundant *motif.*

9.3.1 Generating operations

The redundant motifs need to be generated from the irredundant ones, if required. We define the following generating operations. Let m, m_1 and m_2 be motifs.

1. Unary yank operator, $Y^{\delta}(m)$, $0 < \delta < |m|$.
   This is a valid operation when δ is an integer and $m[\delta]$ is a solid character, since all the operations are closed under motifs. $Y^{\delta}(m)$ is the substring given by $m[0 \ldots \delta]$.

   For example, if $m = AB..CDE$, then $Y^2(m)$ is not a valid operation since $m[2]$ is a dot-character. Also, $Y^4(m) = AB..C$.

2. The following binary operation is valid only if

   - $|m_1| = |m_2|$ and
   - For each i, $0 \leq i \leq |m_1| - 1$, $m_1[i] \leq m_2[i]$ or $m_2[i] \leq m_1[i]$.

   Binary plus operator, $m_1 \bigoplus m_2$.
   $m = m_1 \bigoplus m_2$, where m is such that every solid-character in m is a solid-character in both m_1 and m_2, otherwise it is a dot-character. For example if $m_1 = A..D..G$ and $m_2 = AB...FG$. Then, $m = m_1 \bigoplus m_2 = A.....G$.

The operations satisfy the following properties.

1. If $m = m_1 \bigoplus m_2$, then $m \leq m_1$ and $m \leq m_2$.

2. The operation is symmetric, i.e., $m_1 \bigoplus m_2 = m_2 \bigoplus m_1$.

[1] This notion of redundancy has also been used by the authors in [16].

3. The operation is associative, i.e., $m_1 \bigoplus (m_2 \bigoplus m_3) = (m_1 \bigoplus m_2) \bigoplus m_3$.

4. The Y^δ operator distributes over $\bigoplus$, i.e., $Y^\delta(m_1 \bigoplus m_2) = Y^\delta(m_1) \bigoplus Y^\delta(m_2)$.

These properties are straightforward to verify.

9.3.2 Bounding the number of irredundant motifs

Definition 7 *(Basis) Given a sequence s on an alphabet Σ, let $\mathcal{M}$ be the set of all maximal motifs on s. A set of motifs $\mathcal{B}$ is called the basis of $\mathcal{M}$ iff the following hold:*

1. *for each $m \in \mathcal{K}$, m is irredundant with respect to $\mathcal{K} - \{m\}$, and,*

2. *every motif $m \in \mathcal{M}$, and no more, can be generated from $\mathcal{K}$.*

In general, $|\mathcal{M}| = \Omega(2^n)$. The natural attempt now is to obtain as small a basis as possible. In the following theorem we show that using the notion of fixed-size irredundancy there can be no more than n^2 irredundant motifs where n is the size of the input.

Theorem 11 *Let s be a string with $n = |s|$ and let $\mathcal{B}^f$ be a basis using the fixed-size redundancy and let $\mathcal{B}$ be a basis using the general redundancy of Definition 6. Then $|\mathcal{B}^f|, |\mathcal{B}| < n^2$.*

Proof Sketch. From the definition of irredundancy, it follows that if j motifs are incident on the same location on the input string, then there must exist j distinct locations with at least one of the j motifs incident on each position. This holds both for fixed-size redundancy and the general notion of redundancy. Hence, no location can have more than n motifs incident on it. Thus, it follows that the number of these motifs can be no more than n^2. Hence $|\mathcal{B}^f|, |\mathcal{B}| < n^2$. QED

Back to Examples 1 and 2. Consider Example 1 discussed in the last section. The motifs shown in bold in the example are mutually irredundant. Each of the redundant motif can be constructed from the motifs in the these using the generating operations. For example, $a..c_3b = a.c_2c_3b \bigoplus ac_1.c_3b$ and $\mathcal{L}_{a..c_3b} = \mathcal{L}_{a.c_2c_3b} \cup \mathcal{L}_{ac_1.c_3b}$.

Consider Example 2. For a fixed size of the motif f, $2 \leq f \leq 10$, in this example, the motifs shown in italics are redundant with respect to the other motifs of the same size f. The reader may check the location lists of these motifs to verify the redundancy of the motifs. The basis with respect to fixed-size redundancy consists of all the motifs shown in bold which is $O(n^2)$ in number. However, a basis using the general notion of redundancy for the input string is shown below. To compare it with the basis for the original string s, we reproduce the basis for s as well.

s' = aaaaaXaaaaa

size	motif	[a	a	a	a	a	X	a	a	a	a	a]
2	aa	+	+	+	+			+	+	+	+	
3	aaa	+	+	+				+	+	+		
4	aaaa	+	+					+	+			
5	aaaaa	+						+				
7	a.aaa.a	+				+						
8	aa.aa.aa	+			+							
9	aaa.a.aaa	+		+								
10	aaaa..aaaa	+	+									

s = aaaaaaaaaaa

size	motif	[a	a	a	a	a	a	a	a	a	a	a]
2	aa	+	+	+	+	+	+	+	+	+	+	
3	aaa	+	+	+	+	+	+	+	+	+		
4	aaaa	+	+	+	+	+	+	+	+			
5	aaaaa	+	+	+	+	+	+	+				
6	aaaaaa	+	+	+	+	+	+					
7	aaaaaaa	+	+	+	+	+						
8	aaaaaaaa	+	+	+	+							
9	aaaaaaaaa	+	+	+								
10	aaaaaaaaaa	+	+									

Notice the similarity in the two bases (for s and s'). Irredundant motif of size 6 is missing in the basis for s'. The striking similarity suggests that the general notion

of redundancy is perhaps a more natural notion.

Further, every redundant maximal motif of s' can be obtained from its basis using the generating operations. We give some examples for illustration.

1. $a..aa = aaaaa \bigoplus Y^4(a.aaa.a) \bigoplus Y^4(aa.aa.aa)$ and
   $\mathcal{L}_{a..aa} = \mathcal{L}_{aaaaa} \cup \mathcal{L}_{a.aaa.a} \cup \mathcal{L}_{aa.aa.aa}$.
2. $aaa...aa = Y^7(aaaa..aaaa) \bigoplus Y^7(aaa.a.aaa)$ and
   $\mathcal{L}_{aaa...aa} = \mathcal{L}_{aaaa..aaaa} \cup \mathcal{L}_{aaa.a.aaa}$.

QED

9.3.3 Detecting irredundant motifs

The next natural question is whether the irredundant motifs can be detected in polynomial time. This is true, since in an iterative scheme, which produces motifs in an order of increasing size f, the following two (polynomial time) tests will eliminate the redundant motifs:

1. If *every* occurrence of a motif till this iteration step has been appended with another motif or a character.
2. If a motif of size f is fixed-size redundant.

Chapter 10

Multiple Sequence Alignment [1]

Given a set of N sequences, the Multiple Sequence Alignment problem is to align these N sequences, possibly with gaps, that brings out the best *commonality* of the N sequences. Various alignment cost functions [2, 10, 9, 22, 25, 24, 59, 60], have been used in literature. The general approach to solving the pairwise ($N = 2$) sequence alignment problem has been a dynamic programming technique using different mechanisms of scores which is a function of the *edit distance*, along with *gap penalties*, to evaluate the similarity of the sequences. In [61, 57] the case of $N > 2$ has been handled by first doing a pairwise alignment for some or all possible pairs in some order and then building a N-wise alignment from these.

MUSCA[2] uses a two-stage approach to the alignment problem by identifying two relatively simpler sub-problems which deal separately with the two issues, one of identifying the "local similarities" and the other of aligning the similarities appropriately [44]. We first *discover motifs* in the N sequences, and then use these motifs to obtain a "good" alignment. Informally, a motif is a repeated pattern that appears more than once in a sequence. In the alignment context a motif is a pattern that appears in two or more input sequences (See the previous chapter for a formal definition). A major point of criticism regarding using motifs is that they

[1] This chapter also appears as "MUSCA: An Algorithm for Constrained Alignment of Multiple Data Sequences ", coauthored with Aris Floratos and Isidore Rigoutsos, submitted for publication [44].

[2] Musca is a constellation in the polar region of the Southern Hemisphere near Apus and Carina. Also, MUSCA is an anagram of the salient characters in Constrained Multiple Sequence Alignment.

are usually very large in number (exponential in input size); however, we show that the number of motifs relevant to the alignment problem, termed *irredundant motifs*, is polynomial in the input size. Moreover, in practice, this number is much smaller (sub-linear). Thus, using motifs for the alignment helps in at least two ways: (1) it aids in a direct N-wise alignment, as opposed to composing the alignments from lower order (say pairwise) alignments and (2) the solution is independent of the order of the input sequences. We believe that, in practice, these have important consequences.

The second sub-problem of the alignment problem is that of obtaining a good alignment. Notice that any arbitrary set of motifs need not necessarily give rise to an alignment, under the premise that the alignment that uses a motif *does not introduce gaps in the motif*. Having obtained *all possible* motifs in the first stage, this stage involves pruning this set to obtain a (sub)set that gives an alignment. We solve this problem by mapping the motifs of the first stage to a suitable directed graph. Next we show that obtaining an alignment of the motifs is equivalent to solving a set-covering problem. Here it is worth comparing this with other graph-theoretic approaches in literature (although this is only one of the stages of our approach): firstly our model is much simpler to compute since a significant hard part of the problem is tackled in the first stage of motif discovery; secondly the size of our graph is much smaller (the number of vertices for m sequences of size n each is $O(mn)$, in *most cases*, whereas it is $O(n)$ in our approach) [61, 57, 30, 50].

It is well known that the multiple sequence alignment problem, in addition to being a hard-to-solve problem, is also very hard to *model* to the satisfaction of evolutionary biologists, geneticists and other users. Does our approach have any theoretical contributions to the multiple sequence alignment problem in general? An interesting fallout of our approach is the identification of a user-controlled parameter, the notion of an *alignment number* K $(2 \leq K \leq N)$: this additional requirement constrains the alignment to have at least k sequences agree on a character, whenever possible, in the alignment. This is particularly of interest when a large number of sequences are being aligned. The utility of the alignment number is corroborated by the users who view this as a natural constraint while dealing

with a large number of sequences.

We also identify two cost functions that could be used to evaluate the "goodness" of an alignment while dealing with multiple sequences and give a complete complexity anlysis for these two problems.

Roadmap. We describe our approach of aligning sequences in Section 10.1. We discuss the issues involved in using motifs for alignment and present a simple graph theoretic formulation in Section 10.1.1. We identify the underlying optimization problem, implicit in our approach, in Section 10.1.2 and give a complexity analysis for the same. Having shown the hardness of the optimization problem, we present an approximate solution to this problem by mapping it to a set covering problem in Section 10.1.3. In Section 10.2 we show results of our algorithm on protein sequences.

10.1 Sequence Alignment

In the discussion in the last chapter we have dealt with the first stage of our approach, i.e., discovering motifs from input sequences. Similar definitions extend to discovering common motifs from multiple sequences. TEIRESIAS is an efficient implementation of the motif discovery problem [51] and we use this in our experiments.

We obtain the irredundant motifs and the position of the motif in each sequence it appears in. The offset list, associated with each motif p_i, is a list of two-tuples (s_i, l_i), where s_i is a pointer to the sequence and l_i is the offset in the sequence. We assume that every offset list has exactly one occurrence of each sequence. At the end of the motif discovery step we have motifs, $p_1, p_2, \ldots, p_N$; N is the number of the motifs (not necessarily distinct), each with an offset list consisting of at least $k \geq 2$ (distinct) sequences and the position in the sequence where the motif appears.

Can *all* the motifs be used in an alignment? If the answer is yes, we form an alignment that respects all the motifs. If the answer is no, we remove the "offend-

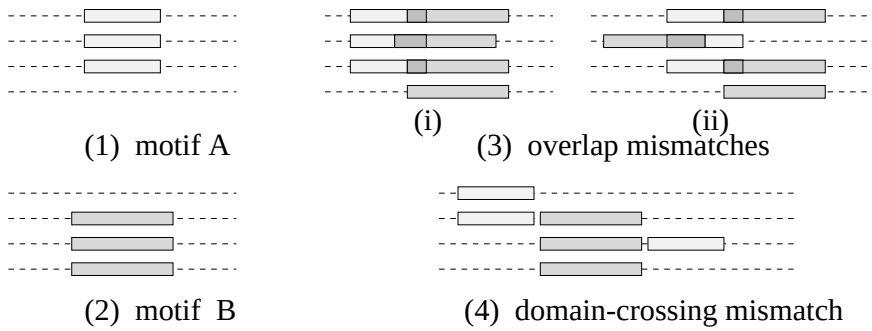


Figure 10.1: A schematic representation of two motifs A and B shown on the left and the different ways they can be incompatible (pairwise) is shown on the right. The four sequences are shown by dashed lines and the motifs are shown as filled rectangles.

ing" motifs in a manner that optimizes a cost function and gives an alignment. To investigate this further, we now explore the conditions under which *two* motifs can be used simultaneously in an alignment.

Definition 8 *(Motif Overlap) Two irredundant motifs p_i and p_j overlap if there exists a sequence s containing both these motifs and the following holds. Let n_i and n_j be the sizes of the motifs p_i and p_j and let l_i and l_j be the locations (offsets) in a sequence s respectively, then the motifs overlap if the intervals $[l_i, l_i + n_i]$ and $[l_j, l_j + n_j]$ have a non-empty intersection.*

Definition 9 *(Pairwise Compatible Motifs) Two motifs, p_1 and p_2, are pairwise compatible if there exists an alignment of the sequences that does not introduce gaps in the motifs p_1 and p_2.*

Lemma 5 *Two irredundant motifs p_i and p_j are pairwise compatible if and only if none of the following holds:*

1. *If p_i and p_j do not overlap in* all *the sequences, then p_i is to the left of p_j, without loss of generality (otherwise a* domain crossing mismatch *is said to have occurred).*

2. *If p_i and p_j overlap in* any *sequence, then p_i is at some fixed distance d to the left of p_j, without loss of generality (otherwise an* overlap mismatch *is said to have occurred).*

See Figure 10.1 for a schematic diagram. We also give examples of overlap mismatch and domain crossing mismatches below. Notice that in each of the cases, there is no alignment that can align both the motifs simultaneously.

1. (1) $A\ \ x\ \ C\ \ D\ \ E$ $\quad$ (i) $A...E$ in seq 1 and 2,
   (2) $A\ \ C\ \ D\ \ E\ \ E$ $\quad$ (ii) CDE in seq 1 and 2.
   (overlap mismatch)

2. (1) $A\ \ B\ \ C\ \ X\ \ Y$ $\quad$ (i) ABC in seq 1 and 2,
   (2) $X\ \ Y\ \ A\ \ B\ \ C$ $\quad$ (ii) XY in seq 1 and 2.
   (domain crossing)

We define the alignment using a set of motifs as follows.

Definition 10 *(sequence alignment, compatible set) Given a set S of motifs,*

$$v_{i_1}, v_{i_2}, \ldots, v_{i_n},$$

a motif-alignment of the sequences, $s_1, s_2, \ldots, s_m$, is the alignment such that in all the sequences, with no gaps in the motifs, *the motifs $v_{i_1}, v_{i_2}, \ldots, v_{i_n}$, are aligned (in all the sequences they appear). If such an alignment exists, the set $s_1, s_2, \ldots, s_m$, is called a compatible set.*

Definition 11 *(linear ordering of motifs) Given a set of compatible motifs, a consistent ordering of the motifs such that, in every sequence, the set of motifs that are present in the sequence appear in the left to right order and this ordering is called the linear ordering.*

Example 3 Consider the following sequences and the corresponding irredundant motifs. The solid-characters of the motifs have been emboldened for easy visualization.

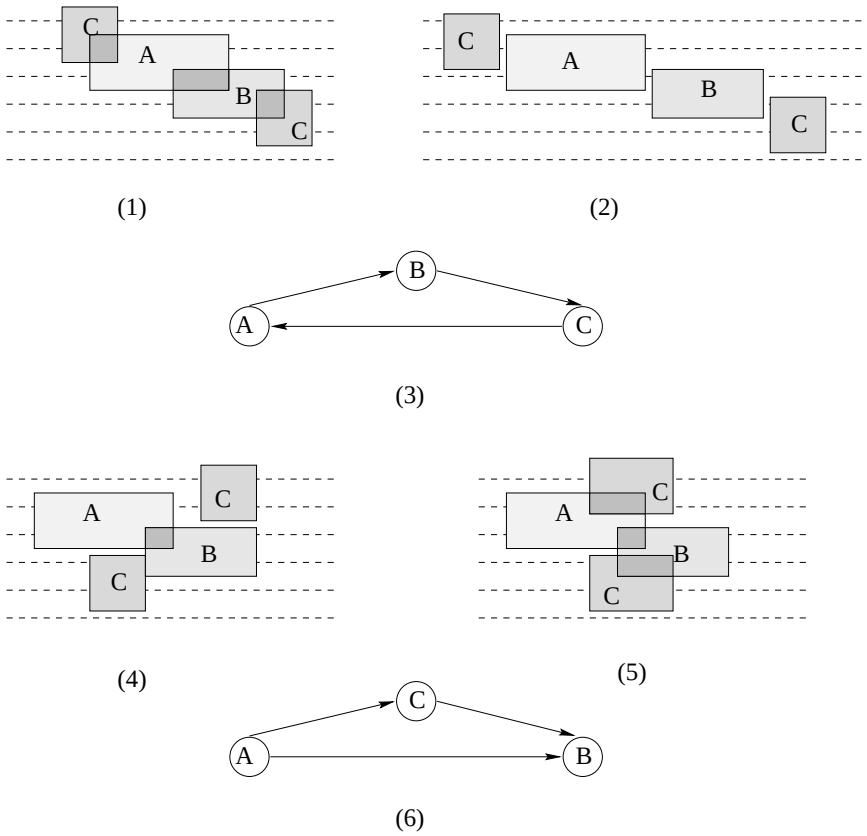


Figure 10.2: The motifs are shown as filled rectangles spanning across the sequences which is shown as dashed lines. (1) and (2) show configurations of motifs A, B and C that can never give an alignment that respects all the three motifs. (3) shows the graph (corresponding to (1) and (2)) where each node corresponds to a motif and a directed edge captures the left-to-right information: notice that the directed cycle in the graph captures the infeasiblity of the three motifs. (4) shows another configuration where it is impossible to obtain an alignment that respects all the three motifs whereas (5) shows a configuration that is very similar to (4), yet has an alignment as shown. (6) shows the graph (corresponding to (4) and (5)) that captures the relationship: notice that closed path of the graph is not sufficent to indicate an infeasibility as demonstrated by the two configurations in (4) and (5).

(1)	**H**	**I**	**A**	**J**	**G**	*L*	**B**				
(2)	**A**	*M*	**C**	*N*	**B**	*Q*	**D**				
(3)	**C**	*O*	*P*	**E**	**D**	*R*	**F**				
(4)	**H**	**I**	*S*	**J**	**G**	**E**	*T*	*U*	**F**		
(5)	**H**	**I**	*V*	**J**	**G**						

(*i*)	*A...B*	in seq 1 & 2,
(*ii*)	*C...D*	in seq 2 & 3,
(*iii*)	*E..F*	in seq 3 & 4,
(*iv*)	*HI.JG*	in seq 1, 4 & 5.

The consistent linear ordering of the motifs is *iv*, *i*, *ii*, *iii* and the alignment of the sequences respecting all the motifs is as follows.

	*	*	*	*	*		*	*	*		*
(1)	**H**	**I**	**A**	**J**	**G**	*L*	**B**	–	–	–	–
(2)	–	–	**A**	*M*	**C**	*N*	**B**	*Q*	**D**	–	–
(3)	–	–	–	–	**C**	*O*	*P*	**E**	**D**	*R*	**F**
(4)	**H**	**I**	*S*	**J**	**G**	–	–	**E**	*T*	*U*	**F**
(5)	**H**	**I**	*V*	**J**	**G**	–	–	–	–	–	–

The asterisk marked columns show the aligned characters (with at least 2 occurrences).

Is it sufficient to just check for pairwise incompatibility of motifs while seeking an alignment? We explore this in the next two examples.

Example 4 Consider the following sequences and the corresponding irredundant motifs. The solid-characters of the motifs have been emboldened for easy visualization.

(1)	**H**	**I**	**A**	**J**	**G**	*L*	**B**			
(2)			**A**	*M*	**C**	*N*	**B**	*Q*	**D**	
(3)					**C**	*O*	*P*	**E**	**D**	*R*	**F**
(4)						**H**	**I**	**E**	**J**	**G**	**F**
(5)						**H**	**I**	*K*	**J**	**G**	

(*i*)	*A...B*	in seq 1 & 2,
(*ii*)	*C...D*	in seq 2 & 3,
(*iii*)	*E..F*	in seq 3 & 4,
(*iv*)	*HI.JG*	in seq 1, 4 & 5.

There does not exist an alignment respecting these motifs due to the *overlap mismatch*. In other words, any alignment that aligns these motifs in the respective sequences must introduce gaps in at least one of the motif. Notice that it is not possible to put the motifs in any consistent left-to-right order, *i.e.*, there is no linear ordering of the motifs.

In the following example we show that having just a linear order is not sufficient and we must also remove the mismatch due to domain crossing.

Example 5 Consider the following sequences and the corresponding irredundant motifs.

(1)	**A**	G	H	**D**	**X**	**Y**	(i)	A..D	in sequences 1 and 2,
(2)	**A**	I	**C**	**D**	J	**F**	(ii)	CD.F	in sequences 2 and 3,
(3)	**X**	**Y**	**C**	**D**	K	**F**	(iii)	XY	in sequences 1 and 3.

The consistent linear ordering of the motifs is *i*, *iii*, *ii*. but there does not exist an alignment respecting these motifs due to the *domain crossing mismatch.*

Definition 12 *(domain crossing error) Given a set of motifs, $m_1, m_2, \ldots, m_n$, a domain crossing error is said to occur if there exists a linear ordering of the motifs $m_{i_1}, m_{i_2}, \ldots, m_{i_n}$, yet there exists no alignment that respects all the n motifs.*

Lemma 6 *A set of irredundant motifs $p_1, p_2, \ldots, p_n$ is feasible* if and only if *none of the following holds:*

1. *There exist distinct motifs p_i and p_j such that p_i and p_j are pairwise incompatible.*
2. *There exists a non-empty subset of the motifs* without *a linear ordering.*
3. *There exists a non-empty subset of the motifs that demonstrate* domain crossing *error.*

10.1.1 The Graph-theoretic Formulation

Next we wish to capture these conditions in a graph as follows. Construct a directed graph $G = (V, E)$ where every motif p_i corresponds to a vertex v_i, thus $N = |V|$. The directed edges are introduced as follows:

1. There is no edge between two vertices where the two corresponding motifs do not occur simultaneously in any sequence.

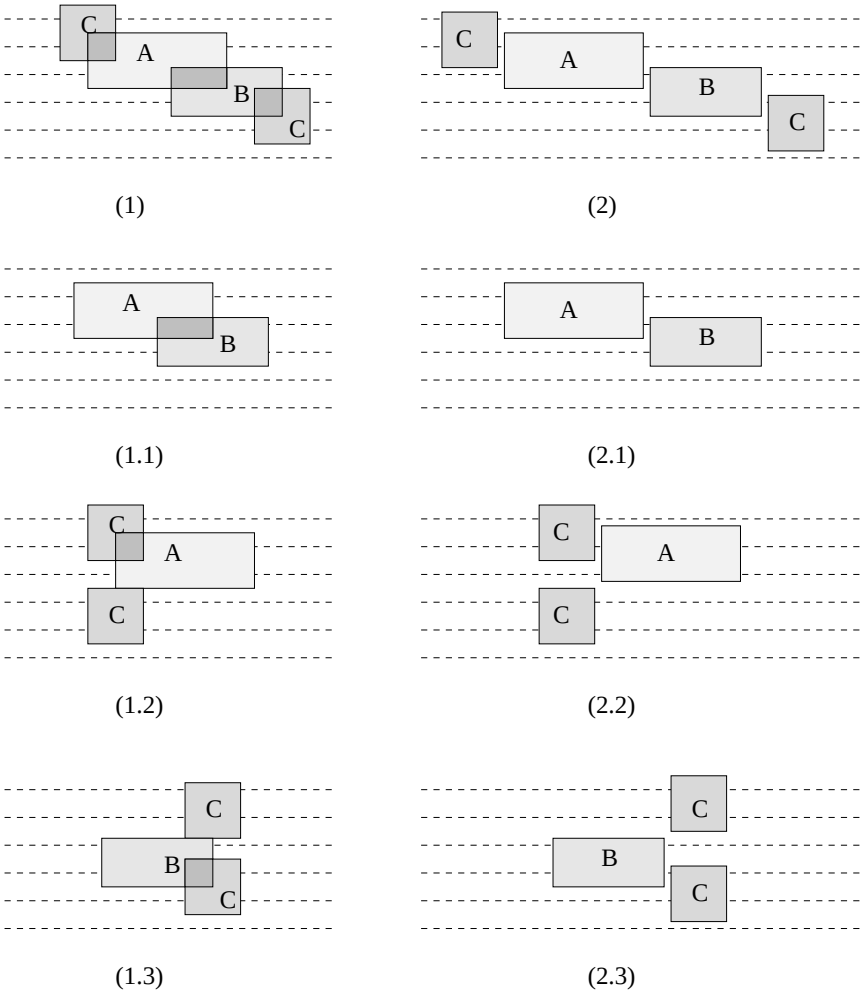


Figure 10.3: (1) and (2) show configurations of motifs that can never give an alignment that respects all the three motifs. (1.1) to (1.3) show the alignments by ignoring exactly one motif, *viz.*, motif C, B and A respectively. (2.1) to (2.3) show similar alignments for the configuration shown in (2).

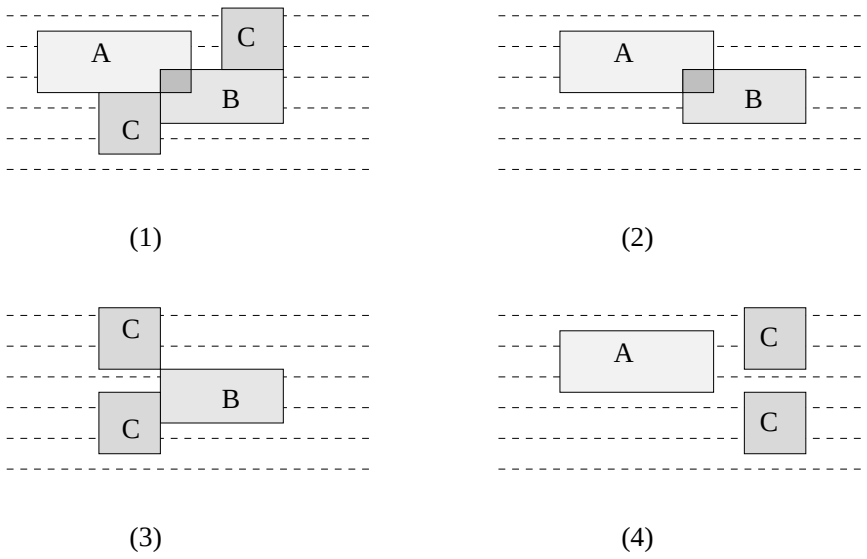


Figure 10.4: (1) shows a configuration of motifs that can never give an alignment that respects all the three motifs. (2) to (4) show the alignments by ignoring exactly one motif, *viz.*, motif C, A and B respectively.

2. If p_i is to the left of p_j in *every* sequence that the two motifs are present, then a directed edge is placed from v_i to v_j. This is to indicate that in the alignment the motif p_i appears to the left of p_j. The edges are labeled as follows:

   (a) Label forbidden, if the motifs corresponding to v_1 and v_2 are not pairwise compatible.

   (b) Label overlap, if the motifs corresponding to v_1 and v_2 overlap.

   (c) Label nonoverlap, if the motifs corresponding to v_1 and v_2 are pairwise compatible but do not overlap.

The linear ordering of motifs is captured by checking for cycles in the graph. However the domain crossing mismatch requires a more careful handling as described below.

Handling domain crossing mismatches. We associate a *distance* $\mathcal{D}_{v_1 v_2}$ with every edge that is *not* labeled forbidden. This is used to compute the feasibility of a collection of motifs corresponding to a solution; this *does not* contribute to the cost of the alignment. (We discuss the weight corresponding to the cost in the next section.) Let p_i and p_j be the two motifs corresponding to the vertices. Then if d is the *minimum* of the distance between the occurrences of the two motifs in every sequence that both of them appear in, $\mathcal{D}_{v_i v_j} = d$.

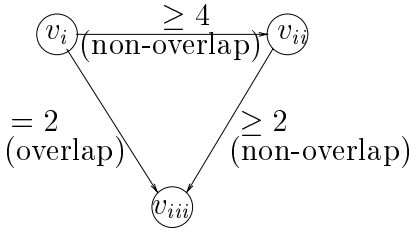
To detect the *domain crossing mismatches* of motifs (that are pairwise compatible), we define the notion of a *consistent graph w.r.t. a vertex.*

Definition 13 *Let $G = (V, E)$ be a labeled, weighted, directed, graph with weights on the edge uv given by $\mathcal{D}(u, v)$ and a label $\in \{forbidden, overlap, nonoverlap\}$. A path, $\mathcal{P}$, is valid if it has no edges labeled* forbidden. *Further, a valid path, $\mathcal{P}$, is called an overlap-path if all the edges in the path are labeled overlap. The weight of the valid path $\mathcal{P}$, $\mathcal{D}_{\mathcal{P}}$, is the sum of the weights of its constituent edges.*

Let $p \in V$. The graph is consistent w.r.t p if $\forall\ q \in V$, for all pairs of vertex-disjoint valid paths from p to q, $\mathcal{P}^1$ and $\mathcal{P}^2$,

1. *$\mathcal{D}_{\mathcal{P}^1} = \mathcal{D}_{\mathcal{P}^2}$, if $\mathcal{P}^1$ and $\mathcal{P}^2$ are both overlap-paths, or,*
2. *$\mathcal{D}_{\mathcal{P}^1} \geq \mathcal{D}_{\mathcal{P}^2}$, if $\mathcal{P}^1$ is an overlap-path and $\mathcal{P}^2$ is not.*

Example 6 Consider the example 5. We construct the corresponding graph as shown below.



The vertices corresponding to the motifs are v_i, v_{ii} and v_{iii}. Let $\mathcal{P}_1$ be the overlap-path $v_i \rightarrow v_{iii}$ and let $\mathcal{P}_2$ be the nonoverlap-path $v_i \rightarrow v_{ii} \rightarrow v_{iii}$. Further, $\mathcal{D}_{\mathcal{P}_1} = 2$ and $\mathcal{D}_{\mathcal{P}_2} = 6$. Thus the graph is not consistent w.r.t the vertex v_i since $\mathcal{D}_{\mathcal{P}_1} \ngeq \mathcal{D}_{\mathcal{P}_2}$.

We now present the straightforward observation that relates the set of compatible motifs to a feasible subgraph.

Lemma 7 *The following two statements are equivalent:*

- *Given a subset of motifs $p_1, p_2, \ldots, p_n$ from the set of all motifs from m sequences of input, the subset is compatible, if the following holds:*
  1. *the motifs are not pairwise incompatible,*
  2. *there exists a linear ordering of $p_1, p_2, \ldots, p_n$, and,*
  3. *there is no domain crossing mismatch in $p_1, p_2, \ldots, p_n$.*
- *Given a subset of vertices $v_1, v_2, \ldots, v_n$, constructed as defined in this section. The induced subgraph on $v_1, v_2, \ldots, v_n$ is feasible, if the following holds:*
  1. *there is no edge labeled* forbidden *in the induced subgraph,*
  2. *the induced subgraph is acyclic, and,*
  3. *the induced subgraph is consistent w.r.t. every vertex v_i, $1 \leq i \leq n$.*

10.1.2 Measuring the quality of an alignment

We have outlined our approach to solving the multiple sequence alignment problem in the preceding sections. Now we identify the underlying optimization function and give two cost functions that arise naturally in our approach. Our motivation for these cost functions comes naturally from our use of common motifs in the sequences to align them.

Problem 1 *(k-MSA Problem) Given m sequences on an alphabet set, Σ, and a natural number k, find an alignment $\mathcal{A}$ of these sequences so that $\mathcal{C}_\mathcal{A}$ is is maximized where $\mathcal{C}_\mathcal{A}$ is the number of characters that match (exactly) in at least k sequences, where k is the alignment number.*

Problem 2 *(k-MSA$_{\max}$ Problem) Given m sequences on Σ, and a natural number k, find an alignment $\mathcal{A}$ of these sequences so that $\mathcal{C}_\mathcal{A}$ is maximized where $\mathcal{C}_\mathcal{A}$ is the number of columns that have at least k identical characters.*

Let $1 - \Upsilon$ denotes the upper bound on the polynomial time approximation factor of the MC problem.

Lemma 8 *k-MSA is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating it within a factor of $1 - \epsilon$ is NP-hard. Also, achieving an approximation ratio $1 - \Upsilon/6$ for this problem is NP-hard.*

Proof Sketch. Consider the special case where the gaps appear only at the left end or the right end of every sequence in the alignment. Further, let the gap size be at most 1 which is either at the left end or at the right end. For this case, we can give a gap-preserving reduction of an instance of the maxcut problem to an instance of the k-MSA problem. The proof follows along the lines of the proof of the hardness result for the binary shift cut problem in [41] and Chapter 8. QED

Lemma 9 *k-MSA$_{\max}$ is NP-hard. Further, there exists a constant $\epsilon > 0$ such that approximating it within a factor of $1-\epsilon$ is NP-hard. Also, if K denotes the number of columns that have at least k characters perfectly aligned, then there does not exist a polynomial time algorithm (assuming $P \neq NP$) that guarantees the estimation of $(1 - \Upsilon/6)K$.*

Proof Sketch. We show that if we have a polynomial time approximation scheme (PTAS) for k-MSA$_{\max}$, we have a PTAS for k-MSA, which is a contradiction. Let $\tilde{X}$ denote an approximate solution and X^* denote the optimal solution. Then if k-MSA$_{\max}$ has a PTAS let $\frac{\tilde{N}}{N^*} \geq \epsilon$ for some $0 < \epsilon \leq 1$. Note that N^* is the number of columns that have at least k identical characters. Let $\tilde{C} \geq \tilde{N}k$, then $C^* \leq N^*k$. Hence we have

$$\frac{\tilde{C}}{C^*} \geq \frac{\tilde{N}k}{N^*k} \geq \epsilon.$$

This concludes the argument. QED

The reader may notice the connection between the motifs and the k-MSA and the k-MSA$_{\max}$ optimization problems. It is quite clear that if motif p is used in an alignment, the cost that p contributes to the optimal configuration is $\geq ks$ for the k-MSA problem and s for the k-MSA$_{\max}$ problem, where s is the number of solid-characters in the motif p. Now, we state the following lemma in support of our use of *only irredundant motifs* in the alignment process.

Lemma 10 *If p is a redundant motif, then using the motif p does not improve the cost of the k-MSA and the k-MSA$_{\max}$ optimization problems.*

Proof Sketch. Let p be rendered redundant by motifs $p_1, p_2, \ldots, p_n$, $n \geq 1$. By definition, motif p has less number of solid-characters than each of p_i, $1 \leq i \leq n$. Thus if an alignment can use motif p, it can certainly use all the motifs $p_1, p_2, \ldots, p_n$, giving a larger number of solid-characters; thus a higher cost for the k-MSA and the k-MSA$_{\max}$ optimization problems. QED

10.1.3 Algorithm to compute the "best" alignment

Given a set of incompatible motifs, the set can be grouped into sets (not necessarily disjoint) such that each set violates *exactly one* of the three conditions of Lemma (6). These sets are called *basic incompatible sets*. Next, it can be easily shown that we can remove *exactly one* motif from a basic incompatible set to make it compatible. See Figures (10.3) and (10.4) for examples.

The algorithm proceeds in the following three steps.

1. Detect the *basic infeasible* (sub)sets.
2. Eliminate motifs to obtain a feasible set that maximizes the cost.
3. Render the alignment.

Step 1. In this step we form subset of vertices that lead to incompatibility of the motifs. Using lemma (7), we compute the following sets:

1. Construct the sets F_1, F_2, ..., F_{n_f} where each set consists of two vertices which are the end points of an edge labeled Forbidden. This is done by simply scanning all the edges and collecting the end points of the edges labeled Forbidden.
2. Construct the sets C_1, C_2, ..., C_{n_c} where each set consists of vertices that form a directed cycle in the graph. All the cycles are captured by carrying out a depth first search (DFS) of the graph.

3. Construct the sets P_1, P_2, ..., P_{n_p} where each set consists of vertices that form a closed path in the graph. These are captured by carrying out a breadth first search (BFS) rooted at each vertex.

It is easy to see that the basic incompatible sets are

$$F_1, F_2 \ldots, F_{n_f}, C_1, C_2 \ldots, C_{n_c}, P_1, P_2 \ldots, P_{n_p}.$$

Step 2. Set-covering problem. An instance $(X, \mathcal{Y})$ of the set-covering problem consists of a finite set X and a family $\mathcal{Y}$ of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{Y}$: $X = \cup_{S \in \mathcal{Y}} S$. We say that a subset $S \in \mathcal{Y}$ *covers* its elements. The problem is to find a minimum-size subset $\mathcal{A} \subseteq \mathcal{Y}$ whose members cover all of X: $X = \cup_{S \in \mathcal{A}} S$. Any $\mathcal{A}$ satisfying this condition covers X. See [13] for details on this problem.

We construct an instance of a set cover problem (the dual of our problem) as follows. Let

$$\{v_1, v_2, \ldots, v_n\} = F_1 \cup F_2 \ldots \cup F_{n_f} \cup C_1 \cup C_2 \ldots \cup C_{n_c} \cup P_1 \cup P_2 \ldots \cup P_{n_p}$$

The elements of the the set (X of the set cover problem) are

$$F_1, F_2 \ldots, F_{n_f}, C_1, C_2 \ldots, C_{n_c}, P_1, P_2 \ldots, P_{n_p}.$$

The subset S_i corresponds to each v_i, $1 \leq i \leq n$ (where $\mathcal{Y} = \{S_1, S_2 \ldots, S_n\}$), and is defined as

$$S_i = \{F_l | v_i \in F_l, 1 \leq l \leq n_f\} \cup \{C_l | v_i \in C_l, 1 \leq l \leq n_c\} \cup \{P_l | v_i \in P_l, 1 \leq l \leq n_p\}.$$

Thus S_i denotes all the basic incompatible set that has a common element v_i, and removing the motif corresponding to v_i suffices to make all the corresponding basic sets compatible. Now, it is easy to see that a solution to the set-covering problem gives the minimum number of motifs that need to be removed so that the remaining set of motifs is compatible.

Associating weights to S_i which reflect the weight of each motif (depending on the cost function), gives a weighted set cover problem. The set-covering problem is known to be MAX SNP hard and the greedy algorithm is the best known

approximation algorithm for the problem, assuming $P \neq NP$ [4]. To reflect the underlying cost function a weight is associated with every motif in the following manner. Let c be the number of solid characters in the corresponding motif, p_i, and let the number of sequences containing p_i be l, then for the k-MSA problem the associated weight is cl and for k-MSA$_{\max}$ it is c. We ignore the change in cost due to the common cost of a set of motifs [3].

Step 3. We define a *block* to be an induced subgraph of the feasible graph where all the edges are labeled overlap. For each block, we compute an ordering of the motifs $p^i_{j_1}$, $p^i_{j_2}$, ..., $p^i_{j_i}$ for each sequence i. Such a linear ordering of the motifs exists, since the set of motifs is compatible. From the original sequence s_i, we obtain the fillers (if any) between two consecutive motifs as f^i_0, $f^i_{j_1}$, $f^i_{j_2}$, ..., $f^i_{j_i}$. f^i_0 is the leftmost portion, possibly empty. We obtain an alignment of the sequences by appropriately aligning each $(p^i_{j_l}+f^i_{j_l})$ and f^i_0, $l = 1, 2, \ldots, j_i$, filling the gaps with '-'. The symbol '+' denotes a string concatenation operation. The alignment is such that each motif of a sequence is perfectly aligned with the corresponding motif in all the other sequences.

10.2 Experimental Results

In this section we show samples of some of results of our implementation. Our experience has shown that, the clique computation helps in trimming down the size of the problem drastically since the pairwise incompatibility succeeds in winnowing out spurious alignments quickly.

The characters appearing in the sequences represent the amino acid residue using the standard notation. For exposition, consider Figure Result-1.

1. The uppercase letter shows the *aligned* characters.

2. The lowercase letters are portions of the input sequence that the algorithm has not taken into account (by picking up a certain set of compatible motifs – see Step 3 of the algorithm).

[3] For a more accurate computation of the cost an appropriate common cost due to a set of overlapping motifs must be used.

3. The dots ('.') represent disagreeing characters along the vertical column or "don't cares".
4. The dashes ('-') represent the gaps enforced by the algorithm. We omit the input sequences due to space constraints.
5. Some portions of the input has been removed to avoid clutter in the presentation – this is denoted by '=='.

The amount of time taken in these cases is of the order of few minutes.

We show the results on three sets of data: the first set is on a highly similar data (histones) set and the second set is on a fairly dissimilar (cytosine-specific DNA methylases listed in PROSITE database Release 13 entry accession #PS00094) data set. The first data set has 20 sequences of about 200 residues each; the second has 46 sequences with lengths going upto 500 residues. For the second example we have truncated some of the sequences in the alignment and show the output in Figure 2 (that runs across pages due to the size).

```
( 1)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
( 2)   ARTKQTA.KSTGGKAPRKQL..KAA.K.AP..GGVKKPH...PGTVAL
( 3)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
( 4)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
( 5)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
( 6)   ARTKQTA.KSTG.KAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
( 7)       == R.S..G.A.R.Q..TK.A.......GGVK.........V--
( 8)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
( 9)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
(10)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
(11)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
(12)   ARTKQTARKSTGGKAPRKQL.TKAA.K.AP..GGVKKPH..RPGTVAL
(13)   ARTKQTA.KSTGGKAPRKQL..KAA.K.AP..GGVKKPH...PGTVAL
(14)       == R.S..........TK.A.......GGVK------------
(15)       == R.S..........TK.A.......GGVK------------
(16) sg-R.K.......GG.A.Rhrkvlrdni......GGVK------------
(17) sg-R.K.......GG.A.Rhrkvlrdni......GGVK------------
```

```
(18) sg-R.K.......GG.A.Rhrkilrdni......GGVK------------
(19) sg-R.K.......GG.A.Rhrkvlrdni......GGVK------------
(20) sg-R.K.......GG.A.Rhrkilrdni......GGVK.........V--


( 1) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
( 2) REIRR.QKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SA..ALQE..EAYL
( 3) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
( 4) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
( 5) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ..A..ALQEA.EAYL
( 6) .EI.RYQKSTELLI...PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
( 7) ----R...STELLI...PFQRLV.EIAQDFKT.LRFQ..A..ALQEA.EA..
( 8) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
( 9) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
(10) .EI..YQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
(11) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
(12) REIRRYQKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SAV.ALQEA.EAYL
(13) REIRR.QKSTELLIR..PFQRLVREIAQDFKT.LRFQ.SA..ALQE..EAYL
(14) R..RR-------L.R.....R....I..D....L......V-----------
(15) R..RR-------L.R.....R....I..D....L......V-----------
(16) R..RR-------L.R.....R....I.......Lkvflen------------
(17) R..RR-------L.R.....R....I.......Lkvflen------------
(18) R..RRggvkrisa--------LV.Eetravlklflen---------------
(19) R..RR-------L.R.....R....I.......Lkiflen------------
(20) R..RR-------L.R.....R....I.......Lks..S.........E---


( 1) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
( 2) V.LFEDTNL.AIH.KRVTI..KD..L.RR.RGER
( 3) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
( 4) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
```

```
( 5) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
( 6) V.LFEDTNLCAIHAKRV.IMPKDIQL.RRIRGERA
( 7) V.LFEDTNLCAIHAK.VT..PKDIQL...I.GERA
( 8) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
( 9) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
(10) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
(11) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
(12) V.LFEDTNLCAIHAKRVTIMPKDIQL.RRIRGERA
(13) V.LFEDTNL.AIH.KRVTI..KD..L.RR.RGER
(14) V.....T.......K.VT.......L.R..Rtlygfgg
(15) V.....T.......K.VT.......L.R..Rtlygfgg
(16) V.....T.......K.VT.......L.R..Rtlygfgg
(17) V.....T.......K.VT.......L.R..Rtlygfgg
(18) V.....T.......K.VT.......L.R..Rtiygfgg
(19) V.....T.......K.VT.......L.R..Rtlygfgg
(20) V.....T.......K.VT.......L.R..Rtlygfgg
```

Figure Result-1: (Histones) 40 protein sequences, with an average size of 150 residues, have been aligned with an *alignment number* $k = 10$. The number of maximal motifs for this is $N = 648$ and the size of the resulting feasible set of motifs is 22.

```
( 1) llpkppahhphyafrfIDLFAGIGG...GFE..G == E....A...Y..N ==
( 2)      mskanakysfv-DLFAGIGG....L...G == E.D..A...Y..N ==
( 3)            mekklI.LF.G.GG...GF..AG == ------------- ==
( 4)          mkfrkge--LF.G.GG..LG...Ak == ekfgei------- ==
( 5) sndeiqyksdkfnvls--LF.G.GG..LGFE.AG == -----A...Y..N ==
( 6)            mkvls--LF.G.GG..LGLE--- == ------------- ==
( 7)           mqqfrfIDLFAGIGG..LGLE..G == E.D..A...Y..N ==
( 8)           mlqias--LFAG.GG..LGFE..G == E.D..A...Y..N ==
( 9)           mkikfv-DLFAGIGG...GFE.A  == E.D..A...Y..N ==
(10)         msklrvms--LF.GIG.....L...G == E.D..A...Ycai ==
```

```
(11) tnnnfekqnrgikltfIDLFAGIGG..LGFE--- == E.D..A...Y..N ==
(12)         msklrvms--LF.GIG.....L...G == E.D..A...Ycai ==
(13) iiekvdfrtdkinvls--LF.G.GG..LGFE.AG == -----A...Y..N ==
(14)         msklrvms--LF.GIG.....L...G == E.D..A...Ycai ==
(15)         mgklrvms--LF.GIG.....L...G == ------------ ==
(16)          msfrtle--LF.GI......Lrgis == ------------ ==
(17)             mlkfIDLFAGIGG..LGFE.A- == E.D..A...Y..N ==
(18)           mkqfrfIDLFAGIGG..LGLE..G == E.D..A...Y..N ==
(19)              mniIDLFAG.GG...GF..AG == E.D..A...Y..N ==
(20)              mktIDLFAG.GG..LGF..AG == -----A...Y..N ==
(21)            mvgavIDLF.G.GG...GL...G == ------------ ==
(22) ivkpveppkeirlatl-D.FAG.GG...GL..AG == anelaaklteeqk ==
(23) psepeieiklpklrtl-D.F.G.GG...GF..AG == kgdvem------- ==
(24) pkepeaaiklpklrtl-D.F.G.GG...GF..AG == kgdvem------- ==
(25)             mkcv-DLF.G.GG..LGFE.AG == E....A...Y..N ==
(26)           mqqfrfIDLFAGIGG..LGLE..G == E.D..A...Y..N ==
(27) mlpeapahhpdyafrfIDLFAGIGG...GFE..G == E....A...Y..N ==
(28)            mklls--LF.G.GG..LGFE.AG == ------------ ==
(29)   mkknimglslfssa-----GIGeyflsrvgid == ------------ ==
(30)              mktIDLFAG.GG..LGF..AG == -----A...Y..N ==
(31)     mkinamslfssa-----GIGeldlhkgnln == ------------ ==
(32)   mieikdkqltglrfIDLFAG.GG..L.LE..G == E.D..A...Y..N ==
(33) lfepesnpnlrekftfIDLFAGIGG........G == E.D..A...Y..N ==
(34)              mnlI.LF.G.GG..LGF..AG == ------------ ==
(35)         mnmdiasf---F.G.GG..LGF..AG == ------------ ==
(36) thieerkdayssdfkfIDLF.GIGG.....E..G == E.D..A...Y..N ==
(37)              mkiI.LF.G.GG..LGFE.AG == ------------ ==
(38) lrlnrpdwnviegdvr--LF.G.G.....L...G == ------------ ==
(39)             mfkiIDLFAGIGG..LGFE.A- == E.D..A...Y..N ==
(40) pdtppypnnengryrmIDLFAGIGG..LGF---- == E.D..A...Y..N ==
(41)         mnkikvve--LFAG.GG..LGLE--- == ------------ ==
```

```
(42) dvsnvrknkdynvfet---FAG.GG..LGLE.AG == ------------ ==
(43) peiapfenrktakykmIDLFAGIGG..LGF---- == E.D..A...Y..N ==
(44)          mlrvfea---FAG.G.....Likan == ktfenradklggq ==
(45) tedilklagvssgnei----------------- == ------------ ==
(46) mskvenktkklrvfea---FAGIG.....L---- == ------------ ==


( 1) P..D.L..GFPCQ.FS.AG......D.RG.LF....RI.....
( 2) ---D.L.GGFPCQ.FS..G...G....RG.LF....RI.....
( 3) ---D...GG.PCQ.FS.AGKR.G..D.RG.L.....R......
( 4) ---D....GFPC...S..G...G.....G.L------I.....
( 5) P......GGFPC..FS.AG.R----D....L.....R......
( 6) ---D...GGFPCQ.FS.AGKR.GF-------------------
( 7) P..D...GG.PCQ..S.AGK...F.D.RG.L.....R......
( 8) P..D.L..GFPC..FS.AG.R.GF.D--G.LF....R......
( 9) P..D.L..GFPCQ.FS.AGK..GF.D.RG.LF....R......
(10) P..D.L..GFPC..FS.AG.R.G--D.RG.LF...........
(11) P..D.L..GFPCQ.FS..GKR.GF...R--------RI.....
(12) P..D.L..GFPC..FS.AG.R.G--D.RG.LF...........
(13) P......GGFPC..FS.AG.R....D....L.....R......
(14) P..D.L.GG.PCQ.FS.AG.R.GF.D.RG.LF...........
(15) P..D.L.GG.PCQ.FS.AG.R.GF.D.RG.LF...........
(16) ---D....GFPC..FS.AG.R.GF------L.....R......
(17) P..D.L..GFPCQ.FS.AGK..GF.D.RG.LF....RI.....
(18) P..D...GG.PCQ..S.AGK...F.D.RG.L.....R......
(19) ---D...GG.PCQ.FS..G.R----D....LF....R......
(20) -------GG.PCQ.FS.AGKR----------------I.....
(21) ----------PCQ..Sqytkksrtgtkwq-L.....R......
(22) ---D...GG.PCQ..S--------------------------
(23) -----L.GG.PCQ.FS--------------------------
(24) -----L.GG.PCQ.FS--------------------------
```

```
(25) ---D...GG.PCQ.FS.AGKR----------------I.....
(26) P..D...GG.PCQ..S.AGK...F.D.RG.L.....R......
(27) P..D.L..GFPCQ.FS.AG........RG.LF....RI.....
(28) P..D...GG.PCQ..S.AG...G..D.RG.LF....RI.....
(29) ----------PCQ..S.AGKnrdvsnmandnrnylimyviami
(30) -------GG.PCQ.FS.AGKR----------------I.....
(31) ----------PCQ..S..GKnkhqdh-----------------
(32) P..D.L..GFPCQ.FS..GK..GF.D.RG.LF....RI.....
(33) ---D.L..GFPCQ.FS.AGKR.GF.D.RG.LF.....I.....
(34) P..D...GG.PCQ..S..G...G..D.RG.LF....RI.....
(35) P......GG.PCQ..S.AG...G..D.RG..F-----------
(36) P..D.L..GFPCQ.FS..GKR.GF--------------------
(37) ---D...GG.PCQ..S.AG...G..D.RG.LF....RI.....
(38) ---D.L.GG.PC..FS.AGK..G..D....LF....R......
(39) P..D.L..GFPCQ.FS.AG...GF.D.RG.LF....RI.....
(40) P....L.GGFPC..FS.AG...GF.D.RG.LF....RI.....
(41) ---D...GGFPCQ..S.A----------G.LF....R......
(42) ---D.L.GG.PCQ.FS.AGKR.GF.D.RG.LF.....I.....
(43) P..D.L.GGFPCQ.FS.AGK..GF.D.RG.LF....RI.....
(44) dfftys---FPCQ..S.AG...G.....Gtrssllwecck...
(45) ---D...GG.PCQ.FS.AGKR.G..D.RG..F-----------
(46) -----L...FPCQ..S..G...G.....Gtrsgllweierald


( 1) P..F.LENVK.L.....G == GY.......Ngpddpkiidgkhfl.VG
( 2) P....LENV..L------ == --------L.A...G.PQ.RERV.I
( 3) P..F..ENVKG......G == -Y......LNA.D.GVPQ.RERV.IVG
( 4) P..F..ENV.GL.....G == -------------YGVPQ.R.R..IVG
( 5) P..F..ENVKG......G == GY......LNA.DYGVPQ.RERV.IVG
( 6) P..F..ENVKGL------ == --------LNA..YGV.Q.RERV...G
( 7) P..F..ENVKGL------ == GY......LN....GV.Q.R.RV.IVG
```

(8) P....LENVK.L.....G == GY......LNA.D.G-PQ.RER..IVG

(9) P..F.LENV.GL.....G == GY......LN....GVPQ.R.R..I.G

(10) P ==

(11) P..F.LENVKGL.....G == -Y........A...G.PQ.RER..IVG

(12) P....LENVKGL.....G == GY......LN.....VPQ.RERV.I.G

(13) P..F..ENVKG......G == GY......LNA.DYGVPQ.RERV.I.G

(14) P..F..ENVKGL.....G == GY......LN.....VPQ.RER..I.G

(15) P..F..ENVKGL.....G == GY......LN.....VPQ.RER..I.G

(16) P....LEN ==

(17) P..F.LENV.GL.....G == --------LN.....VPQ.R.RV.IVG

(18) P..F..ENVKGL------ == GY......LN...YGV.Q.R.RV.I.G

(19) P..F..ENV.G------- == GY......LNA.DYGVPQ.R.RV...G

(20) P.....ENV--------- == GY......L.A...GVPQ.R.R....G

(21) P.....ENV........G == -------------YG.PQ.R.R

(22) P....LENV........G == GY......L.A..YGV.Q.R.R..I

(23) P....LENV--------- == GY......L.A..YGV.Q.R.R..I

(24) P....LENV--------- == GY......L.A..YGV.Q.R.R..I

(25) P.....ENV--------- == GY......L.A...GVPQ.R.R....G

(26) P..F..ENVKGL------ == GY......LN....GV.Q.R.RV.IVG

(27) P..F.LENVK.L.....G == ----------------PQ.RER...VG

(28) P..F..ENVKG------- == GY......LNA.DYGV.Q.RERV..VG

(29) kklk..ENV..L------ == -Y......L.A.DYG.PQ.R.R..I

(30) P.....ENV-Y......LNA.DYG.PQ.RER

(31) ---F...N...Lifevfe == nvprfiemyfpyngqllllleeilkikyask

(32) P.....ENVK.......G ==

(33) P..F.LENVKGL.....G == ---------NA...GVPQ.RER..IVG

(34) P..F..ENVKG------- == GY......LNA.DYGV.Q.R.RV...G

(35) P..F..ENV.G------- == GY......LNA.DYGVPQ.R.RV.IVG

(36) -----LENV.GL.....G == GY......L.A...G.PQ.R.R...Vaflnqnihfe

(37) P..F..ENV.G------- == GY.......NA.DYGV.Q.R.RV...G

(38) P....LENV.GL------ == GY......L.A.DYG..Q.R.RV..Valkneytnff

```
(39) P..F.LENVK.L.....G == GY......L.A.D.G.PQ.RER...VG
(40) P..F.LENVK.L.....G == -Y........A.D.GVPQ.RER..IVG
(41) P....LENV..L------ == GY.......NA.DYG..Q.R.RV.I.G
(42) P..F..ENV.GL.....G == GY......LN.....V.Q.RER..I.G
(43) P..F.LENVK.L.....G == -Y........A.D.G.PQ.RER..IVG
(44) P.....ENVK.L------ == GY......LNA.D.G.PQ.RERV
(45) P.....ENV.GL------ == GY.......N....GVPQ.RERV.I
(46) stekndlNV..L------ == GY......LNA.D.G..Q.R.RV
```

Figure Result-2: Cytosine-specific DNA methylases. 46 protein sequences, some of which were upto 500 residues long, have been aligned with an *alignment number* $k = 18$. The number of maximal motifs for this is $N = 810$ and the size of the resulting feasible set of motifs is 132.

```
( 1)         altekqeal...s.e..k.n.p..s.......i.e.ap..k..fsflkdsn
( 2)  mstlegrgFTe.QEALV..S....K.N.......F...I.E.AP.A..lFsfLkdsn
( 3)  sssevnkvFTe.QEALV.......K.N.......F...I.E.AP.Ak.lFs.Lkdsp
( 4) msssevdkvFTe.QEALV..S....K.N.......F...I.E.AP.Ak.lFs.Lkdsp
( 5)       gvlt..q.alv.ss.e.f..n.p.....f....le.ap.ak.lfsflkgsse
( 6)        msFT.KQEALV.sS.E.FK.N.p..s..Fy...lEkAPaAk.lFsfLknsa
( 7)        msFT.KQEALV.sS.E.FK.N....s..Fy..IlEkAPaAk.lFsfLkdsa
( 8)         gFT.KQEALV.sS.Ef----k.n.p..s..fy...lekapaak.lfsflk
( 9)        gaFTeKQEALV.sS.E.FK.N.p..s..Fy..IlEkAPaAk.lFsfLangv
(10)         gFTeKQEALV.sS...FK.N....s..Fy..Il.kAP.Ak..FsfLkdsa
(11)       galte.q.alv.ss.e.f..n.p.....f....le.apaak.lfsflkgtse
(12)        mgFTeKQEALV.sS.E.FK.N.p..s..fy..ilekapaak..fsflkdta
(13)         gFTeKQEALV..S.E.FK.N.p..s..Fy..IlEkAPaAk..FsfLkdsd
(14)        gaFTeKQEALV.sS.E.FK.N.p..s..Fy..IlEkAPaAk.lFsfLsngv
(15)        mgFT.KQEALV.sS.E.FK.N.p..s..fy..ilekapaak..fsflkdsa
(16)         gFTeKQEALV..S.E.FK.N.p..s..Fy..IlEkAPaAk..FsfLkdfd
(17)        gaFT.KQEALV.sS.E.FK.N.p..s..Fy..IlEkAP.Ak.lFsfLangv
(18)        mgFT..QEALV.sS.E.FK.N.p..s..Fy..IlEkAPaAk..FsfLkdsa
```

```
(19)         gaFTeKQEALV.sS.E.FK.N.p..s..Fy..IlEkAPaAk.lFsfLangv
(20)         vaFTeKQ.ALV.sS.E.FK.N.p..s..Fy..IlEkAPaAk.lFsfLangv
(21)        mggFTeKQEALV.sS.E.FK.N.p..s..Fy..IlEkAPaAk.lF.fLangv


( 1) eipen----NPkL..HA...F.....sA..Lr..G..vwdn==LG.iH......DP
( 2) vpler----NPkL..HA..vF.....sA.qLr..G.vtvr==kLG..H...GV.D-
( 3) vpleq----NPkL..HA..vF.....sA.qLr..Gkatv==sdlg.ih...gv...
( 4) ipleq----NPkL..HA..vF.....sA.qLr..G==.snlkrLG.iH...GV---
( 5) vpqn-----NP.L..HA.kvF......A.qL...G.vasda==LG..H..KGV.D.
( 6) evqds-----p.l..haekvf..vrdsa.qlr..g.vv...atLG.iH..KGV.DP
( 7) gvqds-----p.l..haekvf..vrdsa.qlr..g.vv...aaLG.iH..KGV.DP
( 8) dtagveds--pkl..hae.vf..vrdsa.qlr..g.vv...atLG.iH..KGV..P
( 9) dpt------NPkL..HAek.F..vrdsA.qL...G.vv...a-L..iH..K.V.DP
(10) gvvds-----pkl..haekvf..vrdsa.qlr..g.vvldgkd+g.ih..kgv.dp
(11) vpqn-----NP.L..HA.kvF..v...A.qL...G.vvtda==LG..H..KGV.D.
(12) gvqd----s-pkl..haekvf..vrdsa.qlr..g.vv...atLG.iH..KGV.DP
(13) gvpqn----NP.L..HAekvF..vrdsA.qLr..G.vv...asLG..H..KGV.DP
(14) dps------NPkL..HAek.F..vrdsA.qL...G.vv...a-LG.iH..K...DP
(15) gvqd----s-pkl..haekvf..vrdsa.qlr..g.vv...atLG.iH..KGV.DP
(16) evpqn----NP.L..HAekvF..vrdsA.qLr..G.vv...asLG..H..KGV.DP
(17) dpt------NPkL..HAek.F..vrdsA.qL...G.vv...a-LG.iH..K...DP
(18) gvqds-----p.l..haekvf..vrdsa.qlr..g.vv...atLG.iH..KGV.DP
(19) dpt------NPkL..HAe..F..vrdsA.qLr..G.vv...a-LG.iH..KGV...
(20) dpt------NPkL..HAek.F..vrdsA.qL...G.vv...a-LG..H..K.V.DP
(21) dpt------NPkL..HAek.F..v.dsA.qLr..G.vv...a-LG..H..KGV.DP


( 1) hF.V.K.ALL.TIKEA....WS.E...AW..AY..L...IK..mke
( 2) hF.V.K.ALL.TIKEA....WS.E...AW..AYD.L..aIK..mkpss
( 3) hf.v...all.tikea....ws.e...aw.vayd.l..aik..mkpsst
```

```
( 4) hFetrf-all.tikea....ws.e...aw..ayd.l..aik..mkpsst
( 5) hF.VVKEA.LKTIKE..G..WS.EL..AW..AYD.LA..IKK.mkdaa
( 6) hF.VVKEALLKTIKEA.G..WS.EL..AW.vAYD.LA.aIKK.ms
( 7) hF.VVKEALLKTIKEA.G..WS.EL..AW.vAYD.LA..IKK.ms
( 8) hF.VVKEALL.TIK.A.G..WS.EL..AW.vAYD.LA.aIKK.mkta
( 9) .F.VVKEALLKTIKEA.G..WS.EL..AW.vAYD.LA.aIKKa
(10) hf.vvkeallktikea.g..ws.el..aw.vayd.la.aikaa
(11) hF.VVKEA.LKTIKE..G..WS.EL..AW..AYD.LA..IKK.mndaa
(12) hF.VVKEALLKTIKE..G..WS.EL..AW.vAYD.LA.aIKK.mg
(13) hF.VVKEALLKT.KEA.G..WS.E...AW.vAYD.L..aIKK.ms
(14) .F.VVKEALLKTIKEA.G..WS.EL..AW.vAYD.LA.aIKKaf
(15) hF.VVKEALLKTIKE..G..WS.EL..AW.vAYD.LA.aIKK.mv
(16) hF.VVKEALLKT.KEA.G..WS.E...AW.vAYD.L..aIKK.ms
(17) .F.VVKEALLKTIKEA.G..WS.EL..AW.vAYD.LA.aIKKaf
(18) hF.VVKEALLKTIKEA.G..WS.EL...W.vAYD.LA.aIKK.ms
(19) .F.VVKEALLKT.K.A.G..W...L..A...AYD.LA.aIKKaya
(20) .F.VVKEALLKTIK.A.G..WS.EL..AW.vAYD.LA.aIKKa
(21) .F.VVKEALLKT.KEA.G..WS.EL..AW.vAY..LA.a.KKaf
```

Figure Result-3: 21 protein sequences with an average size of 350 residues, have been aligned with an *alignment number* $k = 14$. The number of non-redundant motifs for this is $N = 873$ and the size of the resulting compatible set of motifs is 42.

10.3 Summary

We have proposed a two-stage approach to the alignment problem by handling two relatively simpler sub-problems which deal separately with the two issues, one of identifying the "local similarities" and the other of aligning the similarities appropriately. In the first stage we identify *all possible* K-wise motifs, i.e., all motifs that appear simultaneously in at least K of the N input sequences ($2 \leq K \leq N$). In the second stage, we give plausible alignments of a carefully chosen subset of

these motifs (that optimize certain cost functions). Using this approach for the alignment helps in at least two ways: (1) it aids in a direct N-wise alignment, as opposed to composing the alignments from lower order (say pairwise) alignments and (2) the resulting alignment is independent of the order of the input sequences. K is an input parameter and is called the *alignment number*. In practice, our approach works particularly well for alignment of a large set of (long) sequences. We have presented the result of running our alignment algorithm on biological data and the results look very promising.

Appendix A

The Exclusive BFC (EBFC) Problem

We give the details of steps 1 and 2 of Theorem 1 here.

Step 1: We make the following observations regarding the relation between a solution to the BMC problem and a solution to the MC problem.

[1.1] In a solution of the BMC, the two sets $\tilde{S}_1, \tilde{S}_2$, are such that $V_{gadget_i} \subset \tilde{S}_1$ or $\tilde{S}_2$, $\forall i$.

If this does not hold, that is $V_{gadget_i} \not\subset \tilde{S}_1$ then the solution can be modified as follows that only improves the solution. By the construction, $|V_{gadget_i}| = 2(d_i + 1)$. Thus $|V_{gadget_i} \cap \tilde{S}_1|$ or $|V_{gadget_i} \cap \tilde{S}_2| \geq d_i + 1$. Without loss of generality, let, $|V_{gadget_i} \cap \tilde{S}_2| \geq d_i + 1$. If $v_{ki} \in S_1, k \neq 0$, then by including v_{ki} in S_2 the cost increases by 2. If $v_{i0} \in S_1$, then it has *exactly* d_i negative edges incident on it with the other ends being in S_2 while *at most* d_i positive edges incident on it with the other ends in S_2. Hence by including v_{i0} in S_2 the "cut" does not suffer a loss.

[1.2] All the edges that contribute to a solution to the BMC have positive weights, since, by observation 1.1, all the negative weight edges must be either in S_1 or in S_2.

[1.3] In a solution to the BMC, if $v_1'v_2''$ is in the cut, so must $v_1''v_2'$ (called the *image* of $v_1'v_2''$). This follows from observation 1.1, as V_{gadget_1} and V_{gadget_2} are in the sets $\tilde{S}_1$ and $\tilde{S}_2$ respectively (without loss of generality).

[1.4] Given a solution to the BMC, the solution to the corresponding MC is constructed as follows: if $v_1'v_2''$ (and its image) is in the solution to the BMC, then v_1v_2 is in the solution to the MC.

Claim (C1.1): MC has solution of size K iff BMC has solution of size $2K$.

Proof: Let the MC have a solution of size K. Assume the BMC has a solution of $2(K+x)$, for some $x > 0$. Let the x edges be $v'_{i_10}v''_{j_10}, v'_{i_20}v''_{j_20}, \ldots, v'_{i_x0}v''_{j_x0}$ and their images (see observations 1.2 and 1.3). Then the solution to MC can be the edges corresponding to K, augmented by $v_{i_1}v_{j_1,i_2}v_{j_2}, \ldots, v_{i_x}v_{j_x}$, thus giving a solution of size $K+x$ to the MC problem, which is a contradiction.

Let the BMC have a solution of size $2K$. Assume the MC has a solution of size $K+x$, for some $x > 0$. Let the x edges be $v_{i_1}v_{j_1}, v_{i_2}v_{j_2}, \ldots, v_{i_x}v_{j_x}$ in the solution of the MC. Now, the solution of the BMC, can be augmented by $v'_{i_10}v''_{j_10}, v'_{i_20}v''_{j_20}, \ldots, v'_{i_x0}v''_{j_x0}$ and their images (see observation 1.3), giving a solution of size $2(K+x)$ for the BMC, which is a contradiction. QED

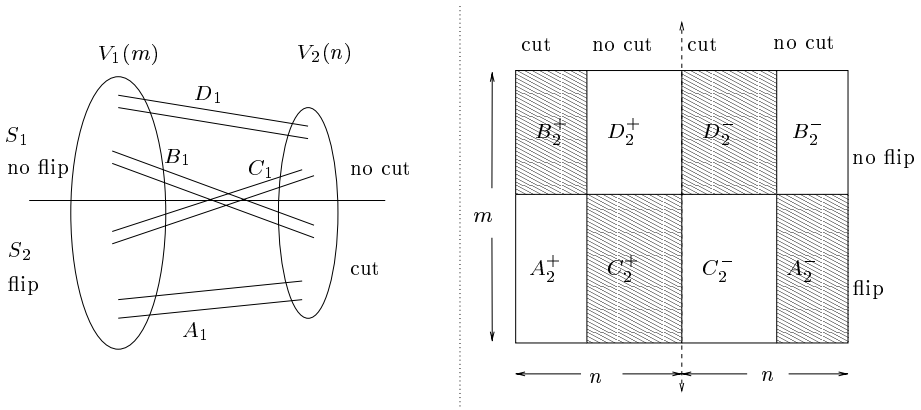
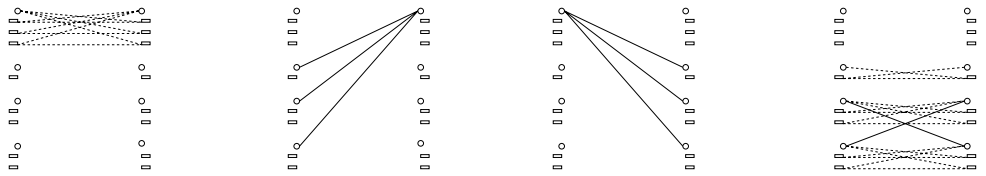


Figure A.1: A schematic representation of the BMC to EBFC reduction: The left shows the BMC problem and the right shows the EBFC problem. See the text on the reduction for other details.

	v''_{1*}	v''_{2*}	v''_{3*}	v''_{4*}
v'_{10}	$A^+A^+A^+A^+$	$\mathbf{C}^+C^+$	$\mathbf{C}^+C^+C^+$	$\mathbf{C}^+C^+C^+$
v'_{11}	$A^+A^+A^+A^+$	C^+C^+	$C^+C^+C^+$	$C^+C^+C^+$
v'_{12}	$A^+A^+A^+A^+$	C^+C^+	$C^+C^+C^+$	$C^+C^+C^+$
v'_{13}	$A^+A^+A^+A^+$	C^+C^+	$C^+C^+C^+$	$C^+C^+C^+$
v'_{20}	$\mathbf{B}^+B^+B^+B^+$	D^+D^+	$D^+D^+D^+$	$D^+D^+D^+$
v'_{21}	$B^+B^+B^+B^+$	D^+D^+	$D^+D^+D^+$	$D^+D^+D^+$
v'_{30}	$\mathbf{B}^+B^+B^+B^+$	D^+D^+	$D^+D^+D^+$	$\mathbf{D}^+D^+D^+$
v'_{31}	$B^+B^+B^+B^+$	D^+D^+	$D^+D^+D^+$	$D^+D^+D^+$
v'_{32}	$B^+B^+B^+B^+$	D^+D^+	$D^+D^+D^+$	$D^+D^+D^+$
v'_{40}	$\mathbf{B}^+B^+B^+B^+$	D^+D^+	$\mathbf{D}^+D^+D^+$	$D^+D^+D^+$
v'_{41}	$B^+B^+B^+B^+$	D^+D^+	$D^+D^+D^+$	$D^+D^+D^+$
v'_{42}	$B^+B^+B^+B^+$	D^+D^+	$D^+D^+D^+$	$D^+D^+D^+$

	$\overline{v''_{4*}}$	$\overline{v''_{3*}}$	$\overline{v''_{2*}}$	$\overline{v''_{1*}}$
v'_{10}	$C^-C^-C^-$	$C^-C^-C^-$	C^-C^-	$\mathbf{A}^-\mathbf{A}^-\mathbf{A}^-A^-$
v'_{11}	$C^-C^-C^-$	$C^-C^-C^-$	C^-C^-	$\mathbf{A}^-A^-\mathbf{A}^-A^-$
v'_{12}	$C^-C^-C^-$	$C^-C^-C^-$	C^-C^-	$\mathbf{A}^-\mathbf{A}^-A^-A^-$
v'_{13}	$C^-C^-C^-$	$C^-C^-C^-$	C^-C^-	$\mathbf{A}^-A^-A^-A^-$
v'_{20}	$D^-D^-D^-$	$D^-D^-D^-$	$\mathbf{D}^-D^-$	$B^-B^-B^-B^-$
v'_{21}	$D^-D^-D^-$	$D^-D^-D^-$	$\mathbf{D}^-D^-$	$B^-B^-B^-B^-$
v'_{30}	$D^-D^-D^-$	$\mathbf{D}^-\mathbf{D}^-D^-$	D^-D^-	$B^-B^-B^-B^-$
v'_{31}	$D^-D^-D^-$	$\mathbf{D}^-\mathbf{D}^-D^-$	D^-D^-	$B^-B^-B^-B^-$
v'_{32}	$D^-D^-D^-$	$\mathbf{D}^-D^-D^-$	D^-D^-	$B^-B^-B^-B^-$
v'_{40}	$\mathbf{D}^-\mathbf{D}^-D^-$	$D^-D^-D^-$	D^-D^-	$B^-B^-B^-B^-$
v'_{41}	$\mathbf{D}^-\mathbf{D}^-D^-$	$D^-D^-D^-$	D^-D^-	$B^-B^-B^-B^-$
v'_{42}	$\mathbf{D}^-D^-D^-$	$D^-D^-D^-$	D^-D^-	$B^-B^-B^-B^-$

Figure A.2: The grouping of the elements of the EBFC matrix due to a solution (not necessarily optimal) shown in Figure 5.2.



A edges (empty A^+ & dashed edges in A^-) | B edges (empty B^- & solid edges in B^+) | C edges (empty C^- & solid edges in C^+) | D edges (solid edges in D^+ & dashed edges in D^-)

Figure A.3: The corresponding edges due to the grouping shown in Figure A.2.

Observations: We make the following observations that relate a solution to the EBFC problem with that of the BMC problem: we describe a way to construct a solution to the BMC problem given a solution to the EBFC problem and secondly we also give a relation between the costs of the two solutions (equation (A.6)).
[2.1] In a given alignment of the EBFC, the elements of M_{ij} can be grouped into the following sets, as shown in Figure A.1:

$$A_2^+ = \{(i,j) | i \text{ has been flipped, } j \text{ is a cut, } j \leq n\}, A_2^- = \{(i,\overline{j}) | (i,j) \in A_2^+\},$$
$$B_2^+ = \{(i,j) | i \text{ has not been flipped, } j \text{ is a cut, } j \leq n\}, B_2^- = \{(i,\overline{j}) | (i,j) \in B_2^+\},$$
$$C_2^+ = \{(i,j) | i \text{ has been flipped, } j \text{ is not a cut, } j \leq n\}, C_2^- = \{(i,\overline{j}) | (i,j) \in C_2^+\},$$
$$D_2^+ = \{(i,j) | i \text{ has not been flipped, } j \text{ is not a cut, } j \leq n\}, D_2^- = \{(i,\overline{j}) | (i,j) \in D_2^+\}.$$

Let $\sum A_2^+ = \sum_{(i,j) \in A_2^+} M_{ij}$. Similarly define $\sum A_2^-$, $\sum B_2^+$, $\sum B_2^-$, $\sum C_2^+$, $\sum C_2^-$, $\sum D_2^+$, $\sum D_2^-$. Figure A.2 shows an illustrative example.

Recall that the cost in the EBFC is the number of 1's in the cut columns with the rows flipped appropriately. Thus the cost is $\sum A_2^- + \sum B_2^+ + \sum C_2^+ + \sum D_2^-$ (corresponding to the shaded rectangular region shown in Figure A.1).
[2.2]

$$\sum A_2^- + \sum B_2^- + \sum C_2^- + \sum D_2^- = \sum_{i=1}^{m} \sum_{j=n+1}^{2n} M_{ij} = e^-. \tag{A.1}$$

[2.3] Given an alignment for the EBFC with cost C_{EBFC} as

$$C_{EBFC} = \sum A_2^- + \sum B_2^+ + \sum C_2^+ + \sum D_2^-, \tag{A.2}$$

(see observation 2.1), a solution for the BMC is constructed as follows. Define the sets as below:

$$A_1^+ = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in A_2^+\}, A_1^- = \{v_i^1 v_j^2 | M_{i\bar{j}} \neq 0, (i,j) \in A_2^-\},$$
$$B_1^+ = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in B_2^+\}, B_1^- = \{v_i^1 v_j^2 | M_{i\bar{j}} \neq 0, (i,j) \in B_2^-\},$$
$$C_1^+ = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in C_2^+\}, C_1^- = \{v_i^1 v_j^2 | M_{i\bar{j}} \neq 0, (i,j) \in C_2^-\},$$
$$D_1^+ = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in D_2^+\}, D_1^- = \{v_i^1 v_j^2 | M_{i\bar{j}} \neq 0, (i,j) \in D_2^-\}.$$

Let $A_1 = A_1^+ \cup A_1^-$, $B_1 = B_1^+ \cup B_1^-$, $C_1 = C_1^+ \cup C_1^-$, $D_1 = D_1^+ \cup D_1^-$. Then S_1 and S_2, the partition of the vertices, are defined as follows:

$$S_1 = \{v_i^1 | v_i^1 v_j^2 \in B_1\} \cup \{v_j^2 | v_i^1 v_j^2 \in C_1\} \cup \{v_i^1, v_j^2 | v_i^1 v_j^2 \in D_1\}, \tag{A.3}$$
$$S_2 = \{v_j^2 | v_i^1 v_j^2 \in B_1\} \cup \{v_i^1 | v_i^1 v_j^2 \in C_1\} \cup \{v_i^1, v_j^2 | v_i^1 v_j^2 \in A_1\}. \tag{A.4}$$

Notice that $|A_1^+| = \sum A_2^+$, $|A_1^-| = \sum A_2^-$ and so on. Also notice that B_1^+ is the set of edges with positive weights and B_1^- is the set of edges with negative weights. Similarly for the other sets. Thus the corresponding cost, C_{BMC} for the BMC is,

$$C_{BMC} = |B_1^+| - |B_1^-| + |C_1^+| - |C_1^-|. \tag{A.5}$$

[2.4] It can be seen from the above that given a partition of the vertices in the BMC, an alignment (assignments of flips/no-flips to rows and cuts/no-cuts to columns) can be obtained for the EBFC, and, vice-versa.

[2.5] If C_{EBFC} denotes the cost for an alignment in the EBFC problem, and, if C_{BMC} denotes the cost for the corresponding alignment in the BMC problem, the following holds (using equations (A.1), (A.2), (A.5)):

$$C_{EBFC} - e^- = C_{BMC}. \tag{A.6}$$

Recall that e^- is the number of edges with negative weights in the BMC problem.

Claim (C2.1): BMC has an optimal solution of size K iff EBFC has an optimal solution of size $K + e^-$.

Proof: It can be verified from the above construction that, improving the solution for the EBFC by $x > 0$, results in improving the BMC by x and vice-versa. Hence using equation (A.6) we have the required result. QED

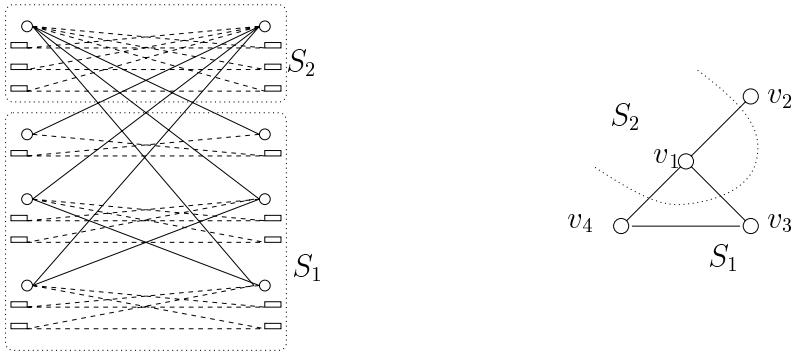


Figure A.4: The solution to the BMC problem obtained from Figure A.2 using equations (A.3), (A.4) and the corresponding solution in the MC problem, introduced in Figure 5.1. S_1 and S_2 are the partition of the vertices in the graphs.

We make the following observations about the solution to an EBFC, that has been constructed from a BMC that was in turn constructed from an MC problem.
[3.1] The EBFC matrix is of size $L \times 2L$ where $L = v + 2e$. Recall that v is the number of vertices and e the number of edges for the MC problem.
[3.2] See Figure (A.5) which shows the rows and columns associated with a vertex v_i of the MC problem in the EBFC matrix. Let $X_{v_{i0}}$ denote the variable associated with the row that corresponds to the vertex v'_{i0}, and, Y_{v_i0} denote the variable associated with the column that corresponds to the vertex v''_{i0} of the BMC. Similarly $X_{v_{i0}}, X_{v_{i1}}, X_{v_{i2}}, \ldots, X_{v_{id_i}}, Y_{v_{i1}}, Y_{v_{i2}}, \ldots, Y_{v_{id_i}}$.

The reader can verify (see Figure (A.5)) that one can always obtain a solution for which the following holds: $X_{v_{i0}} = X_{v_{i1}} = X_{v_{i2}} = \ldots = X_{v_{id_i}} = Y_{v_{i1}} = Y_{v_{i2}} = \ldots = Y_{v_{id_i}}$. This is important since it implies that in the BMC, $\tilde{V}_{gadget_i} \subset \tilde{S}_1$ or $\tilde{S}_2$. Thus the corresponding solution of size $2K$ in the BMC corresponds directly to a solution of size K of the MC. Thus for any solution of the EBFC, all the 1's corresponding to the negative edges are counted in the solution. If not, the

		v'_{i0}	v'_{i1}	v'_{i2}	v'_{i3}			$\overline{v''_{i3}}$	$\overline{v''_{i2}}$	$\overline{v''_{i1}}$	$\overline{v''_{i0}}$	
		$\vdots$	$\vdots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$	
		0	0	0	0			0	0	0	0	
		1	0	0	0			0	0	0	0	
		0	0	0	0			0	0	0	0	
		$\vdots$	$\vdots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$	
		0	0	0	0			0	0	0	0	
		1	0	0	0			0	0	0	0	
		0	0	0	0			0	0	0	0	
		$\vdots$	$\vdots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$	
v'_{i0}	$\ldots 010 \ldots 010 \ldots$	0	0	0	0	$\ldots 010 \ldots$	$\ldots 0 \ldots$	**1**	**1**	**1**	0	$\ldots 0 \ldots 0 \ldots$
v'_{i1}	$\ldots 000 \ldots 000 \ldots$	0	0	0	0	$\ldots 000 \ldots$	$\ldots 0 \ldots$	**1**	0	**1**	0	$\ldots 0 \ldots 0 \ldots$
v'_{i2}	$\ldots 000 \ldots 000 \ldots$	0	0	0	0	$\ldots 000 \ldots$	$\ldots 0 \ldots$	**1**	**1**	0	0	$\ldots 0 \ldots 0 \ldots$
v'_{i3}	$\ldots 000 \ldots 000 \ldots$	0	0	0	0	$\ldots 000 \ldots$	$\ldots 0 \ldots$	**1**	0	0	0	$\ldots 0 \ldots 0 \ldots$
		$\vdots$	$\vdots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$	
		0	0	0	0			0	0	0	0	
		1	0	0	0			0	0	0	0	
		0	0	0	0			0	0	0	0	
		$\vdots$	$\vdots$	$\vdots$	$\vdots$			$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Figure A.5: For clarity of exposition, let $d_i = 3$ in the MC problem. Abusing notation, let the conjugates of the columns corresponding to $v''_{i0}, v''_{i1}, v''_{i2}, v''_{i3}$ be $\overline{v_{i0}}'', \overline{v_{i1}}'', \overline{v_{i2}}'', \overline{v_{i3}}''$ respectively. The above table shows the rows and the columns corresponding to v_i of the MC. Note that the number of 1's in the column v'_{i0} is 3 and the number of 1's in the row v'_{i0} is $3 + 3$, where, half of the 1's are due to positive weight edges (left half of matrix) and the other half due to the negative weight edges (right half of matrix). The 1's shown in bold correspond to the negative weight edges on the vertices of the gadget corresponding to v_i of the MC problem. Notice that $v'_{i1}, v'_{i2}, v'_{i3}$ have the same flip as v'_{i0} without any conflict with other vertices. Also, since the column due to v''_{i0} has no 1's at all, v''_{i0} is a cut if v'_{i0} is flipped and is not a cut if v'_{i0} is not flipped.

solution can be altered, in linear time, without reducing the cost so that this condition holds. Thus

$$C_{EBFC} \geq 2e, \tag{A.7}$$

where e is the number of edges of the MC.

Appendix B

The Binary Shift Cut (BSC) Problem

We give the details of steps 1 and 2 of Theorem 5 here.

Step 1. MC to BMC reduction (see Figure B.1).

Consider an MC problem with vertices and edges $(V, E), n = |V|, e = |E|$. Let a solution be of size K inducing a partition of the vertices.

Reduction: Construct an instance of BMC with $(\tilde{V}, \tilde{E})$ as follows For each $v_i \in V$, with degree d_i, construct $3d_i + 6$ vertices, v_0^{i-}, v_0^{i*}, v_0^i, v_0^{i+}, v_0^{l-}, v_0^{l+}, $v_0^{i'}$, u_0^i, u_l^i, $l = 1, 2, \ldots, d_i$. Thus the total number of vertices are $6e + 6n$.

The edges have $+1$ or -1 weights which are constructed as shown in the table (Again see Figure B.1).

Edges with weights $+1$			Edges with weights -1		
edges $(l = 1, 2, \ldots, d_i)$	count per vertex	total count	edges $(l = 1, 2, \ldots, d_i)$	count per vertex	total count
$u_0^i v_0^l$, $u_0^l v_0^i$	d_i	$2e$	$u_0^i v_0^{l*}$, $u_0^l v_0^{i*}$	d_i	$2e$
$u_0^i v_l^{i+}$	d_i	$2e$	$u_0^i v_l^{i-}$	d_i	$2e$
$v_0^{i+} u_l^i$	d_i	$2e$	$v_0^{i-} u_l^i$	d_i	$2e$
$u_l^i v_l^{i+}$	d_i	$2e$	$u_l^i v_l^{i-}$	d_i	$2e$
$u_0^i v_0^{i+}$	2	$2n$	$u_0^i v_0^{i-}$	2	$2n$

Thus the number of edges with weight $+1$ is the same as the number with weight -1 which is $8e+2n$. Further, it can be seen that this construction gives a bipartite graph with $\tilde{V} = V' \cup V''$ where $v_x^y \in V', u_x^y \in V''$.

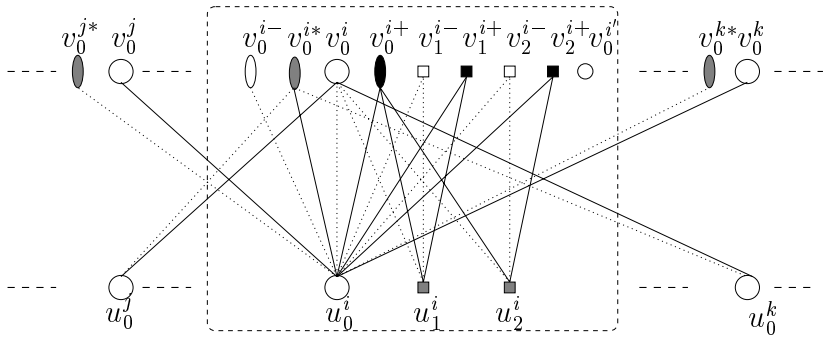


Figure B.1: The MC to BMC reduction: Let the degree of the vertex numbered i in the MC be 2; its neighbors are the vertices numbered j and k. Here we show the "gadget" that is constructed for the vertex numbered i (in the dotted rectangle). The hollow circles correspond to the two copies of the vertex i of the MC problem, u_i^0 and v_i^0. The solid lines denote edges with weight $+1$ and the dotted lines denote edges with weight -1.

Observations: We make the following observations. Let

$$\begin{aligned} V_i^{in} &= \{v_0^{i-}, v_0^i, v_1^{i-}, v_2^{i-}, \ldots, v_{d_i}^{i-}, u_0^i, u_1^i, u_2^i, \ldots, u_{d_i}^i\}, \\ V_i^{out} &= \{v_0^{i+}, v_1^{i+}, v_2^{i+}, \ldots, v_{d_i}^{i+}\}, \\ V_i^{any} &= \{v_0^{i*}, v_0^i, u_0^i\}. \end{aligned}$$

[1.1] It can be verified that in a solution of the BMC, the two sets S_1, S_2, are such that if $V_i^{in} \subset S_1$, then $V_i^{out} \subset S_2$ (or vice-versa). Further, if this does not hold, the solution can be modified, without decreasing the cost, so that the above condition holds.

[1.2] Thus, we get a partition, that contains both v_0^i and u_0^i; we use this to construct

the solution for the MC. The partition corresponding to the MC is the one that of the BMC, where we replace the set V_i^{in} by v_i (and, removing V_i^{out} and $\{v_0^{i*}\}$). Thus if $u_0^j v_0^i$ is a cut, then so is $v_0^j u_0^i$. Thus if K is the solution to the MC, then the solution to the BMC is at least $2K$.

[1.3] The cost due to the vertices $V_i^{in} \cup V_i^{out} - \{v_0^i, u_0^i\}$ in a gadget is $3d_i + 2$, for every vertex, as can be verified from the construction. Notice that we are able to make such a claim since these vertices have only "local" connectivity.

Thus the total cost due to the gadgets in all the vertices is $C_{gadget} = 6e + 2n$.

[1.4] Once we have a solution, and condition of observation 1.1 is satisfied, we can modify it to move the vertex v_0^{i*} around as follows. Using observation 1.2, the neighbors of every vertex v_i (of the MC) can be partitioned into two sets. Let $V_i^{in} \in S_1$. Let C_i denote the neighbors of v_i in the MC that are in S_2, and $c_i = |C_i|$. Then $d_i - c_i \leq c_i$, where d_i is the degree of the vertex v_i in the MC problem. If not, V_i^{in} can be moved to S_2, that can only increase the cost. Thus the cost due to the vertex v_0^{i*} is $1 - (d_i - c_i)$ (recall that there is a positive edge between u_0^i and v_0^{i*}, hence the 1). If K is the solution to the maxcut, note that $2K = \sum_i c_i$. Thus the total contribution due to $v_0^{l*}, l = 1, 2, \ldots, n$, is $\sum_l (1 - (d_l - c_l)) = n - 2e + 2K$.

Note that the cost for any solution $K \geq e/2$, and the cost excluding the gadgets, $C_{\overline{gadget}}$, is as follows:

$$\begin{aligned} C_{\overline{gadget}} &= n - 2e + 2K + 2K \\ &= n - 2e + 4K \\ &\geq 0 \end{aligned} \tag{B.1}$$

[1.4] Thus, noting that $K = C_{MC}$, we have,

$$\begin{aligned} C_{BMC} &= C_{gadget} + C_{\overline{gadget}} \\ &= (6e + 2n) - (n - 2e + 4K) \\ \Rightarrow \quad K &= C_{MC} \\ &= \tfrac{C_{BMC} - 4e - 3n}{4} \end{aligned} \tag{B.2}$$

using observations 1.2, 1.3 and 1.4.

Claim (C1.1): MC has solution K iff BMC has solution $4K + 4e + 3n$.

Proof: It can be verified from the above construction that, improving the solution for the BMC by $x > 0$, results in improving the MC by x and vice-versa. QED

Step 2. BMC to BSC reduction.

Let the incidence matrix of the BMC be $[\tilde{M}_{ij}]$. Define the matrix $[M_{ij}]$ for the BSC problem satisfying the invariance $\tilde{M}_{ij} = M_{ij} - M_{i(j-1)}, j > 1$.

Observations: We make the following observations.

[2.1] In a given alignment of the BSC, the elements of M_{ij} can be grouped into the following sets:

$$A_2^+ = \{(i,j)|i \text{ is right aligned, } j \text{ is a cut}\}, A_2^- = \{(i,j-1)|(i,j) \in A_2^+, j > 1\},$$
$$B_2^+ = \{(i,j)|i \text{ is left aligned, } j \text{ is a cut}\}, B_2^- = \{(i,j-1)|(i,j) \in B_2^+, j > 1\},$$
$$C_2^+ = \{(i,j)|i \text{ is right aligned, } j \text{ is not a cut}\}, C_2^- = \{(i,j-1)|(i,j) \in C_2^+, j > 1\},$$
$$D_2^+ = \{(i,j)|i \text{ is left aligned, } j \text{ is not a cut}\}, D_2^- = \{(i,j-1)|(i,j) \in D_2^+, j > 1\},$$

Note that $A_2^+ = C_2^-$, $A_2^- = C_2^+$, $B_2^+ = D_2^-$, $B_2^- = D_2^+$. Let $\sum A_2^+ = \sum_{(i,j)\in A_2^+} M_{ij}$. Similarly define $\sum A_2^-$, $\sum B_2^+$, $\sum B_2^-$, $\sum C_2^+$, $\sum C_2^-$, $\sum D_2^+$, $\sum D_2^-$.

We define the p_j's as follows:

$$p_{v_l^{i+}} = p_{v_l^{i-}} = 2/m, p_{v_0^{i-}} = 1/m, p_{v_0^{i+}} = d_i/m, p_{v_0^{i*}} = p_{v_0^i} = (d_i+1)/2m. \quad \text{(B.3)}$$

where $l = 1, 2, \ldots, d_i$, $\forall i$. This definition of p_j's ensures that for any two consecutive columns j and $j+1$, if by the alignment, the number of 1's in column $j+1$ is larger than in column j, then $Y_{j+1} = 1$, that is column $j+1$ is a consensus cut. This is similar to the EBFC problem where if column $\overline{j} = n - j + 1$ has larger number of 1's in the alignment than j then $Y_{\overline{j}} = 1$ and vice-versa.

[2.3] Given an alignment for the BSC problem, with cost C_{BSC} as

$$C_{BSC} = B_2^+ + C_2^-, \quad \text{(B.4)}$$

a solution for the BMC is constructed as follows. Define the sets as below:

$$A_1^- = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in A_2^+\}, A_1^+ = \{v_i^1 v_j^2 | (i,(j+1)) \in A_1^-\},$$
$$B_1^- = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in B_2^+\}, B_1^+ = \{v_i^1 v_j^2 | (i,(j+1)) \in B_1^-\},$$

	$\ldots$	v_0^{j-}	v_0^{j*}	v_0^{j}	v_0^{j+}	$\ldots$	v_0^{i-}	v_0^{i*}	v_0^{i}	v_0^{i+}	v_0^{1-}	v_0^{1+}	v_0^{2-}	v_0^{2+}	$v_0^{i'}$	$\ldots$	v_0^{k-}	v_0^{k*}	v_0^{k}	v_0^{k+}	$\ldots$

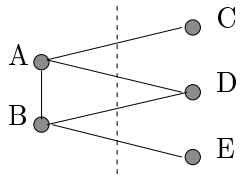
(BMC)

										$\vdots$											
u_0^j	$\ldots$	-1	$+1$	-1	$+1$	$\ldots$	0	-1	$+1$	0	0	0	0	0	0	$\ldots$	0	0	0	0	$\ldots$
										$\vdots$											
u_0^i	$\ldots$	0	-1	$+1$	0	$\ldots$	-1	$+1$	-1	$+1$	-1	$+1$	-1	$+1$	0	$\ldots$	0	-1	$+1$	0	$\ldots$
u_1^i	$\ldots$	0	0	0	0	$\ldots$	0	0	-1	$+1$	-1	$+1$	0	0	0	$\ldots$	0	0	0	0	$\ldots$
u_2^i	$\ldots$	0	0	0	0	$\ldots$	0	0	-1	$+1$	0	0	-1	$+1$	0	$\ldots$	0	0	0	0	$\ldots$
										$\vdots$											
u_0^k	$\ldots$	0	0	0	0	$\ldots$	0	-1	$+1$	0	0	0	0	0	0	$\ldots$	-1	$+1$	-1	$+1$	$\ldots$
										$\vdots$											

(BSC)

										$\vdots$											
u_0^j	$\ldots$	1	0	1	0	$\ldots$	0	1	0	0	0	0	0	0	0	$\ldots$	0	0	0	0	$\ldots$
										$\vdots$											
u_0^i	$\ldots$	0	1	0	0	$\ldots$	1	0	1	0	1	0	1	0	0	$\ldots$	0	1	0	0	$\ldots$
u_1^i	$\ldots$	0	0	0	0	$\ldots$	0	0	1	0	1	0	0	0	0	$\ldots$	0	0	0	0	$\ldots$
u_2^i	$\ldots$	0	0	0	0	$\ldots$	0	0	1	0	0	0	1	0	0	$\ldots$	0	0	0	0	$\ldots$
										$\vdots$											
u_0^k	$\ldots$	0	0	0	0	$\ldots$	0	1	0	0	0	0	0	0	0	$\ldots$	1	0	1	0	$\ldots$
										$\vdots$											

Figure B.2: Portions of incidence matrix of the BMC and the corresponding BSC matrix, for the graph shown in Figure B.1.



A	1010...0100...0100...0100...0000
B	0100...1010...0000...0100...0100
C	0100...0000...1010...0000...0000
D	0100...0100...0000...1010...0000
E	0000...0100...0000...0000...1010

A*	-101...0010...0010...0010...00000
B*	-010...0101...0000...0010...00100
C	0100...0000...1010...0000...0000-
D	0100...0100...0000...1010...0000-
E	0000...0100...0000...0000...1010-

Figure B.3: An example of MC to BSC reduction. Note that only the columns corresponding to vertices v_0^{i-}, v_0^{i*}, v_0^i, v_0^{i+} and rows corresponding to u_i^0, for each of the vertex A, B, C, D, E in the MC, is shown for the sake of clarity. (The complete example is described at the end of this section). The rows that are right aligned are marked by asterisk. Notice how the 1's in the columns align when the first 2 rows are right aligned.

$$C_1^- = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in C_2^+\}, C_1^+ = \{v_i^1 v_j^2 | (i,(j+1)) \in C_1^-\},$$
$$D_1^- = \{v_i^1 v_j^2 | M_{ij} \neq 0, (i,j) \in D_2^+\}, D_1^+ = \{v_i^1 v_j^2 | (i,(j+1)) \in D_1^-\},$$

Let $A_1 = A_1^+ \cup A_1^-$, $B_1 = B_1^+ \cup B_1^-$, $C_1 = C_1^+ \cup C_1^-$, $D_1 = D_1^+ \cup D_1^-$. Then S_1 and S_2, the partition of the vertices, are defined as follows:

$$S_1 = \{v_i^1 | v_i^1 v_j^2 \in B_1\} \cup \{v_j^2 | v_i^1 v_j^2 \in C_1\} \cup \{v_i^1, v_j^2 | v_i^1 v_j^2 \in D_1\},$$
$$S_2 = \{v_j^2 | v_i^1 v_j^2 \in B_1\} \cup \{v_i^1 | v_i^1 v_j^2 \in C_1\} \cup \{v_i^1, v_j^2 | v_i^1 v_j^2 \in A_1\}.$$

Notice that $|A_1^+| = \sum A_2^-$, $|A_1^-| = \sum A_2^+$ and so on. Also notice that B_1^- is the set of edges with positive weights and B_1^+ is the set of edges with negative weights. Similarly for the other sets. Thus the corresponding cost, C_{BMC} for the BMC is,

$$C_{BMC} = |B_1^-| - |B_1^+| + |C_1^-| - |C_1^+| = |C_1^+| - |C_1^+|. \tag{B.5}$$

since $|B_1^+| = |B_1^-|$ by the construction.

[2.4] It can be seen from the above that given a partition of the vertices in the BMC, an alignment (assignments of left/right-aligns to rows and cuts/no-cuts to columns) can be obtained for the BSC, and, vice-versa.

[2.5] Let C_{BSC} denote the cost for an alignment in the BSC problem, and, let C_{BMC} denote the cost for the corresponding alignment in the BMC problem. Further let $L = |\{(i,j)|i \text{ is left aligned}\}|$ and $R = |\{(i,j)|i \text{ is right aligned}\}|$. Then,

$$\begin{aligned} 2C_{BSC} - (L+R) &= (2B_2^+ - L) + (2C_2^- - R) \quad \text{using eqn(B.4)} \\ &= B_2^+ - B_2^- + C_2^- - C_2^+ \\ &= C_2^- - C_2^+ \end{aligned} \tag{B.6}$$

Thus,

$$2C_{BSC} - c = C_{BMC}. \tag{B.7}$$

where c $(= L + R = 8e + 2n)$ is the number of 1's in the BSC matrix.

Claim (C2.1): BSC has an optimal solution of size K iff BMC has an optimal solution of size $2K - c$.

Proof: It can be verified from the above construction that, improving the solution for the BSC by $x > 0$, results in improving the BMC by x and vice-versa.

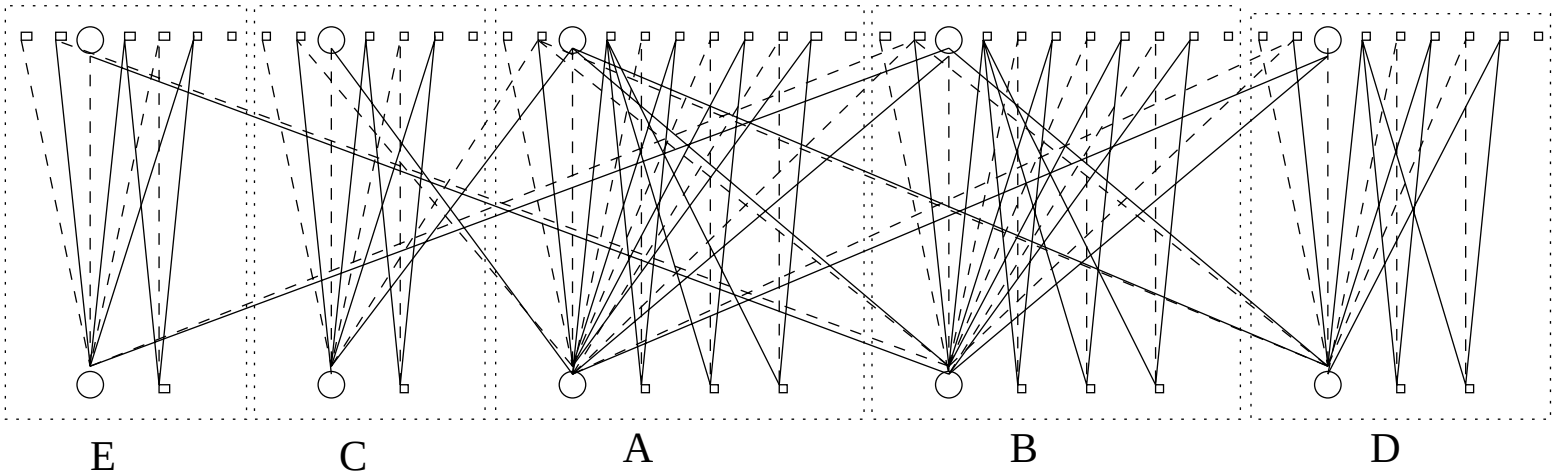
Also, it can be verified (see Figure B.3 and the Appendix for examples) that, given an arbitrary solution to BSC, it can be modified, without decreasing the cost, so that the following holds (let Y_x denote the indicator variable (cut or no-cut) associated with vertex x of the BMC):

$$Y_{v_0^{i*}} = Y_{v_0^{i+}} = Y_{v_l^{i+}},\ Y_{v_0^{i\prime}} = 0, Y_{v_0^i} = Y_{v_0^{i-}} = Y_{v_l^{i-}},\ \text{and},\ Y_{v_0^{i*}} \neq Y_{v_0^i}.$$

where $l = 1, 2, \ldots, d_i$, $\forall i$.

This concludes Step 2 of the proof. QED

Example of an instance of a BSC problem



B1. The BMC graph for the problem in Figure B.3. The 5 dotted boxes are the "gadgets" corresponding to the 5 vertices of the graph. The solid edges denote weight $+1$ and the dashed edges denote weight -1.

Example (continued)

A	B	C	D	E	
1 0 1 0 1 0 1 0 1 0 0	0 1 0 0 0 0 0 0 0 0 0	0 1 0 0 0 0 0	0 1 0 0 0 0 0 0 0	0 0 0 0 0 0 0	√
0 0 1 0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	√
0 0 1 0 0 0 1 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	√
0 0 1 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	√
0 1 0 0 0 0 0 0 0 0 0	1 0 1 0 1 0 1 0 1 0 0	0 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0	0 1 0 0 0 0 0	√
0 0 0 0 0 0 0 0 0 0 0	0 0 1 0 1 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	√
0 0 0 0 0 0 0 0 0 0 0	0 0 1 0 0 0 1 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	√
0 0 0 0 0 0 0 0 0 0 0	0 0 1 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	√
0 1 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	1 0 1 0 1 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	×
0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	0 0 1 0 1 0 0	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	×
0 1 0 0 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	1 0 1 0 1 0 1 0 0	0 0 0 0 0 0 0	×
0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 1 0 1 0 0 0 0	0 0 0 0 0 0 0	×
0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 1 0 0 0 1 0 0	0 0 0 0 0 0 0	×
0 0 0 0 0 0 0 0 0 0 0	0 1 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	1 0 1 0 1 0 0	×
0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0	0 0 1 0 1 0 0	×
×√×√×√×√×√×	×√×√×√×√×√×	√×√×√××	√×√×√×√××	√×√×√×××	S'
2 4 4 6 4 4 4 4 4 4 4	2 4 4 6 4 4 4 4 4 4 4	2 2 2 2 4 4 4	2 3 3 4 4 4 4 4 4	2 2 2 2 4 4 4	$30p_j$

B2. The BSC matrix is constructed from the graph shown in B1. The rows/molecules that are marked by √ denote the ones that are shifted to the right by one unit. The consensus map S' is shown at the bottom with √ denoting a consensus cut site and × its absence. The p_j for each column is shown in the bottom line: x denotes probability $x/30$ (note that the number of rows is 15).

Appendix C

Acronyms used in the thesis

The following is a list of acronyms used in the thesis along with a pointer to the most relevant section describing the term.

BFC	Binary Flip Cut	Section 5.2
$\text{BFC}_{\max}$	Variant of BFC	Section5.2
BMC	Bipartite Maximum Cut	Section 6.1
BPC	Binary Partition Cut	Section 8.1.1
$\text{BPC}_{\max}$	Variant of BPC	Section 8.1.1
BSC	Binary Shift Cut	Section 8.1.2
$\text{BSC}_{\max}$	Variant of BSC	Section 8.1.2
BSeC	Binary Sizing-error Cut	Section 8.1.3
$\text{BSeC}_{\max}$	Variant of BSeC	Section 8.1.3
CG	Consistency Graph	Section 5.3
dM	d-wise Match	Section 5.3
EBFC	Exclusive Binary Flip Cut	Section 5.2
FPTAS	Fully Polynomial Time Approximation Scheme	Section 5.1
GMC	Generalized Maximum Cut	Section 6.2
MC	Maximum Cut	Section 6.1
MSA	Multiple Sequence Alignment	Section 10
Kpop	K-Populations	Section 8.2.6
PTAS	Polynomial Time Approximation Scheme	Section 5.1
WCG	Weighted Consistency Graph	Section 5.3

Bibliography

[1] F. Alizadeh, R.M. Karp, D.K. Weisser, and G. Zweig. Physical mapping of chromosomes using unique probes. *Journal of Computational Biology*, 2(2):153–158, 1995.

[2] S. Altschul. Gap costs for multiple sequence alignment. *J. Theor. Biol.*, 138:297–309, 1989.

[3] T.S. Anantharaman, B. Mishra, and D.C. Schwartz. Genomics via optical mapping II: Ordered restriction maps. *Journal of Computational Biology*, 4(2):91–118, 1997.

[4] S. Arora, D. Karger, and M. Karpinski. Polynomial time approximation schemes for dense instances of NP-Hard problems. In *Proc. STOC*, 1996.

[5] S. Arora and C. Lund. Hardness of approximations. In D.S. Hochbaum, editor, *Approximation algorithms for NP-Hardness Problems*. PWS Publishing Company, MA, 1997.

[6] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proc. of the 33th IEEE symposium on the Foundations of Computer Science*, 1992.

[7] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of np. In *Proc. of the 33th IEEE symposium on the Foundations of Computer Science*, 1992.

[8] W. Cai, J. Jing, B. Irvine, L. Ohler, E. Rose, H. Shizua, U. J. Kim, M. Simon, T. Anantharaman, B. Mishra, and D. C. Schwartz. High-resolution restriction maps of bacterial artificial chromosomes constructed by optical mapping. *Proc. Natl. Acad. Sci. USA*, 95:3390–3395, 1998.

[9] H. Carrillo and D. Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics*, pages 1073–1082, 1988.

[10] K. M. Chao, R. Hardison, and W. Miller. Recent developments in linear-space alignment methods: A survey. *J. Computational Biology*, 3:271–291, 1994.

[11] S. Cook. The complexity of theorem-proving procedures. *Proc. of the 3rd Annual ACM Symposium on the Theory of Computing*, pages 151–158, 1971.

[12] N. G. Cooper, editor. *The Human Genome Project – Deciphering the Blueprint of Heredity.* University Science Books, Mill Valley, California, 1994.

[13] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, 1990.

[14] V. Dančík, S. Hannehalli, and S. Muthukrishnan. Hardness of flip-cut problems for optical mapping. *J. Computational Biology*, 4(2), 1997.

[15] D. Fasulo, T. Jiang, R. Karp, R. Settergren, and E. Thayer. An algorithmic approach to multiple complete digest mapping. In *Proceedings of the First Annual Conference on Computational Molecular Biology (RECOMB97)*, pages 118–127. ACM Press, 1997.

[16] Y. Gao, M. Yang, X. Wang, K. Mathee, and G. Narasimhan. Detection of HTH motifs via data mining. 1997.

[17] 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, 1979.

[18] D. Geiger and L. Parida. A model and solution to the DNA flipping string problem. Technical Report TR1996-720, Courant Inst. of Math. Sciences, New York University, May 1996.

[19] D. Geiger and L. Parida. Mass estimation of DNA molecules & extraction of ordered restriction maps in optical mapping imagery. *Algorithmica*, 1998. (in press).

[20] M. X. Goemans and D. P. Williamson. .878-approximation algorithms for MAX CUT and MAX 2SAT. In *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing*, pages 422–431, Montreal, Quebec, Canada, 23-25 May 1994.

[21] P.W. Goldberg, M.C. Golumbic, H. Kaplan, and R. Shamir. Four strikes against physical mapping of DNA. *Journal of Computational Biology*, 2(1):139–152, 1995.

[22] S. K. Gupta, J. Kececioglu, and A. A. Schaffer. Improving the practical space and time efficiency of the shortest-paths approach to sum-of-pairs multiple sequence alignment. *Journal of Computational Biology*, 2(3):459–472, 1995.

[23] J. Håstad. Some optimal inapproximability results. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 1–10, El Paso, Texas, 4-6 May 1997.

[24] D. Higgins and P. Sharpe. CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene*, 73:237–244, 1988.

[25] M. Hirosawa, Y. Totoki, M. Hoshida, and M. Ishikawa. Comprehensive study on iterative algorithms of multiple sequence alignment. volume 11(1), pages 13–18, 1995.

[26] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[27] R. Karp. Reducibality among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[28] R. Karp. Mapping the genome: Some combinatorial problems arising in molecular biology. In *Proc. STOC*, 1993.

[29] R. M. Karp and R. Shamir. Algorithms for optical mapping. In *Proceedings of the Annual Conference on Computational Molecular Biology (RECOMB98)*, pages 117–124. ACM Press, 1998.

[30] J. Kececioglu. The maximum weight trace problem in multiple sequence alignment. In *Proc. of the Fourth Symp. on Comp. Pattern Matching*, volume 684 of *Lecture Notes in Computer Science*, pages 106–119, Berlin, 1993. Springer-Verlag.

[31] M. Krawczak. Algorithms for restriction-site mapping of DNA molecules. *Proc. Nat. Acad. Sci.*, 85:7298–7301, 1988.

[32] J. K. Lee, V. Dančík, and M. S. Waterman. Estimation for restriction sites observed by optical mapping using reversible-jump Markov chain Monte Carlo. In *Proceedings of the Annual Conference on Computational Molecular Biology (RECOMB98)*, pages 147–152. ACM Press, 1998.

[33] L. Levin. Universal search problems (in russian). *Problemy Peredachi Informatsii*, 9(3):265–266, 1973.

[34] M. Levitt and C. Chottia. Structural patterns in globular proteins. *Nature*, 261:552–558, 1976.

[35] J. R. Griggs M. Waterman. Interval graphs and maps of DNA. *Bull. of Math. Biol.*, 48:189–195, 1986.

[36] X. Meng, K. Benson, K. Chada, E. J. Huff, and D. C. Schwartz. Optical mapping of lambda bacteriophage clones using restriction endonuclease. *Nature Genetics*, 9:432–438, 1995.

[37] B. Mishra. Some results based on a simple model for optical mapping. 1998. (unpublished manuscript).

[38] S. Muthukrishnan and L. Parida. Towards constructing physical maps by optical mapping: An effective simple combinatorial approach. In *Proceedings*

of the First Annual Conference on Computational Molecular Biology (RECOMB97), pages 209–215. ACM Press, 1997.

[39] G. Nemhauser and L. Wolsey. *Integer and Combinatorial Optimization*. Series in Discrete Math and Optimization. Wiley Interscience, 1988.

[40] C. Papadimitriou and M. Yannakakis. Optimization, approximation and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[41] L. Parida. Algorithmic complexity of physical mapping problems arising in single molecule methods. 1997.

[42] L. Parida. Computational molecular biology: Problems and tools. *Journal of the Indian Institute of Science*, 77:283–326, 1997.

[43] L. Parida. A uniform framework for ordered restriction map problems. *Journal of Computational Biology*, 1998. (in press).

[44] L. Parida, A. Floratos, and I. Rigoutsos. Musca: An algorithm for constrained alignment of multiple data sequences. 1998. (submitted for publication).

[45] L. Parida and B. Mishra. Partitioning k clones: Hardness results and practical algorithms for the *k-populations problem*. In *Proceedings of the Second Annual Conference on Computational Molecular Biology (RECOMB98)*, pages 192–201. ACM Press, 1998.

[46] P. Pevzner. DNA physical mapping. *Computer Analysis of Genetic Texts*, pages 154–158, 1990.

[47] P. Pevzner and A. Mironov. An efficient method for physical mapping of DNA molecules. *Molec. Bio.*, 21:788–796, 1987.

[48] P. Pevzner and M. Waterman. Open combinatorial problems in computational molecular biology. In *Proceedings of the Third Israel Symposium on Theory of Computing and Systems*, 1995.

[49] J. Reed. *Optical Mapping*. PhD thesis, Dept of Chemistry, New York University, June 1997.

[50] K. Reinert, H. P. Lenhof, P. Mutzel, K. Melhorn, and J. D. Kececioglu. A branch-and-cut algorithm for multiple sequence alignment. In *Proceedings of the First Annual Conference on Computational Molecular Biology (RECOMB97)*, pages 241–249. ACM Press, 1997.

[51] I. Rigoutsos and A. Floratos. Motif discovery in biological sequences without alignment or enumeration. In *Proceedings of the Annual Conference on Computational Molecular Biology (RECOMB98)*, pages 221–227. ACM Press, 1998.

[52] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[53] A. Samad, W. W. Cai, X. Hu, B. Irvin, J. Jing, J. Reed, X. Meng, J. Huang, E. Huff, B. Porter, A. Shenker, T. Anantharaman, B. Mishra, V. Clarke, E. Dimalanta, J. Edington, C. Hiort, R. Rabbah, J. Skiadas, and D. Schwartz. Mapping the genome one molecule at a time – optical mapping. *Nature*, 378:516–517, 1997.

[54] D.C. Schwartz, X. Li, L. I. Hernandez, S. P. Ramnarain, E. J. Huff, and Y. K. Wang. Ordered restriction maps of *saccharomyces cerevisiae* chromosomes constructed by optical mapping. *Science*, 262:110–114, 1993.

[55] D. Smith. Evolution of a vision. *Human Genome News*, 7(3-4):2, 1995.

[56] J. C. Venter, M. D. Adams, G. G. Sutton, A. R. Kerlavage, H. O. Smith, and M. Hunkapillar. Shotgun sequencing of the human genome. *Science*, 280:1540–1542, 1998.

[57] M. Vihinen. An algorithm for simultaneous comparison of several sequences. volume 4, pages 89–92, 1998.

[58] Y.K. Wang, E.J. Huff, and D.C. Schwartz. Optical mapping of the site-directed cleavages on single DNA molecules by the RecA-assisted restriction endonuclease technique. *Proc. Natl. Acad. Sci. USA*, 92:237–242, 1984.

[59] M.S. Waterman. Parametric and ensemble alignment algorithms. *Bulletin of Mathematical Biology*, 56(4):743–767, 1994.

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

[61] W. Miller Z. Zhang, B. He. Local multiple alignment vis subgraph enumeration. *Discrete Applied Mathematics*, 71:337–365, 1996.

Algorithmic Techniques in Computational Genomics

by

Laxmi Parida

Advisor: Bud Mishra

This thesis explores the application of algorithmic techniques in understanding and solving computational problems arising in Genomics. In the first part of the thesis we focus on the problem of reconstructing physical maps from data, related to "reading" the genome of an organism, and in the second part we focus on problems related to "interpreting" (in a very limited sense) the genome. The main contributions of the thesis are modeling, understanding the computational complexity of, and designing algorithms for some key problems in both these domains.

In the first part of the thesis, we focus on the problem of computing physical maps from data that arise in single molecule methods. We describe two combinatorial models of the problem termed Exclusive Binary Flip Cut (EBFC) and Weighted Consistency Graph (WCG) problems. We show that both the problems are MAX SNP hard and give bounds on the approximation factors achievable. We give polynomial time 0.878-approximation algorithm for the EBFC problem and 0.817-approximation algorithm for the WCG problem. We also give a low polynomial time practical algorithm that works well on simulated and real data. *Naksha* is an implementation of this algorithm and a demonstration is available at `http://www.cs.nyu.edu/parida/naksha.html`. We also have similar results on complexity for generalizations of the problem which model various other sources of errors. We have generalized our complexity and algorithmic results to the case where there is more than one population in the data (which we call the K-populations problem).

In the second part of the thesis, we focus on "interpreting" the genome. We consider the problem of discovering patterns/motifs in strings on a finite alphabet:

we show that by appropriately defining irredundant motifs, the number of irredundant motifs is only quadratic in the input size. We use these irredundant motifs in designing algorithms to align multiple genome or protein sequences. Alignment of sequences aids in comparing similarities, in structure and function of the proteins.