Università degli Studi di Milano Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea in Informatica

Integer Linear Programming approaches on the DNA recombination problem

Relatore: *Bonizzoni Paola*
Co-relatore: *Della Vedova Gianluca*

Relazione della prova finale di:
Antonio Vivace
Matricola 793509

Anno Accademico 2016–2017

Abstract

We introduce the *Computational Biology* field and familiarise with *Integer Linear Programming*, defining its inception, uses and approach, and how ILP-based approaches have become a standard optimization technique in bioinformatics, reviewing an application.

Then, we formalise the "DNA Recombination and Rearrangement" problem based on the what is observed in some species of ciliates, followed by an analysis and report of some of existent approaches and their central ideas, limitations and reductions applied.

Finally, an ILP formulation of the DNA Recombination problem is given, describing the implementative tools used and the main encountered difficulties.

Contents

1 — Introduction

1.1 Computational Biology

Computational Biology is defined as the development and application of data-analytical and theoretical methods, mathematical modeling and computational simulation techniques to the study of biological, behavioral, and social systems[1].

The field is now thirty years old and it's covered by many conferences and journals publishing papers. It features graph theory, network flows, combinatorics, integer and linear programming problems, statistical approaches, probabilistic methods, hidden Markov models, neural networks as its tools.

Some of the most important challenges are[2]:

- Protein structure prediction;
- Homology searches;
- Multiple alignment and phylogeny construction;
- Genomic sequence analysis and gene-finding.

In particular, *Computational Molecular Biology* (bioinformatics) focuses on studying existing and emerging approaches, techniques and algorithms for string computation (sequences) providing a significant intersection between computer science and molecular biology [3].

For these reasons, the field is inherently multidisciplinary: it's appealing to the Mathematical Programming and Operations Research community. Today, computational biology papers are written by computer scientists, biologists, statisticians, physicists and mathematicians, pure and applied.

Concretely, the application areas are [4]:

- Sequence analysis. Comparison of genomic sequences within individuals of a same species, or intra-species, in order to highlight their

differences and similarities. Reconstruction of long sequences by assembly of shorter sequence fragments. Error correction for sequencing machines.

- HYBRIDIZATION AND MICROARRAYS. Use of hybridization for sequencing. Use of microarrays for tissue identification, clustering and feature selection discriminating healthy from diseased samples. Design of optimal primers for PCR experiments. Physical mapping (ordering) of probes by hybridization experiments.

- PROTEIN STRUCTURES. Protein fold prediction from aminoacid sequence (ab-initio), or from sequence + other known structures (threading of sequences to structures). Alignment of RNA sequences depending on their structure. Protein fold comparison and alignment of protein structures. Study of protein docking and synthesis of molecules of given 3D structure.

- HAPLOTYPING. Reconstruction and/or correction of haplotypes from partial haplotype fragments or from genotype data. Analysis of resulting haplotypes and correlation with genetic diseases [5].

- EVOLUTION. Comparison of whole genomes to highlight evolutionary macro events (inversions, transpositions, translocations). Computation of evolutionary distances between genomes. Computation of common evolutionary subtrees or of evolutionary supertrees.

Note that the term *bioinformatics* is used also as an umbrella term for the (wider) body of biological studies using computer programming as part of their methodology, as well as a reference to specific analysis "pipelines" that are repeatedly used, particularly in the field of genomics.

2 — Integer Programming

2.1 Definition

Linear programming (ILP) is a technique for the mathematical optimization of a linear objective function, subject to linear equality and linear inequality constraints.

Linear programs are problems that can be expressed in canonical form as:

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \quad \text{(cost function)} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0} \\ & (\mathbf{x} \in \mathbb{Z}^n) \end{array}$$

If the variables are forcibly constrained to be integers, we call the program *Integer* or *Integer Linear* (ILP).

0-1 integer programming or binary integer programming (BIP) is the special case of integer programming where variables are required to be 0 or 1 ($\mathbf{x} \in \{0, 1\}$).

A particular case of integer linear programming is represented by Combinatorial Optimization (CO), that is the class of problems in which the feasible region is a subset of the vertices of the unit hypercube $F \subseteq \mathbf{B}^n = \{0, 1\}^n$, i.e., more simply, problems in which variables can only take value 0 or 1. Linear {0,1} (or binary) programming problems belong to this class [6].

In contrast to linear programming, which can be solved efficiently in the worst case, integer programming problems are in many practical situations (bounded variables) NP-hard. Binary Integer Programming problems are classified as NP-hard too ("0–1 integer programming" is one of the *Karp's 21 NP-complete problems*).

2.2 Algorithms

2.3 In Computational Biology

At its inception, the focus of Computational Biology was on the development of efficient algorithms and data structures that were able to deal with the data being introduced in life science applications. Lately, the introduction of high throughput methods for biomedical data analysis and the rise of Systems Biology (the study of systems of biological components) made Statistical Learning approaches a standard [7].

Furthermore, new and accessible sequencing methods caused an exponential growth of the available genomic data.

This element and the fact that biological processes are usually reduced and studied as simulations (because the actual nature of them is still being investigated, as in the case of our problem) lead to the introduction of a lot new optimization problems in the field.

In most cases, these optimisazion problems are discrete ones: hence the approval of ILP-based approaches as a standard.

Some of the most successful Integer Programming approaches for computational biology problems are described in [8].

2.3.1 Advantages

There are a number of additional reasons why ILP should be taken into consideration, even when the problems seems to not require it or the advantage of introducing an ILP formulation isn't initally clear[9]:

- Commercial ILP *solvers* are available (with academic licenses);
- The progress of those solvers has been spectacular: benchmark ILP problems can be solved *200-bilion* times faster than twenty-years ago;
- Even for a problem where a worst-case efficient general algorithm might be possible, the time and effort needed to find it, implement it as a computer program, is typically much greater than the time and effort needed to formulate and implement an ILP solution to the problem.
- Some problems can be modeled in a much more efficient way with ILP.
- A new mathematical formulation for classic problems may be studied, allowing the original one to be attacked in new ways. New techniques and relaxations can be applied.

To give a real example, the deeply studied Multiple Sequence Alignment problem [10] shows how many different approaches and formulations can be theorised and exploited: tt was reformulated as an optimisation problem introducing the concept of *trace* in [11], given branch-and-cut algorithms in [12] and relaxations, such as [13], which proposes a branch-and-bound algorithm with a Lagrangian relaxation.

Among others, [14] reduce the multiple alignment problem to the minimum routing cost tree (MRCT) problem, i.e., finding a spanning tree in a complete weighted graph, which minimizes the sum of the distances between each pair of nodes. They propose a Branch-and-Price algorithm for the MRCT problem and then use it. [15] reduce multiple sequence alignment to a facility location problem. The reduction is then used to provide a Polynomial Time Approximation Scheme for a certain class of multiple alignment problems.

2.3.2 A typical approach

Usually,

2.4 Design of an ILP formulation

2.4.1 Idioms

Here's how many logic expressions can be expressed as linear disequalities without side effects or uncovered cases [9].

Suppose L is an integer linear function of binary variables with M being its upper limit and b a positive integer. Many of these idioms can be reduced if some or all variables are binary, strictly positive, or bounded.

If-Then

$$L \geq b \rightarrow z$$

Linearly:

$$L - (M \times z) \leq b - 1$$

Only-If

$$z = 1 \text{ only if } L \geq b$$

Let s be the smallest value that L can achieve and set $m = s - b$. Linearly:

$$L + m \times z \geq m + b$$

These two idioms can be used as building blocks for many more:

NAND

Let L_1 and L_2 be linear functions whose variables are bounded, and $L_1 \geq b_1$ and $L_2 \geq b_2$. We require that at *most* one of the two linear inequalities is satisfied.

$$z_1 + z_2 \leq 1$$

Where $z_1 = 1$ if $L_1 \geq b_1$ and $z_2 = 1$ if $L_2 \geq b_2$. We use the *If-Then* twice idiom to express these two conditions.

OR

Here we require that at *least* one of the two linear inequalities is satisfied.

$$z_1 + z_2 \geq 1$$

Followed by two *Only-If* idioms to express $z_1 = 1$ *only* if $L_1 \geq b_1$ and $z_2 = 1$ *only* if $L_2 \geq b_2$.

XOR

If we want *exactly* one of the inequalities to be satisfied:

$$z_1 + z_2 = 1$$

Again, followed by two *Only-If* idioms to express $z_1 = 1$ *only* if $L_1 \geq b_1$ and $z_2 = 1$ *only* if $L_2 \geq b_2$ and two *If-Then* (if and only if).

Implied Satisfaction

To express

$$L_1 \geq b_1 \rightarrow L_2 \geq b_2$$

We need an *If-Then* idiom for the first equality, an *Only-If* idiom for the second and

$$z_1 \leq z_2$$

Not-Equal

Suppose Z_1 and Z_2 are linear functions of integer variables whose values are bounded. Then $Z_1 - Z_2$ and $Z_2 - Z_1$ are bounded to integer values, too. Then, we can express

$$Z_1 \neq Z_2$$

as

$$(Z_1 - Z_2 \geq 1) \text{ OR } (Z_2 - Z_1 \geq 1)$$

Using our *OR* idiom previously explained: let s_1 the lower bound for $Z_1 - Z_2$ and s_2 the lower bound for $Z_2 - Z_1$. Set $m_1 = s_1 - 1$ and $m_2 = s_2 - 1$. The final inequalities will be

$$(Z_1 - Z_2) + m_1 \times l_1 \geq m_1 + 1$$
$$(Z_2 - Z_1) + m_2 \times l_2 \geq m_2 + 1$$
$$l_1 + l_2 \geq 1$$

3 — The Problem

3.1 Biological Background

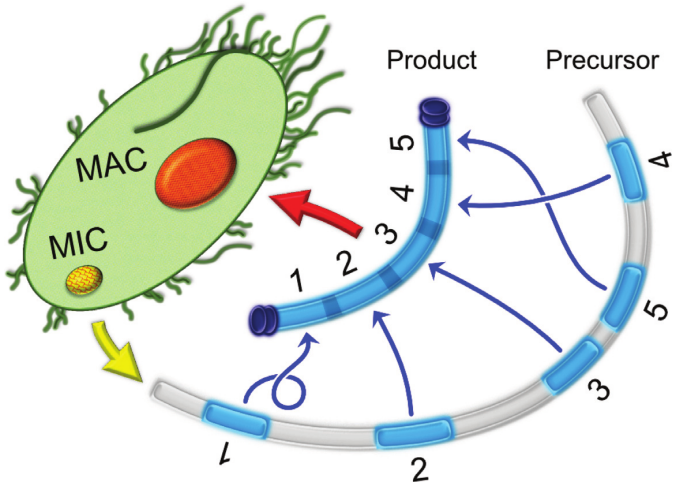

Figure 3.1: In the somatic macronucleus (MAC), chromosomes assemble from precursor MDS building blocks (blue), which may be scrambled in some species. In the germline micronucleus (MIC), the Macronuclear Destined Sequences (MDSs) for all somatic chromosomes are dispersed over the long chromosome, and interrupted by Internally-Eliminated Sequences (IESs) and other noncoding DNA (gray). In some cases, an MDS may appear in a permuted order, or inverted[16].

Ciliated protists (microbial eukaryotes using cilia for locomotion) exhibit nuclear dimorphism through the presence of separate germline and somatic nuclei. The somatic macronucleus (MAC) provides templates for the transcription of all genes required for asexual growth while the germline micronucleus (MIC) is used for the exchange of meiotic products during sexual reproduction [16]. The MAC DNA is the one actively expressed and effectively results in the phenotype of the organism.

Several species of ciliates, such as *Stylonychia* or *Oxytricha*, go through extensive gene rearrangement while differentiating somatic macronuclei from

germline micronuclei. This process entails an extensive fragmentation, elimination and sometimes broader rearrangement of the germline DNA, coupled to DNA amplification and telomere addition [17] and form the somatic macronuclei, all under the epigenetic control of novel non-coding RNA pathways [18]. The extent and the nature of these operations varies among ciliate species.

Each gene in the macronucleus may be present in the micronucleus as several nonconsecutive segments (macronuclear destined sequences, **MDSs**) separated by non-coding DNA. During macronuclear differentiation, the non-coding fragments (internal eliminated sequences, **IESs**) that interrupt MDSs in the micronucleus are deleted. Moreover, the order of the MDSs in the micronucleus may not be consecutive, in which case formation of the macronucleus requires unscrambling of the MDS order, as well as IES removal. There exist **pointer**-like sequences that are repeated at the end of the nth MDS and at the beginning of the $(n + 1)$st MDS in the micronucleus. Each pointer sequence is retained as only one copy in the sequence in the macronucleus [19].

The general RNA-guided mechanism that regulate and lead this process of assembly is not known, theoretical investigations can be found in [20] and [21].

3.2 Biological Motivation

The guided genome rearrangement problem has (and it's) been extensively [21] studied, both as biochemical process and mathematical model, as it provides an exaggerated case of a phenomenon observed among different species in different ways [22]. Similar broad scale, somatic rearrangement events occur in many eukaryotic cells and tumors.

Many discrete and topological models, mathematical approaches, biological and biochemical explanations and speculations on the theme can be found in literature (such as [23] [24] [19] and [18]).

3.3 Formalisation

The recognised events in the rearrangement process are:

- The MAC begins a copy of the MIC DNA. The chromosomes are fragmented and amplified. The result of this process is the *precursor*. $\sim$90% of the complexity is lost.

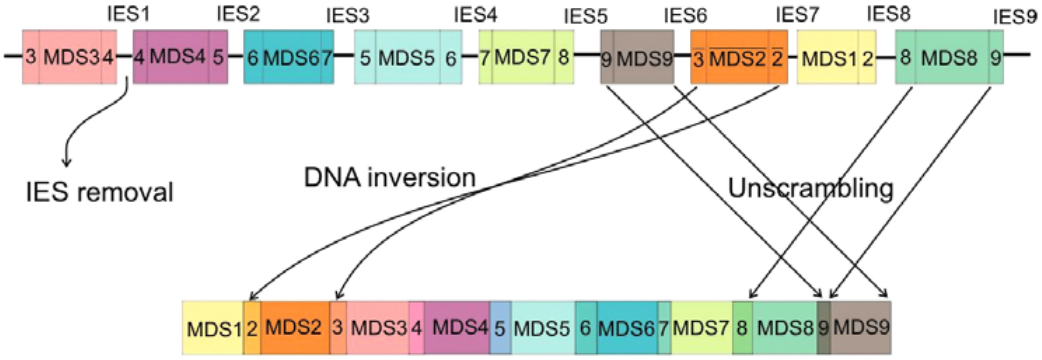

Figure 3.2: Schematic representation of the scrambled Actin I micronuclear germline gene in Oxytricha nova (top) and the correctly assembled macronuclear gene (bottom). Each block represents an MDS, and each line between blocks is an IES. The numbers at the beginning and at the end of each segment represent the pointer sequences. Note that MDS3-MDS8 require permutation and inversion to assemble into the orthodox, linear order MDS1-MDS9 in the macronucleus. The bars above MDS2 and its pointers indicate that this block is inverted relative to the others, i.e., this sequence is the Watson - Crick reverse complement of the version in the macronucleus; from[25].

  - Fragmentation
  - Amplification

- From the precursor the final MAC DNA is produced through these further operations:
  - Elimination
  - Inversion
  - Gene Scrambling - Unscrambling
  - Telomere Addition

We focus on the second phase, trying to map the following "building blocks":

- **MDSs**, the contiguous sequences copied, inverted or (order) scrambled in the MAC;
- **IESs**, sections not present in the MAC;
- **Pointers**, overlap sections between MDSs in the MAC (maybe inverted), present in multiple copies in the MIC;

With "inverse", "reverse", "reverse complement" terms we refer to the *Watson-Crick reverse complement* of the sequence.

The goal is to produce a *rearrangement map*: a set of disjoint substrings representing the building blocks, eventual operations they will go through the process (scrambling, inversion) and their "destination" on the produced genome.

3.3.1 Instance

3.4 Existent Approaches

3.5 Formulation

3.5.1 Reduced Instance

3.5.2 ILP

objective function: $min \sum_{i,j} MDS_{MACstart}(i,j)$

3.5.3 Variables definitions

$$*Eq(i,j,h,l) = \begin{cases} 0 \\ 1, & \text{if } \texttt{MIC[i:j]} = \texttt{MAC[h:l]} \end{cases}$$

$$*cwc(i,j,h,l) = \begin{cases} 0 \\ 1, & \text{if } \texttt{MIC[i:j]} \text{ is the reverse complement of } \texttt{MAC[h:l]} \end{cases}$$

$$*Possible_{MDSMAC}(i,a,b) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ can start at } a \text{ and finish at } b \text{ in the MAC} \end{cases}$$

$$*Possible_{MDSMIC}(i,a,b) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ can start at } a \text{ and finish at } b \text{ in the MIC} \end{cases}$$

$$Possible_{assignment}(a,b,c,d) = Eq(a,b,c,d)$$

$$MDS_{MICstart}(i,j) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ starts at position } j \text{ in the MIC} \end{cases}$$

$$MDS_{MICend}(i,j) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ ends at position } j \text{ in the MIC} \end{cases}$$

$$MDS_{MACstart}(i,j) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ starts at position } j \text{ in the MAC} \end{cases}$$

$$MDS_{MACend}(i,j) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ ends at position } j \text{ in the MAC} \end{cases}$$

$$Inv(i) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ is inverted in the MAC} \end{cases}$$

$$P_{start}(i,j) = \begin{cases} 0 \\ 1, & \text{if } MDS_{MACstart}(i,j) = 1, \text{ Pointer } i \text{ starts at position } j \text{ in the MAC} \end{cases}$$

$$P_{end}(i,j) = \begin{cases} 0 \\ 1, & \text{if } MDS_{MACend}(i-1,j) = 1, \text{ Pointer } i \text{ ends at position } j \text{ in the MAC} \end{cases}$$

$$*MAC(i,c) = \begin{cases} 0 \\ 1, & \text{if } c \text{ is the character at position } i \text{ in the MAC} \end{cases}$$

$$*MIC(i,c) = \begin{cases} 0 \\ 1, & \text{if } c \text{ is the character at position } i \text{ in the MIC} \end{cases}$$

$$Cov_{MIC}(i,j) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ covers the position } j \text{ in the MIC} \end{cases}$$

$$Cov_{MAC}(i,j) = \begin{cases} 0 \\ 1, & \text{if MDS } i \text{ covers the position } j \text{ in the MAC} \end{cases}$$

3.5.4 Constraints

¿ Internally Eliminated Sequences

$$IES(j) = \begin{cases} 0 \\ 1, & \text{if } i \text{ is part of an IES: } \sum_{0 \le i \le q} Cov_{MIC}(i,j) = 0 \end{cases}$$

¿ MDSs must correspond to identical or reverse and complemented substrings of MIC and MAC. The following constraints enforce this fact:

$MDS_{MICstart}(i,a)+MDS_{MICend}(i,b)+MDS_{MACstart}(i,c)+MDS_{MACend}(i,d)+Inv(i)-5cwc(a,b,c,d)=0$

$MDS_{MICstart}(i,a)+MDS_{MICend}(i,b)+MDS_{MACstart}(i,c)+MDS_{MACend}(i,d)-4Eq(a,b,c,d)=Inv(i)$

$$\sum_j MDS_{MICstart}(i,j) \le 1$$

$$\sum_j MDS_{MICend}(i,j) = \sum_j MDS_{MICstart}(i,j)$$

¿ Coverage

$$\sum_{l \leq j} MDS_{MICstart}(i,l) + \sum_{l \geq j} MDS_{MICend}(i,l) - 2Cov_{MIC}(i,j) = 0$$
$$\sum_{l \leq j} MDS_{MACstart}(i,l) + \sum_{l \geq j} MDS_{MACend}(i,l) - 2Cov_{MAC}(i,j) = 0$$

3.5.5 Preprocessing

Bibliography

[1] NIH Biomedical Information Science and Technology Initiative Consortium. NIH working definition of bioinformatics and computational biology. `https://web.archive.org/web/20120905155331/http://www.bisti.nih.gov/docs/CompuBioDef.pdf`, 2000.

[2] David B. Searls. Chapter 1 - grand challenges in computational biology. In David B. Searls Steven L. Salzberg and Simon Kasif, editors, *Computational Methods in Molecular Biology*, volume 32 of *New Comprehensive Biochemistry*, pages 3 – 10. Elsevier, 1998.

[3] Dan Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, New York, NY, USA, 1997.

[4] Giuseppe Lancia. Mathematical programming in computational biology: an annotated bibliography. *Algorithms*, 1(2):100–129, 2008.

[5] Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, and Jing Li. The haplotyping problem: An overview of computational models and solutions. *Journal of Computer Science and Technology*, 18(6):675–688, Nov 2003.

[6] Federico Malucelli. "Introduction to Operation Research - Integer Linear Programming".

[7] Ernst Althaus, Gunnar W. Klau, Oliver Kohlbacher, Hans-Peter Lenhof, Knut Reinert. Integer Linear Programming in Computational Biology. In: Lecture Notes in CS 5760.

[8] Giuseppe Lancia. Integer programming models for computational biology problems. *Journal of Computer Science and Technology*, 19(1), 2004.

[9] Dan Gusfield. Integer linear programming in computational biology tutorial. In *Integer Linear Programming in Computational Biology: An entry-level course for biologists (and other friends)*. Cambridge Press, 2018.

[10] Humberto Carrillo and David Lipman. The multiple sequence alignment problem in biology. *SIAM Journal on Applied Mathematics*, 48(5):1073–1082, 1988.

[11] John Kececioglu. *The maximum weight trace problem in multiple sequence alignment*, pages 106–119. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993.

[12] John D. Kececioglu, Hans-Peter Lenhof, Kurt Mehlhorn, Petra Mutzel, Knut Reinert, and Martin Vingron. A polyhedral approach to sequence alignment problems. *Discrete Applied Mathematics*, 104(1):143 – 186, 2000.

[13] Ernst Althaus and Stefan Canzar. *A Lagrangian Relaxation Approach for the Multiple Sequence Alignment Problem*, pages 267–278. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[14] Matteo Fischetti, Giuseppe Lancia, and Paolo Serafini. Exact algorithms for minimum routing cost trees. *Networks*, 39(3):161–173, 2002.

[15] Winfried Just and Gianluca Della Vedova. Multiple sequence alignment as a facility location problem. In *Proceedings of the Prague Stringology Club Workshop 2000, Prague, Czech Republic, September 2, 2000*, pages 60–70. Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University, 2000.

[16] Jonathan Burns, Denys Kukushkin, Kelsi Lindblad, Xiao Chen, Nataša Jonoska, and Laura F. Landweber. mds ies db: a database of ciliate genome rearrangements. *Nucleic Acids Research*, 44(D1):D703–D709, 2016.

[17] D.M. Prescott. *The DNA of Ciliated Protozoa. Microbiol.* 1994.

[18] Talya Yerlici and Laura F Landweber. Programmed genome rearrangements in the ciliate oxytricha. 2, 12 2014.

[19] Angela Angeleska, Nataša Jonoska, Masahico Saito, and Laura F. Landweber. Rna-guided dna assembly. *Journal of Theoretical Biology*, 248(4):706 – 720, 2007.

[20] Robert Brijder, Hendrik Jan Hoogeboom, and Grzegorz Rozenberg. *From Micro to Macro: How the Overlap Graph Determines the Reduction Graph in Ciliates*, pages 149–160. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[21] Andrzej Ehrenfeucht, Tero Harju, and Ion Petre. *Computation in Living Cells: Gene Assembly in Ciliates (Natural Computing Series)*. SpringerVerlag, 2004.

[22] Angela Angeleska, Nataša Jonoska, and Masahico Saito. Dna recombination through assembly graphs. *Discrete Applied Mathematics*, 157(14):3020 – 3037, 2009.

[23] David M. Prescott. Genome gymnastics: Unique modes of dna evolution and processing in ciliates. 1:191–8, 01 2001.

[24] Robert Brijder and Hendrik Jan Hoogeboom. *The Algebra of Gene Assembly in Ciliates*, pages 289–307. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[25] D.M. Prescott, A.F. Greslin. Scrambled actin I gene in the micronucleus of Oxytricha nova. *Developmental Genetics*, (13):66–74, 1992.