Eötvös Loránd University

Institute of Mathematics

Combinatorics on words and its applications

Péter Ligeti

Doctoral thesis

Doctoral school: Mathematics

Director: Professor Miklós Laczkovich

Doctoral program: Applied mathematics

Director: Professor András Prékopa

Supervisor: Péter Sziklai

Associate professor, CSc.

Department of Computer Science

Acknowledgements

Before all, I am most grateful to Péter Sziklai for introducing me the field of mathematical research and the world of *words*. His guidance and stimulating supervision was invaluable. I promise him my undying gratitude.

I would like to thank all my coauthors for the fruitful scientific work. Furthermore, I am indebted to István Miklós for checking the correctness of the biological parts, to Marci Makai for helping me with latex and to the Rényi Institute for the encourageing and inspiring atmosphere.

Last, but not least, my infinite gratitude is due to my family, especially my son Ábel for the silent nights and my wife Eszter for all.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Motivation and notation

Combinatorics on words is a relatively new research area of discrete mathematics partly inspired by problems in theoretical computer science, as dealing with formal languages and automata theory and other fields of mathematics, such as number theory, group theory and probability theory.

The beginning of systematic research concerning the topic is devoted to the pioneering work of Axel Thue who started to work on repetition-free words and proved, among others, the existence of an infinite square-free word over an alphabet of three elements.

Until the eighties, words were used in various areas of mathematics but only as tools to achieve other goals. The first comprehensive presentation of results and methods related to the field was the book "Combinatorics on words" of Lothaire [40].

The book, which collected the current developments of the topic, was followed by others. The recent one [41] is devoted to the applications of combinatorics on words only. It covers several areas of science, such as natural languages, bioinformatics, applied and pure mathematics.

Now we review the most fundamental **notion and notation** used through the thesis. For more details, see the corresponding chapters.

Let Σ be a finite set of t elements. Σ is called the *alphabet* and its elements are called the *letters*. For the sake of simplicity we will use the alphabet $\Sigma = \{0, 1, \ldots, t-1\}$. A finite sequence composed of the elements of Σ is a *word*, i.e. $w = w_1 \ldots w_n$ such that $w_i \in \Sigma$ for all i. Here we notice that all the structures examined in the thesis are finite. For a word $w = w_1 \ldots w_n$ let n be the *length* of w, it will be denoted by $|w|$. Let Σ^n denote the set of all words of length n over the alphabet Σ and let $\Sigma^* = \bigcup_{n \in \mathbb{N}} \Sigma^n$. A (not necessarily consecutive) subsequence u of w is called a *subword*, which will be denoted by $u \leq w$. The consecutive subsequences are called *factors*.

Let $|w|_i$ denote the maximal number of occurrences of the letter i in the word w. More generally for $w \in \Sigma^*$ and $G \subseteq \Sigma$, let w_G denote the maximal subword of w containing the letters of G only. A word is *homogeneous* if all of its letters are the same. Maximal homogeneous consecutive subwords are called *runs*. (So e.g. 01111001 has four runs.)

Let the ordered pair $\mathcal{P} = (\mathcal{A}, <)$ be a *partial ordered set* or a *poset* shortly where $\mathcal{A}$ is the ground set of $\mathcal{P}$ and $<$ is the partial order of $\mathcal{P}$. $a, b \in \mathcal{P}$ are *comparable* if $a < b$ or $b < a$.

If $a < b$, but there is no $x \in \mathcal{P}$ such that $a < x < b$ for $a, b \in \mathcal{P}$, then a is called a *child* of b, and b is called a *parent* of a. Furthermore we say that a_1 and a_2 are *brothers* if they have a common parent.

A poset $\mathcal{P}$ is called *graded* or *ranked* if all of its maximal chains (are finite and) have the same length. Every graded poset has a *rank function* $\mathbf{r} : \mathcal{P} \to \mathbb{Z}$ such that if a is a minimal element, then $\mathbf{r}(a) = 0$, furthermore, if b is a parent of a, then $\mathbf{r}(b) = \mathbf{r}(a) + 1$. Let $\mathcal{P}_l$ denote the elements of rank l in the poset $\mathcal{P}$, called the *l-th level*.

Let $\mathcal{B}$ be a family of elements of rank l in a poset $\mathcal{P}$, i.e. $\mathcal{B} \subseteq \mathcal{P}_l$. Let $\Delta_k \mathcal{B}$ denote for $0 \leq k \leq l$ the family of elements of rank k being comparable to at least one element of $\mathcal{B}$, the *k-shadow* of $\mathcal{B}$. For $k = l-1$ we simply say *shadow* and denote it by $\Delta \mathcal{B}$. Similarly, for $l \leq k$ the *k-shade* $\nabla^k \mathcal{B}$ is the family of the elements of rank k being comparable to at least one element of $\mathcal{B}$. For $k = l+1$ we simply say *shade* and denote it by $\nabla \mathcal{B}$.

In the case of $\mathcal{B} = \{a\}$, let $\Delta_k \mathcal{B}$ be called the *k-deck* of a. Less formally speaking the k-deck of the element a is the family of elements of rank k being comparable to it.

Let $\operatorname{Aut}(\mathcal{P})$ denote the automorphism group of the poset $\mathcal{P}$.

All logarithms are of natural base through the thesis, except when it is denoted.

In mathematics, there is a notable number of problems that deal with reconstruction either of an object by some incomplete information about it or of a whole by its parts. One can find such problems in various topics, such as the reconstruction of a function by its values at some points, the reconstruction of a group by its subgroups, pattern recognition or decision making by partial observations or fragmentary information, information with missed values, error-correcting codes, etc. Many problems of discrete mathematics can be reduced to this scenario, furthermore it is often difficult to determine whether the missing information is important and whether it is possible to restore it using the available data.

In a substantial part of this thesis we will examine different generalizations and applications of the following reconstruction problem:

Basic problem *Let the length n of a word and an alphabet Σ be given. Determine the smallest k such that every word $w \in \Sigma^n$ can be reconstructed from the k-deck of its subwords.*

In the literature two versions of this problem are studied concerning the elements of the k-deck:

i. reconstruction from the *k-deck consisting of the multiset of the $\binom{n}{k}$ subwords* of the original word of length n;

ii. reconstruction from the *k-deck consisting of the set of different subwords* of the original word of length n.

The problem was examined first by Kalashnik [28] who handled the case of reconstruction from the k-deck consisting of the multiset of $\binom{n}{k}$ subwords. In his paper the author is related to the information-theoretic study of noisy deletion channels in which characters of a transmitted sequence are randomly (but not necessarily independently) omitted. The examined problem addresses the variant in which, out of n original characters, $n-k$ are chosen uniformly at random and deleted; the problem amounts to characterizing the greatest loss rate at which it is possible to determine the message using unlimited repeated sampling. This is similar to asking whether ancestral genomes can be inferred from modern specimens, although the genetic process is more complicated, than this mathematical model, since in the real world not only deletions occur but insertions and substitutions as well; furthermore, the number of deleteted genes is not constant.

The problem is partially motivated by a non-standard direction of coding theory, the *insertion-deletion codes.* Codes with an upper bound on the length of common subwords of two codewords are called insertion-deletion codes, which are introduced by Levenshtein [35] for the correction of synchronization errors. Contrary to the conventional coding theory, the length of the sent and received messages are different, hence the classical results are not applicable. As an analogue of the Hamming-distance, a new metric called *Levenshtein-distance* was introduced as the minimum number of insertions and deletions of letters needed to transform a word into another. In the paper some constructions capable correcting single deletions are given via ordered Steiner systems.

One way to generalize the problem is to consider *DNA-words* and their *substrands* instead of words and subwords. This variant of the problem is motivated by the basic properties of the DNA-strand: one can build an exact mathematical model of DNA-words, which can be handled well, by emphasizing the presence of complement pairs, and some structural properties of the Watson-Crick double helix (i.e. double-strandedness and reverse complementation).

D'yachkov et al. [19] introduced a new family of insertion-deletion codes, called *DNA codes*, based upon the structure of the DNA-strand. A DNA code is a set of codeword strands over the alphabet $\{\{A, T\}, \{C, G\}\}$ with the following properties:

- no codeword strand equals to its reverse complement
- the reverse complement of every codeword strand is a codeword strand as well
- the length of a common subword of any two distinct codeword strands of length n is at most k

Then a DNA code defined above is capable to correct $n-k$ deletions.

Another way of generalization is to increase the "dimension" of the problem. This leads to reconstruction of square matrices from their square submatrices which question is not well-studied in contrast to the case of words, except the lonely paper of Manvel and Stockmeyer [43]. Note that there are two natural variants of this problem, i.e. the case of deleting some rows and columns of the original matrix symmetrically, or arbitrarily. Beyond its theoretical interest this problem has a connection to the famous graph reconstruction problem of Kelly [31] and Ulam [60]: it is equivalent with the special case of ordered graphs or ordered bipartite graphs, respectively.

The results of the original problem cannot be generalized trivially in neither case. The reason of the difficulties is the greater complexity of the structures.

The algorithmic aspect of combinatorics on words is very important from the practical point of view as well. In pure mathematics, computer science, bioinformatics and other areas of science there is a wide range of applications related to this topic, such as language processing, inference of network expressions, DNA sequencing, pattern matching, etc. For more on the applications and algorithms on combinatorics on words see the third book of Lothaire [41].

1.2 Overview of the thesis

Here we outline the main results presented in the thesis.

In Chapter 2 we examine the most simple version of reconstruction from different subwords in the case of DNA-words and matrices: the reconstruction from the $n-1$-deck. Contrary to the case of graph-reconstruction, the presented problems can be solved easily, these are only tools for proving other interesting statements: as an application of this kind of results we determine the automorphism groups of different posets: the first one arising from all DNA-words of length at most n, partially ordered by the substrand relation, the second one arising from all square matrices of order at most n partially ordered by the submatrix relation. There are two obvious types of automorphisms in both cases:

DNA-words: the one induced by a permutation on the complement pairs, and the ones induced by a map which interchanges the elements of a given complement pair

Matrices: the one induced by a permutation on the alphabet, and the congruence group of the square

We will see that for most cases (i.e. for $n \geq 3$) there are no more automorphisms. At last as one more illustration of the method we will give a significantly shorter proof for a theorem of Burosh, Gronau and Laborde [12] which determines the automorphism group of a poset consisting of all subwords of the word which has maximum number of different subwords among the words of given length n and over given alphabet. The results of Chapter 2 are based on a joint paper with Péter Sziklai [L4] and on a manuscript [L3].

Simon [53] and Lothaire [40] proved that every word $w \in \Sigma^n$ can be reconstructed from its $\lceil \frac{n+1}{2} \rceil$-deck consisting of its different subwords which result is sharp for binary alphabet. In Chapter 3 we give an improvement of this result for a general alphabet and with lower and upper bounds for the number of occurrences of letters in the words. The results of Chapter 3 are based on a joint paper with Péter Sziklai [L5].

In Chapter 4 we examine the reconstruction of DNA-words from the k-deck consisting of the set of its different substrands. We prove the analogue of the result of Theorem 10 for DNA-words, i.e. every DNA-word of length n can be reconstructed from its $\lfloor \frac{2(n+1)}{3} \rfloor$-deck. The significant part of this chapter is devoted to this proof which has two main stages. In the first one we prove a bit stronger result when the alphabet consists of one complement pair, and in the second part we prove the general statement when the alphabet consists of two complement pairs. In addition we give some further related simple results. The results of Chapter 4 based are on a joint paper with Péter L. Erdős, Péter Sziklai and David C. Torney [L1].

In Chapter 5 we consider the other generalization of the basic problem, i.e. the reconstruction of square matrices from the k-deck consisting of the multiset of its square submatrices and one more special problem, the reconstruction from the matrix consisting the *sum* of the elements of the k-deck. Clearly matrices with the same k-decks have the same sum-matrix but the other direction is not trivial. We examine the cases of symmetric and non-symmetric deletions as well, and prove lower and upper bounds in both cases. (The reconstruction problem from the sum-matrix is motivated by the proof of the lower bound.)

In the first part of this chapter using simple combinatorial and linear algebraic arguments we prove that the following is a necessary condition for the reconstruction problem of matrices (this condition is sufficient in the case of reconstruction from the sum of the the elements of the k-deck only): if two square matrices of order n have different k-decks consisting of the multiset of their square submatrices, then there exists a polynomial $p(x, y)$ with real coefficients such that $\deg p < k$ and two subsets $H_1, H_2 \subset \{1, 2, \ldots, n\}^2$, such that

$$\sum_{(x,y)\in H_1} p(x, y) \neq \sum_{(x,y)\in H_2} p(x, y).$$

The second part of the chapter is devoted to the construction of such a polynomial using Chebishev polynomials and some geometrical arguments.

The main results of this chapter are (i) an asymptotical upper bound of order of magnitude $O(n^{2/3})$, (ii) the fact that one cannot get essentially better bound using this method yielding a lower bound for the reconstruction from the sum of the elements of the k-deck which differs in a factor $O(\sqrt[3]{\log n})$ only from the upper bound.

The results of Chapter 5 are based on a manuscript joint with Géza Kós and Péter Sziklai [L2].

In Chapter 6 we present an algorithm on words which has an application to bioinformatics. The algorithm determines the evolutionary distance between two organisms, hence it can be used in the statistical analysis of the genome rearrangement via *Markov chain Monte Carlo methods.*

The evolutionary distance can be determined by comparing the order of appearance of orthologous genes in the genomes of the organisms. Among the numerous parsimony approaches that try to obtain the shortest sequence of rearrangement operations sorting one genome into the other, Bayesian Markov chain Monte Carlo methods have been introduced a few years ago. The computational time for convergence in the Markov chain is the product of the number of steps needed in the Markov chain and the computational time needed to perform one MCMC step. Therefore faster methods for making one MCMC step can reduce the mixing time of an MCMC in terms of computer running time.

We introduce an efficient algorithm for characterizing and sampling transpositions and inverted transpositions for Bayesian MCMC. The algorithm characterizes the mutations by the change in the number of cycles in the *graph of desire and reality*. We show that this is equivalent with the following searching problem on words: for every position of a given signed permutation of length n decide whether there can be found a given 3-long permutation starting in this position. The algorithm runs in $O(n)$ time and uses $O(n)$ memory, where n is the size of the genome. This is a significant improvement compared with the so far available brute force method with $O(n^3)$ running time and memory usage.

The results of Chapter 6 are based on a joint paper with István Miklós and Timothy P. Brooks [L6].

In most of the chapters some open problems are presented.

1.3 Mathematical background

In the few subsequent sections we give a brief survey of the related problems and motivations on combinatorial reconstruction problems and of the known results concerning the reconstruction problem of words as well.

1.3.1 Combinatorial reconstruction problems

There is a wide range of reconstruction problems in combinatorics, however the name "The Reconstruction Problem" is devoted to the famous and notorious conjecture of Kelly [31] and Ulam [60]:

Conjecture 1 *Every finite graph of at least three vertices is determined up to isomorphism by the $n-1$-deck, consisting of its one-vertex-deleted subgraphs.*

For some graph-classes the conjecture is known to be true, e.g. for disconnected graphs and trees and for graphs of at most as many edges as vertices, etc., however the problem is still open in general. For an interested reader we recommend the chapter of Babai [4] who gives a comprehensive discussion of this topic.

An *ordered graph* on n vertices is a graph with a linear ordering on its vertices inherited by its subgraphs, i.e. the vertices are indexed by the numbers $1, \dots, n$, and in the case of considering its subgraph on k vertices with vertex-indices $i_1, \dots, i_k$ the vertices must be re-indexed with the numbers $1, \dots, k$ keeping the order of the indices.

A bipartite graph with color-classes on n_1 and n_2 vertices is ordered if both of the color-classes are ordered and no order relation between the vertices of different color-classes is given. In other words the vertices are indexed by the numbers $1, \dots, n_1$ and $1, \dots, n_2$, respectively, and the subgraphs inherit the orderings, similarly as above.

Note that the vertex-reconstruction of ordered graphs is a special case of reconstruction of square matrices with symmetric deletions by considering the adjacency matrix of the graph, see Figure 1.1.

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

$$\downarrow$$

$$\overbrace{\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}}$$

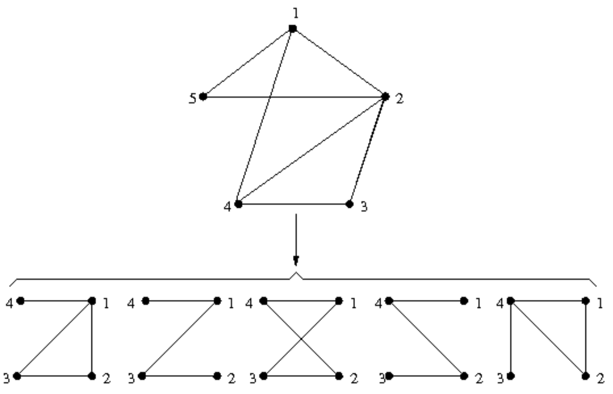


Figure 1.1: The 4-deck of an ordered graph on 5 points and the 4-deck of its adjacency matrix

Similarly the vertex-reconstruction of ordered bipartite graphs is a special case of reconstruction of square matrices with not necessarily symmetric deletions by considering the adjacency matrix of the bipartite graph.

Tardos [58] settled a series of conjectures and solved problems with the help of 0-1 matrices and ordered graphs. Marcus and Tardos [44] examined the extremal problem of how many 1 entries a square binary matrix of order n, that avoids a certain fixed submatrix P, can have. They proved a linear bound for any permutation matrix P, settling the conjecture of Stanley and Wilf on the number of n-permutations avoiding a fixed permutation and other related open problems. Tardos [59] determined the order of magnitude of the maximal number of 1 entries of a square binary matrix of order n avoiding P for all patterns P with four 1 entries. In the same framework Pach and Tardos [49] disproved the general conjecture of Füredi and Hajnal and presented a new proof of a theorem of Spencer et al. [55] claiming that the number of occurrences of the unit distance among n points on the plane is $O(n^{4/3})$.

Alon et al. [3] examined the reconstruction problems in a more general framework with the help of tools from permutation group theory. In their model they supposed that a set X and a group of automorphisms G of X are given. Then considering an arbitrary $Y \subseteq X$ on m elements and the set of representatives R of orbits of Y, the k-deck of Y tells the number of subsets of Y on k elements which are in the same orbit with a representative r for every $r \in R$.

In this scenario they considered the following reconstruction problem:

Problem: *Given a set X, a group of automorphisms G of X, and two integers m, k $(m > k)$, can every subset Y on m elements be reconstructed (up to G-equivalence) from its k-deck?*

One can consider the edge-reconstruction conjecture of Harary [26] within this framework as a nice example. Let X be the set of all $\binom{n}{2}$ edges of a complete graph on n vertices and G is the group of all permutations of X induced by permuting the vertices of the complete graph. Then a subset $Y \in X$ on m elements is a graph on n vertices and m edges, and its orbit is the set of every graph isomorphic to it. The $m-1$-deck is the set of isomorphism types of all edge-deleted subgraphs of Y with multiplicities.

The authors proved several relationships between the parameters of the problem (i.e. m, k, the size of G and of the orbits, etc.) which ensure reconstructability, and discussed more combinatorial and geometrical reconstruction problems.

1.3.2 Preliminary results on reconstruction of words

As we noted above, there are two different versions of the reconstruction problem of words: reconstruction from the k-deck consisting of the multiset of subwords and reconstruction from the k-deck consisting of the different subwords. In the following we present the known results of both problems in chronology.

Reconstruction from the k-deck consisting of the multiset of subwords

The original problem raised from Kalashnik [28] who proved the following upper bound:

Theorem 2 *Every word w of length n can be reconstructed from its k-deck if $k \geq \lfloor \frac{n}{2} \rfloor$.*

Later Aleksanjan [1] gave an incorrect assertion claimed that every word w of length n can be reconstructed from its 3-deck. Zenkin and Leont'ev [62] proved the first lower bound.

Theorem 3 *It is not possible to reconstruct every word w of length n from its k-deck if $k \leq \frac{\log n}{\log \log n}$.*

Later Manvel et al. [42] improved the lower bound and gave a new proof of the upper bound of Theorem 2.

Theorem 4 *Every word w of length n can be reconstructed from its k-deck if $k \geq \lfloor \frac{n}{2} \rfloor$.*

It is not possible to reconstruct every word w of length n from its k-deck if $k \leq \log_2 n$.

The lower bound is given by the well-known Prouhet-Thue-Morse sequence and its complement. The Prouhet-Thue-Morse sequence is a binary sequence and its n-th element is 1 if the number of ones in the binary form of n is odd and 0 otherwise. The Prouhet-Thue-Morse sequence has applications in many fields of science such as combinatorics on words, number theory, physics, etc. For a comprehensive discussion of this topic see Allouche and Shallit [2]. Here we present the first three examples of length 2^k of Manvel et al. denoted the same k-decks by $\sim_k$:

$$01 \sim_1 10, \quad 0110 \sim_2 1001, \quad 01101001 \sim_4 10010110, \quad \ldots$$

Choffrut and Karhumäki [15] improved the lower bound with a constant factor by showing that k is bounded by $\phi(n)$, where $\phi()$ is a Fibonacci-sequence defined by $\phi(1) = 2, \phi(2) = 5$ and $\phi(n) = \phi(n-1) + \phi(n-2)$ for $n \geq 3$. From this the authors proved the following:

Theorem 5 *It is not possible to reconstruct every word w of length n from its k-deck if $k \leq 4.77 \log n + 1.1$.*

Scott [51] made a substantial improvement of the upper bound using simple number theoretic and linear algebraic arguments:

Theorem 6 *Every word w of length n can be reconstructed from its k-deck if $k \geq (1 + o(1))\sqrt{n \log n}$.*

Dudík and Schulman [17] improved the lower bound with a generalization of the Prouhet-Thue-Morse sequence called templates:

Theorem 7 *It is not possible to reconstruct every word w of length n from its k-deck if $k \leq 3^{(\sqrt{2/3}-o(1))\log_3^{1/2} n}$.*

Independently Krasikov and Roddity [32] improved the upper bound using similar ideas than Scott [51]. They proved that if there are two different words of length n with the same k-deck, then for some s the system

$$a_1^h + a_2^h + .. + a_s^h = b_1^h + b_2^h + .. + b_s^h, h = 1, .., k-1$$

$$a_1 < a_2 < .. < a_s, \quad b_1 < b_2 < .. < b_s$$

has a nontrivial solution with $a_i, b_i \in [0, n-1]$.

This is the Prouhet-Tarry-Escott problem of classical Diophantine analysis. Using a new result of Borwein et al. [11] on this problem the authors get the following:

Theorem 8 *Every word w of length n can be reconstructed from its k-deck if $k \geq \lfloor \frac{16}{7}\sqrt{n} \rfloor + 5 \approx 2.286\sqrt{n}$.*

Furthermore they proved that this bound cannot be improve significantly *with this method*, since for $k \leq \sqrt{2\log 2}\sqrt{\dfrac{n}{\log n}} - \dfrac{1}{2}$ the above system has a non-trivial solution.

Foster and Krasikov [22] improved the result of Borwein et al. [11] concerning the Prouhet-Tarry-Escott problem which yields a slight improvement in the constant factor of the upper bound.

Theorem 9 *Every word w of length n can be reconstructed from its k-deck if $k \geq 2\lfloor \sqrt{n \log 2} \rfloor + 3 \approx 1,665\sqrt{n}$.*

Leont'ev and Smetanin [34] proved that determining whether a given word can be uniquely reconstructed from a given set of subwords is an NP-complete problem.

Reconstruction from the k-deck consisting of the different subwords

This problem originates from Simon [53] who proved the following:

Theorem 10 *Every word w of length at most n is uniquely determined by its length and the $\lceil \frac{n+1}{2} \rceil$-deck of its different subwords.*

Lothaire [40] gave an elegant proof of this bound for general alphabet. Since we were inspired by they train of thought, and we have proved a few generalizations of Theorem 10, we present their original proof in 4.3. Contrary to the case of reconstruction from the multiset of subwords, this result is sharp for a binary alphabet. One can see that easily from the pair of words *abab..ab* and *baba..ba*.

Levenshtein in his papers [36; 37; 38] considered more generalizations of this reconstruction problem. In [37] the author examines which other sets of subwords or super-words determine uniquely the original word, in [36] the maximum size of the set of common subwords (or super-words) of two different words of fixed length is given in a recursive way. In [38] every unknown sequence is reconstructed from its versions distorted by errors of a certain type, which are considered as outputs of repeated transmissions over a channel, and a minimal number of transmissions sufficient to reconstruct the original word (either exactly or with a given probability) is given. In both of the last papers simple reconstruction algorithms are given.

Leont'ev and Smetanin [34], Simon [54], and Dress and Erdős [16] approached the problem from algorithmic point of view. The authors presented algorithms of linear running-time which reconstruct a word from the set of its different subwords.

1.4 Biological background

In the next sections we briefly outline the structural properties of the DNA and the main notions of the genome rearrangement necessary to the examined mathematical models.

1.4.1 The structure of the DNA

DNA (*deoxyribonucleic acid*) is a nucleic acid that contains the genetic instructions and information used in the development and functioning of all known living organisms. The DNA segments that carry this genetic information are called *genes*, but other DNA sequences have structural purposes, or are involved in regulating the use of this genetic information.

DNA is a strand composed of four *nucleotides* or *bases* called *adenine*, *cytosine*, *guanine* and *thymine*, abbreviated by A, C, G and T, respectively. These nucleotides form two base-pairs which are $A - T$ and $C - G$ called *complement pairs*. The chemical structure of DNA induces an orientation of the strands: there is a starting and an ending point of the DNA, denoted by 5' and 3'. Furthermore

DNA is double-stranded (which is called the *double helix*), i.e. every strand bonds to an other strand, to its *reverse complement*. Reverse means that in a double helix the direction of the nucleotides in one strand is opposite to their direction in the other strand and complement means that every nucleotide of a strand bonds to the other base of its complement pair in the other strand. See Figure 1.2 as an illustration of the structural properties above.

$$\begin{array}{lcccccccccccl}
 & \multicolumn{10}{c}{\xrightarrow{\hspace{12em}}} & \\
5' - \ldots & C & G & C & C & T & T & A & T & A & C & \ldots - 3' \\
 & | & | & | & | & | & | & | & | & | & | & \\
3' - \ldots & G & C & G & G & A & A & T & A & T & G & \ldots - 5' \\
 & \multicolumn{10}{c}{\xleftarrow{\hspace{12em}}} &
\end{array}$$

Figure 1.2: Part of a DNA double helix with complementary bonds

Obviously one can get the reverse complement of a DNA strand in two steps: first by reverting the whole strand and second by replacing every nucleotide with its complement.

1.4.2 Genome rearrangement

Genes are finite sequences of nucleotides, and since these sequences are double-stranded every gene has a "reading direction". Genomes are assumed to have the same gene content, and each gene is represented in one copy in every genome. Gene orders are described as signed permutations, numbers correspond to genes, signs represent the reading direction of genes.

There are different types of *mutations* or *rearrangement operations* acting on genomes, here we present the ones which will be taken into consideration in our mathematical model and some others. Let's denote the genes by γ_i-s, some of the possible mutations are the following:

- *inversions*: this type of mutations reverses a consecutive block of the genome and changes the reading directions of all genes in the block, i.e.:

$$\ldots \gamma_i \overrightarrow{\gamma_{i+1} \cdots \gamma_j} \gamma_{j+1} \cdots \rightsquigarrow \ldots \gamma_i \overleftarrow{-\gamma_j \cdots - \gamma_{i+1}} \gamma_{j+1} \cdots$$

- *transpositions*: this type of mutations interchanges two consecutive blocks of the genome, i.e.:

$$\cdots\gamma_i\overrightarrow{\gamma_{i+1}\cdots\gamma_j}\overrightarrow{\gamma_{j+1}\cdots\gamma_k}\gamma_{k+1}\cdots \rightsquigarrow \cdots\gamma_i\overrightarrow{\gamma_{j+1}\cdots\gamma_k}\overrightarrow{\gamma_{i+1}\cdots\gamma_j}\gamma_{k+1}\cdots$$

- *inverted transpositions*: this type of mutations can be considered as a concatenation of a transposition and an inversion on one block of the transposition performed, i.e.:

$$\cdots\gamma_i\overrightarrow{\gamma_{i+1}\cdots\gamma_j}\overrightarrow{\gamma_{j+1}\cdots\gamma_k}\gamma_{k+1}\cdots \rightsquigarrow \cdots\gamma_i\overleftarrow{-\gamma_k\cdots-\gamma_{j+1}}\overrightarrow{\gamma_{i+1}\cdots\gamma_j}\gamma_{k+1}\cdots$$

- *translocations*: this type of mutations insert one consecutive block of a genome into another genome, i.e.:

$$\cdots\gamma_i\overrightarrow{\gamma_{i+1}\cdots\gamma_j}\gamma_{j+1}\cdots \rightsquigarrow \cdots\rho_k\overrightarrow{\gamma_{i+1}\cdots\gamma_j}\rho_{k+1}\cdots$$

Given two genomes π_1, π_2 with the same gene content, the problem of the genome rearrangement is to determine the *evolutionary distance* of two genomes, which is the minimal number of mutations transforming π_1 to π_2.

These processes are studied by *parsimony methods*, i.e. what is the minimum number of a certain type of events (mutations) needed to turn one genome into another.

Chapter 2

Reconstruction from the $n-1$-deck and automorphism groups

2.1 DNA-words

2.1.1 Motivation and notation

Let $\mathcal{A} = \{\{\alpha_1, \bar{\alpha}_1\}; \{\alpha_2, \bar{\alpha}_2\}; ...; \{\alpha_q, \bar{\alpha}_q\}\}$ be an alphabet of q pairs of symbols (called *complement pairs*), the finite sequences composed from $\mathcal{A}$ are called *DNA-words*. Define $\bar{\bar{\alpha}}_i = \alpha_i$; and for a word $w = x_1 x_2 ... x_t$ over $\mathcal{A}$ let $\widetilde{w} = \bar{x}_t \bar{x}_{t-1} ... \bar{x}_1$ be the *reverse complement* of w. Note that $\widetilde{(\widetilde{w})} = w$. Since we want to keep the essence of the partial ordering of ordinary words, we identify each DNA-word with its reverse-complement, i.e. every w with the corresponding $\widetilde{w}$, denoted by $w \equiv \widetilde{w}$. For obvious reasons, when $q = 2$ then the alphabet is often denoted by $\{\{A, \bar{A} = T\}; \{G, \bar{G} = C\}\}$.

Let $D^{q,n}$ denote the poset of all DNA-words of length at most n (defined over an alphabet of q complement pairs), partially ordered by the extension of the subword relation, i.e. $\{u, \widetilde{u}\} \leq \{v, \widetilde{v}\}$ if u is a subsequence of v or $\widetilde{u}$ is a subsequence of v. In this case we say that u *precedes* v or u is a *substrand* of v, and we use the notation $u \prec v$. A small example for a DNA-poset can be found in Figure 2.1: the poset of substrands of the DNA-word $ACGT$.

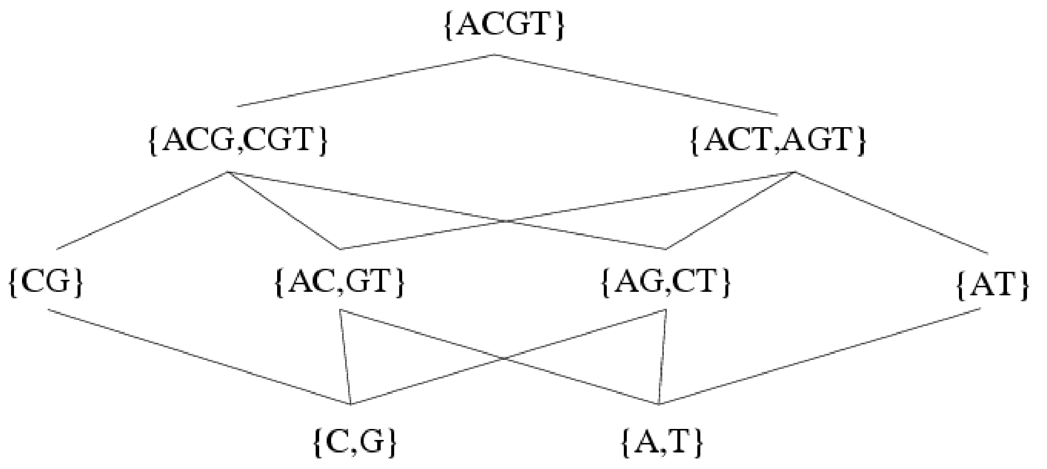


Figure 2.1: The poset of substrands of the DNA-word *ACGT*

2.1.2 Reconstruction from the $n-1$-deck

In the following we will prove that every DNA-word of length n can be reconstructed from the set of different DNA-words of length $n-1$ which precede it.

As a first step we reduce the case of a general alphabet to the case of the alphabet $\{\{A,T\},\{C,G\}\}$.

Lemma 11 *We can solve a reconstruction problem of all DNA-words over an alphabet with $q \geq 2$ complement pairs if and only if we can do it for the similar problem for $q = 2$, i.e. if and only if we can reconstruct all DNA-words over the alphabet $\{\{A,T\},\{C,G\}\}$.*

Proof: It is clear that if we can reconstruct all DNA-words over an alphabet with $q \geq 2$ complement pairs, then we can reconstruct them over $\{\{A,T\},\{C,G\}\}$. Conversely, suppose that we can reconstruct all DNA-words over $\{\{A,T\},\{C,G\}\}$. Then replace the first complement pair with $A-T$, and all the others with $C-G$. Now we can reconstruct the strand, and so we find the places of letters from the first complement pair in the original strand (now $A-T$-s are there); then we can repeat the procedure in order to find the other complement pairs. □

Now we show that a DNA-word can be reconstructed from its $n-1$-deck.

Lemma 12 *If $3 \leq i$ then every DNA-word of length i is uniquely determined by its substrands of length $(i-1)$.*

Proof: Because of Lemma 11, it is enough to consider DNA-words over the conventional $\{\{A, T\}, \{C, G\}\}$ alphabet. It is easy to see that for $i = 2$ the lemma is false: the DNA-words AT and TA have the same one letter substrand $\{A \equiv T\}$ but $AT \not\equiv TA$. Now w.l.o.g. we can suppose that we have a substrand with first letter A. Now consider substrands of form $A^k \alpha T^l$, where k is maximal and then l is maximal with respect to this k. Then $0 \leq l \leq k$ and $k \geq 1$.

Such a word can arise after deleting a letter from one of the following words: $A^k T^{l+1}$ (then α was empty, in this case $k > l$), $A^k \beta T^l$ for $|\beta| = |\alpha| + 1$, $A^m x A^{k-m} \alpha T^l$ for $m = 0, ..., k-1$ and $A^k \alpha T^{l-m} y T^m$ for $m = 0, ..., l-1$. Because of the maximality of k and l there are no more cases and the first letter of β is not A, the last is not T, further $x \neq A, y \neq T$.

Now we search for substrands of form $A^{m-1} x A^{k-m} \alpha T^l$. If we have such a substrand for some $m = m^* \in \{1, ..., k-1\}$ and for $m = m^* + 1$, then the original word was $A^{m^*} x A^{k-m^*} \alpha T^l$. If we have for $m = 1$ but not for $m = 2$ then the original word was $x A^k \alpha T^l$. If we have for $m = k$ but not for $m = k-1$, then $A^k x \alpha T^l$. If we are not in the above cases, we search for substrands of form $A^k \alpha T^{l-m} y T^{m-1}$ and follow the above train of thought. At last if we haven't found a substrand of the above forms, then we have a substrand $A^{k-1} \beta T^l$, where we get β from α by inserting a letter. Then the word is $A^k \beta T^l$. The proof is complete.$\square$

2.1.3 The automorphism group of the DNA poset

Now our aim is to determine $\mathrm{Aut}(D^{q,n})$. There are two obvious types of automorphisms: a permutation $\pi \in Sym_q$ on the complement pairs induces an automorphism σ_π on $D^{q,n}$. Denote also by Sym_q the automorphism group generated by these σ_π-s. Furthermore, consider a map which interchanges the elements of the i-th complement pair. This induces an automorphism $\widehat{\sigma_i}$ on $D^{q,n}$. Denote by Z_2 the automorphism group generated by $\widehat{\sigma_i}$. We will prove that for $n \geq 3$ there are no more automorphisms (note that the automorphism that reverses the order of the letters, which is a natural one, is $\widehat{\sigma_1}\widehat{\sigma_2}...\widehat{\sigma_q}$; e.g. $\widehat{\sigma_1}\widehat{\sigma_2}(ab) = \bar{a}\bar{b}$, which is identified with its reverse complement, i.e. ba).

Theorem 13 *(i) if $n = 1$, then $\mathrm{Aut}(D^{q,1}) = Sym_q$;*
(ii) if $n = 2$, then $\mathrm{Aut}(D^{q,2}) = Sym_q \otimes Sym_3^q \otimes Sym_4^{\binom{q}{2}}$;
(iii) if $n \geq 3$, then $\mathrm{Aut}(D^{q,n}) = Sym_q \otimes Z_2^q$.

Proof: The case $n = 1$ is considered only for the sake of completeness. In this case an automorphism is a simple permutation on the q complement pairs.

It is clear, that the levels of the poset are invariant under an automorphism. Furthermore, an automorphism transfers complement pairs to complement pairs. Take an arbitrary automorphism $\sigma_0 \in \mathrm{Aut}(D^{q,n})$, and consider its action on $D_1^{q,n}$ (i.e. on the set of complement pairs). Thus, this is a permutation on $\mathcal{A}$, take its inverse π^{-1} on $\mathcal{A}$. This permutation induces an automorphism $\sigma_{\pi^{-1}}$ on the poset $D^{q,n}$. Let $\sigma_1 = \sigma_0\sigma_{\pi^{-1}}$. Then σ_1 fixes all of the complement pairs. Now one can partition the second level into $q + \binom{q}{2}$ blocks: we have q blocks of size 3 with elements $\{a_ia_i \equiv \bar{a}_i\bar{a}_i, a_i\bar{a}_i, \bar{a}_ia_i\}$; and $\binom{q}{2}$ blocks of size 4, with elements $\{a_ia_j \equiv \bar{a}_j\bar{a}_i, a_i\bar{a}_j \equiv a_j\bar{a}_i, \bar{a}_ia_j \equiv \bar{a}_ja_i, \bar{a}_i\bar{a}_j \equiv a_ja_i\}$ for all $i \neq j$, each block is fixed by σ_1 (setwise).

If $n = 2$, then within these blocks one can specify the image of all elements freely. This means q copies of Sym_3 and $\binom{q}{2}$ copies of Sym_4, and these automorphisms differ and commute.

Now let $n \geq 3$ and consider the effect of σ_1 on $D_2^{q,n}$.

Claim 14 *σ_1 fixes all sequences of form $a_ia_i \equiv \bar{a}_i\bar{a}_i$.*

Proof: To the contrary, suppose that $\sigma_1(a_ia_i) = a_i\bar{a}_i$ (or $\bar{a}_ia_i$ which case is similar). Then $\sigma_1(a_ia_ia_i) = a_i\bar{a}_i\bar{a}_i$ or $\sigma_1(a_ia_ia_i) = a_ia_i\bar{a}_i$. In both cases $\triangle\,\sigma_1(a_ia_ia_i)$ has two elements, but $\triangle(a_ia_ia_i)$ has only one element, hence we cannot define $\sigma_1(a_ia_ia_i)$. □

Let $\widehat{\sigma_i}$ be the automorphism which interchanges the elements of the i-th complement pair. Take the product of those $\widehat{\sigma_i}$'s for which $\sigma_1(a_i\bar{a}_i) = \bar{a}_ia_i$, then let $\sigma_2 := \sigma_1 \prod_{\substack{i:\sigma_1(a_i\bar{a}_i)= \\ =\bar{a}_ia_i}} \widehat{\sigma_i}$ for all i. Then σ_2 fixes all elements in the 3-blocks. If $q = 1$, the proof is complete. Let $q \geq 2$.

Claim 15 *σ_2 fixes all sequences of form $a_ia_j, i \neq j$.*

Proof: To the contrary, suppose first that $\sigma_2(a_i a_j) = a_j a_i$. Then $\sigma_2(\bar{a}_i a_i a_j) = \bar{a}_i a_j a_i$ because $\bar{a}_i a_i$ is fixed, therefore $\bar{a}_i a_j$ is fixed too. But then we cannot define $\sigma_2(a_i \bar{a}_i a_j)$. Now suppose that $\sigma_2(a_i a_j) = a_i \bar{a}_j$ ($\bar{a}_i a_j$ is similar). Then $\sigma_2(a_i a_j \bar{a}_j) = a_i a_j \bar{a}_j$ because $a_j \bar{a}_j$ is fixed, then $\sigma_2(a_i \bar{a}_j) = a_i a_j$. But then we cannot define $\sigma_2(\bar{a}_j a_i a_j)$. □

Now we know that σ_2 is the identity on $D_1^{q,n}$ and $D_2^{q,n}$. Because of Lemma 12 it is true for $D_3^{q,n}$, and by induction for all $D_i^{q,n}$ ($i = 4, \ldots, n$). The proof is complete. ■

2.2 Matrices

2.2.1 Motivation and notation

Let Σ be a finite alphabet and $\Sigma^{n\times n}$ the set of all $n \times n$ matrices over Σ, furthermore let $\Sigma^{\square} = \bigcup \Sigma^{n\times n}$, the set of all finite square matrices over Σ . For $A \in \Sigma^{k\times k}$ and $B \in \Sigma^{n\times n}$ we say that A is a *submatrix* of B if we can get A by deleting some $n-k$ rows and some $n-k$ columns of B (we use the notation $A \trianglelefteq B$). Denote by $s_k(B) = \{A \in \Sigma^{l\times l} : A \trianglelefteq B, l = 1, ..., k\}$ the set of all submatrices of B of size at most $k \times k$. Let M_q^n denote the partially ordered set of all elements of $\Sigma^{i\times i}$ for $i = 1, ..., n$ and for $\Sigma = \{0, 1, ..., q-1\}$, partially ordered with this submatrix relation.

2.2.2 Reconstruction from the $n-1$-deck

We begin this section with some basic observations. The case $q = 1$ (i.e. homogeneous 0-matrices) is not interesting, so we can suppose that $q \geq 2$. We can prove easily, that the size of the alphabet is not important.

Lemma 16 *We can solve the reconstruction problem for all square matrices if and only if we can solve it for $q = 2$, i.e. if and only if we can do it for $0-1$ square matrices.*

Proof: It is clear that if we can solve the problem for an arbitrary alphabet, then we can solve it for a two-element alphabet. Conversely, suppose that we can solve it for $0-1$ square matrices, and consider $A, B \in \Sigma^{n\times n}$ such that $A \neq B$. This means that there exist $1 \leq i, j \leq n$, such that $a = a_{i,j} \neq b_{i,j} = b$. Then replace all occurrences of a by 0 and all the other letters by 1, denote A^* and B^* the new $0-1$ square matrices respectively. Clearly $A^* \neq B^*$, then by the assumption $s_k(A^*) \neq s_k(B^*)$, i.e. there exist $C^* \in \{0,1\}^{k\times k} : C^* \trianglelefteq A^*, C^* \ntrianglelefteq B^*$ and C^* has a 0 entry. One can obtain the corresponding matrix C by deleting the set of rows and columns from A which arises C^* from A^*. Then clearly $C \ntrianglelefteq B$, which completes the proof. □

Now we show that matrices can be reconstructed from its $n-1$-deck.

Lemma 17 *For $n \geq 3$ every square matrix in $\Sigma^{n\times n}$ is uniquely determined by its submatrices of size $(n-1)\times(n-1)$.*

Proof: For $n = 2$ the statement is clearly not true. Furthermore, because of Lemma 16, we consider $0-1$-matrices only. We prove the lemma by induction. Let $n = 3$, there exist $2^9 = 512$ binary matrix of size 3×3. We can check by a simple Matlab program that all of its submatrix-sets are different. Now suppose that the statement is true for $n = k$. Let $A \in \Sigma^{(k+1)\times(k+1)}$ and consider $s_k(A)$. Now by deleting the last row and the last column from all elements of $s_k(A)$ we get the set s'. Clearly $s' = s_{k-1}(A')$ where A' denotes the matrix, which we get from A by deleting its last row and last column. Applying the induction for A', we can determine the upper left $(n-1)\times(n-1)$ of A. Similarly we can determine the first $n-1$ rows, and last $n-1$ column, etc. □

2.2.3 The automorphism group of the matrix poset

Now our aim is to give the automorphism group of the posets M_q^n for all q and n. There are two obvious types of automorphisms: a permutation $\pi \in Sym_q$ on the elements of Σ induces an automorphism σ_π on M_q^n. Denote also by Sym_q the automorphism group generated by these σ_π-s. At second consider the elements of the congruence group of the square (the matrix), which is generated by a rotation (of order 4), and a vertical reflection (of order 2). These induce the automorphisms $\sigma_{rot}, \sigma_{ref}$ on M_q^n. Denote by D_4 the automorphism group generated by the congruences. Clearly, these automorphisms differ and commute with the ones in Sym_q. We will see that for $n \geq 3$ there are no more automorphisms of the matrix poset.

Theorem 18 *(i) if* $n = 1$*, then* $\mathrm{Aut}(M_q^1) = Sym_q$
(ii) if $n = 2$*, then* $\mathrm{Aut}(M_q^2) = Sym_q \otimes Sym_{14}^{\binom{q}{2}} \otimes Sym_{36}^{\binom{q}{3}} \otimes Sym_{24}^{\binom{q}{4}}$
(iii) if $n \geq 3$ *, then* $\mathrm{Aut}(M_q^n) = Sym_q \otimes D_4$

Proof: The case $n = 1$ is considered only for the sake of completeness. In this case an automorphism is a simple permutation on the q letters.

Let $n \geq 2$. Take an arbitrary automorphism $\sigma_0 \in \mathrm{Aut}(M_q^n)$ and consider its action on the first level of the poset, i.e. on the letters. This is a permutation on Σ, take its inverse π^{-1} on the letters. This induces an automorphism $\sigma_{\pi^{-1}}$ on the whole poset. Let $\sigma_1 = \sigma_0\sigma_{\pi^{-1}}$. Then σ_1 fixes the letters (the first level of the poset). Now we can partition the second level of the poset into four parts: in the first part there are the q homogeneous 2×2 matrices, these are fixed, because the letters are fixed. In the second part there are the matices containing two letters only, for every pair of letters there are 14 such a 2×2 matrices, these build a block, and there are $\binom{q}{2}$ blocks. In the third part are the matrices containing only three letters, for every triple of letters there are 36 such a 2×2 matrices, and there are $\binom{q}{3}$ blocks. And at last in the fourth part are the matrices containing four different letters, these build $\binom{q}{4}$ blocks of size 24. Clearly, these blocks are setwise fixed. Now in $n = 2$, then in each block we can specify the image of all elements freely. These automorphisms differ and commute, hence we get the second part of the theorem.

Now let $n \geq 3$. Our aim is to prove that except the elements of $D_4 =< \sigma_{rot}, \sigma_{ref} >$, (the rotation and the reflection w.r.t horizontal axis) there are no more. Now consider the 14-element block of the matrices containing the 0 and 1 letters only. For the sake of simplicity we present these matrices:

$$\Bigg\{ \overbrace{\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}}^{type1\ subblock}; \overbrace{\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}}^{type1\ subblock};$$
$$\underbrace{\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}}_{type2\ subblock}; \underbrace{\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}}_{type2\ subblock}; \underbrace{\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{type3\ subblock} \Bigg\}$$

We have some restrictions concerning the images of the elements of this block: there are five subblocks of three types, i.e. there are some properties which are not changing under every automorphisms, if the first level of the poset is fixed. The properties determining the subblocks are the following:

type1 subblock There are matrices which have *only one parent* who has *one more child only*, a *homogeneous* one (this parent is for example $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$ for the first matrix of the block). These matrices form two four-element subblocks concerning their homogeneous brother . It is easy to see, that there is no other matrix in this block with this property.

type2 subblock There are two matrix-pairs such that they have *two common parents*, such that they have only *one more common brother*, and it is *homogeneous* (one of these parents is for example $\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$, then the matrix-pair is $\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$.) As above we can see, that no other matrix-pairs satisfy this assumption. Notice that we cannot distinguish these matrix-pairs in the poset, because $\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ with the parent $\begin{pmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{pmatrix}$ have the same structure in the poset as of the above example.

type3 subblock The remaining two matrices $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ and $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ form one more subblock.

So the automorphism must fix each subblock setwise. We will prove that, if the four elements of the above matrix-pairs (i.e. the second type of the subblock) are fixed, then the whole second level of the poset is fixed.

Suppose first that $\sigma_1(\begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}) \neq \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix}$. We can fix this matrix easily by a rotation. Now let $\sigma_2 = \sigma_1 \sigma_{rot}$. This yields that the other element in the subblock (i.e. $\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$) is fixed too. Now we must fix the elements of the other subblock. Suppose that $\sigma_2(\begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}) = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$. Now let $\sigma_3 = \sigma_2 \sigma_{ref}$. Notice that this automorphism do not change the elements of the other matrix-pair, this results that all the matrices of the two matrix-pairs are fixed under σ_3.

Claim 19 *σ_3 fixes the 0-1-matrices in the second level of the poset .*

Proof: Because of the above we have some restrictions for the images of the elements. We analyze one case in detail, the proof of other cases are same, here we present only the counterexamples.

Suppose that $\sigma_3(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$. Consider the children of $\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ in the poset. These are $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$. Only the first child is not fixed by σ_3, that is the children of $\sigma_3(\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix})$ are $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$, $\begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$ and $\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$, and we can check fast that there is no 0-1-matrix in $\Sigma^{3\times 3}$ with these children, i.e. $\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ is fixed under σ_3.

Suppose that $\sigma_3(\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, or $\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$. Now, as above, we cannot define $\sigma_3(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix})$. At last suppose, that $\sigma_3(\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, now we cannot define $\sigma_3(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 1 \end{pmatrix})$.

Similarly we can see, that all the matrices of the four-element subblocks are fixed under σ_3, the proof is complete. □

Now our aim is to prove that the elements in the other blocks of size 14, i.e. the matrices containing only two letters (but not 0 and 1) are fixed under σ_3.

Claim 20 *σ_3 fixes all matrices in the second level of the poset containing only two letters.*

Proof: We prove at first that the 2×2 matrices with children 0 and a are fixed, for $a = 2, ..., q-1$. Remember that the letters are fixed, and we have some restrictions for the images of the matrices as above. In the proof we must consider five cases, these steps are similar to the above observations, so we omit here the full analysis, it is left to the reader.

1. $\begin{pmatrix} 0 & 0 \\ 0 & a \end{pmatrix}$ is fixed, otherwise we cannot define $\sigma_3(\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & a \end{pmatrix})$.

2. $\begin{pmatrix} 0 & a \\ a & a \end{pmatrix}$ is fixed, otherwise we cannot define $\sigma_3(\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & a \\ 0 & a & a \end{pmatrix})$.

3. $\begin{pmatrix} 0 & a \\ a & 0 \end{pmatrix}$ is fixed, otherwise we cannot define $\sigma_3(\begin{pmatrix} 0 & a & 1 \\ a & 0 & a \\ 1 & a & 0 \end{pmatrix})$.

4. $\sigma_3(\begin{pmatrix} a & 0 \\ a & 0 \end{pmatrix}) \neq \begin{pmatrix} a & a \\ 0 & 0 \end{pmatrix}$, or $\begin{pmatrix} 0 & 0 \\ a & a \end{pmatrix}$, else we cannot define $\sigma_3(\begin{pmatrix} a & 0 & 0 \\ 0 & 1 & 0 \\ a & 0 & 0 \end{pmatrix})$.

5. $\begin{pmatrix} a & 0 \\ a & 0 \end{pmatrix}$ is fixed, otherwise we cannot define $\sigma_3(\begin{pmatrix} a & 0 & 0 \\ a & 0 & 0 \\ a & a & 0 \end{pmatrix})$.

Now we have that all matrices having the children 0 and a only are fixed, next we have to prove that every matrix composed of $a, b \notin \{0, 1\}$. Here we present one case only, the other cases can be handle by a similar way:
$\begin{pmatrix} a & b \\ b & a \end{pmatrix}$ is fixed, otherwise we cannot define $\sigma_3(\begin{pmatrix} a & 0 & b \\ 0 & 0 & 0 \\ b & 0 & a \end{pmatrix})$)
Now we have that all matrices containing only two letters are fixed. □

Claim 21 *σ_3 fixes all matrices in the second level of the poset.*

Proof: The only remaining thing is to prove that the matrices containing three and four letters are fixed too. From the previous claim and the same type of counterexamples as the last one in Claim 20 this will follows. We present here one case of the three-letter case only, the remaining ones and the four-letter case can be handle similarly:
$\begin{pmatrix} a & b \\ c & a \end{pmatrix}$ is fixed, otherwise we cannot define $\sigma_3(\begin{pmatrix} a & 0 & b \\ 0 & 0 & 0 \\ c & 0 & a \end{pmatrix})$.
□

Now we know that σ_3 is the identity on the first two levels of the poset. Because of Lemma 17 it is the identity on the third level, and by induction on the whole poset. This completes the proof. ■

2.3 A short proof for a theorem of Burosch, Gronau and Laborde

2.3.1 Motivation and notation

At the end of this chapter we show an application of the method used above, i.e. the reconstruction from the $n-1$-deck, by giving a short proof for the nice theorem by Burosch, Gronau and Laborde [12]. Their result determines the automorphism group of a poset consisting of all subwords of a certain word $u_{m,n}$ which has maximum number of different subwords among the words of length n over an m-letter alphabet. Chase [14] showed that the maximum is realized if and only if the word is a repeated permutation of the alphabet and a prefix of the permutation, e.g. 4021340213402.

Based on this result, let $u_{m,n}$ denote the word $a_1...a_n$ where $m \geq 2$, $a_1 = 0$ and $a_{i+1} = a_i + 1$ modulo m, and let $B_{m,n}$ denote the set of all subwords of $u_{m,n}$ partially ordered by the subword relation. As a special case of the result of Erdős et al. [20] and Theorem 10, the elements of the poset $B_{m,n}$ can be reconstructed from its $n-1$-deck.

Lemma 22 *If $3 \leq i$ then every word of length i is uniquely determined by its subwords of length $(i-1)$.* $\square$

2.3.2 The automorphism group of the Burosch poset

Similarly, as in the previous sections of this chapter, based on the Lemma 22, we give a simple proof for the following theorem by Burosch, Gronau and Laborde [12] with proof of 13 pages:

Theorem 23 *(i) if $1 \leq n \leq m$, then $\mathrm{Aut}(B_{m,n}) = Sym_n$;*
(ii) if $m+1 \leq n \leq 2m-1$, then $\mathrm{Aut}(B_{m,n}) = Z_2 \otimes Sym_{2m-n}$;
(iii) if $2m \leq n$, then $\mathrm{Aut}(B_{m,n}) = Z_2$.

Before the proof let us describe the involutory automorphism of the "typical case" no.(iii). Consider $u_{m,n} = 01...(m-1)01...(m-1)...01...(m-1)01...(k-1)$.

Let σ^* be the mapping that reverses all the words, and let $\nu_{k,m}$ be the mapping that changes the letters in the words in the following way: for $0 \le i \le k-1$ the letter i is changed for $k-1-i$, and for $k \le j \le m-1$ the letter j is changed for $m+k-1-j$. Clearly neither σ^* nor $\nu_{k,m}$ is an automorphism of $B_{m,n}$, but $\sigma^*\nu_{k,m} \in \mathrm{Aut}(B_{m,n})$.

Proof: It is clear, that the levels of the poset are invariant under an automorphism. Also homogeneity (i.e. the property that a word has exactly one 1-letter subword) and total inhomogeneity (i.e. the property that a t-letter word has exactly t 1-letter subwords) are kept by every automorphism.

(i) Take an arbitrary automorphism $\sigma_0 \in \mathrm{Aut}(B_{m,n})$, and consider its action on the first level of the poset. Thus, this is a permutation π on $\{a_1, ..., a_n\}$, take its inverse π^{-1} on $\{a_1, ..., a_n\}$. This permutation induces an automorphism $\sigma_{\pi^{-1}}$ on the poset. Let $\sigma_1 = \sigma_0\sigma_{\pi^{-1}}$. Then σ_1 fixes all of the letters. Furthermore, σ_1 fixes all sequences of form ij where $i < j$ because $\sigma_1(ij) \ne (ji)$ as ji is not a subword of $u_{m,n}$. Then σ_1 is the identity on the two lowest levels of the poset and, by Lemma 22, on the whole poset. □

(ii) In this case $u_{m,n} = 01...(m-1)01...(k-1)$ where $n = m+k$, $1 \le k \le m-1$ and let σ_0 be an arbitrary automorphism.

It is easy to see that if $i \le k-1$ then $\sigma_0(i) \le k-1$. Indeed, $\sigma_0(i) = j \ge k$ is impossible as ii is a subword of $u_{m,n}$ but jj is not.

Claim 24 *Let e be an element of the third level of the poset, $\triangle_1 e$ contains the two letters i, j only and suppose that ii is a subword of e. Then we can read from the poset whether j is the middle letter or not.*

Proof: In that case $e = iij, jii$, or iji. The shadows of the first two words have two elements, but the shadow of the third word has three elements. □

Claim 25 *Let $j_1 < j_2 \le k-1$ and $i \le k-1$, $i \ne j_1, j_2$, then we can tell the difference between the j_1iij_2-type subwords and the j_1j_2ii-type or iij_1j_2-type subwords in the poset.*

Proof: Consider the elements of the shade of j_1iij_2 containing the subword j_1j_1 (by inserting a letter j_1 in the above word: $j_1ij_1ij_2$ or $j_1iij_1j_2$). Now consider the element of the 3-shadow of this word which contain the letter j_2 and the subword $j_1 j_1$. In this case the inserted letter is the middle one what we can see from the poset because of Claim 24. We get the same by inserting the letter j_2; the inserted letter is the middle one. Now insert the letters j_1 or j_2 in $j_1 j_2 ii$: we get $j_1 j_2 i j_1 i$ or $j_1 j_2 i j_2 i$. We see by considering the elements of the 3-shadow that in the first case the inserted letter is not the middle one. For $ii j_1 j_2$ we get the same: the inserted letter is never the middle one. □

Claim 26 *The image of the letter i is i or $(k-i-1)$ by any automorphism.*

Proof: Consider the family of subwords of length $k+1$ having a 2-long homogeneous and a k-long totally inhomogeneous subsequence (i.e. with k different letters) and put $v_i = 01...(i-1)ii(i+1)...(k-1)$. Clearly the set $\{v_0, v_1, .., v_{k-1}\}$ is fixed by every automorphism. With the previous claim $\sigma_0(v_0) = v_0$ or v_{k-1}, because only for $i = 0$ or $k-1$ will v_i have no $j_1 ii j_2$-type subwords, as we can see because of Claim 25. Hence the image of 0 is 0 or $k-1$.

Now one can forget all the words in the poset containing 0 or $(k-1)$ and, inductively, as in the previous paragraph, it can be proved, that the image of 1 is either 1 or $(k-2)$, etc. □

It is also easy to see that every sequence $i(k-i-1)$ is fixed by every automorphism for $i < k-i-1$, as $\sigma(ii(k-i-1)) = ii(k-i-1)$ or $i(k-i-1)(k-i-1)$. Then $i(k-i-1)$ is fixed and so $(k-i-1)i$ is fixed, too.

From the statements above we can see that we have restrictions for the images of the letters $0, 1, ..., k-1$, but we are free to choose the images of the remaining $2m-n$ letters. Let σ_0 be an arbitrary automorphism, and consider its action on the letters $k, ..., m-1$ (i.e. on the first level of the poset), this induces a permutation π on these letters (still on the first level), take its inverse π^{-1}. This permutation induces an automorphism $\sigma_{\pi^{-1}}$ on the poset. Let $\sigma_1 = \sigma_0 \sigma_{\pi^{-1}}$. Then σ_1 is the identity on the letters $k, ..., m-1$ and, as above, σ_1 fixes all sequences of form ij where $k \leq i < j$.

Now we define a mapping ρ: given a word $w = x_1 x_2 ... x_s y_1 y_2 ... y_t z_1 z_2 ... z_u$, where $0 \leq x_i, z_i \leq k-1; k \leq y_i \leq m-1$; let $\rho(w) = z_u z_{u-1} ... z_1 y_1 y_2 ... y_t x_s x_{s-1} ... x_1$. Let ν be the mapping that changes the letters i $(0 \leq i \leq k-1)$ for $k-1-i$ in each word (and does not change the letters j for $k \leq j \leq m-1$). Clearly neither ρ nor ν is an automorphism but $\rho\nu$ is an involution in $\mathrm{Aut}(B_{m,n})$. Finally, if $\sigma_1(0) = (k-1)$ then let $\sigma = \rho\nu\sigma_1$ and if $\sigma_1(0) = 0$ then let $\sigma = \sigma_1$. Hence $\sigma(0) = 0$.

Claim 27 *The two lowest levels of the poset are fixed under σ.*

Proof: We prove first that $0i$ is fixed for $1 \leq i \leq k-1$, hence we get that the first level of the poset is fixed (remember that σ_1 fixes all sequences of form ij where $k \leq i < j$). Now suppose that $0i$ is not fixed. Then we have that $\sigma(0i)$ is either $0(k-i-1)$ or $i0$ or $(k-i-1)0$. Let $\sigma(0i) = 0(k-i-1)$, hence $\sigma(0(k-i-1)i) = 0(k-i-1)i$ because $(k-i-1)i$ is fixed, so $\sigma(0(k-i-1)) = 0i$ but then we cannot define $\sigma((k-i-1)0i)$. Now let $\sigma(0i) = i0$ (or similarly $(k-i-1)0$). Then $\sigma(0ii) = ii0$ which is not a subword of $u_{m,n}$. Note that, as a by-product, we have just proved that each letter $0 \leq i \leq k-1$ is fixed as well. (For $k \leq i \leq m-1$ we knew it already.)

It follows from this reasoning that $i0$ is fixed if i is at most $k-1$. Now let's see the image of ij. Suppose that $\sigma(ij) = ji$ (this is the only case to exclude as the first level is fixed). If $0 \leq i, j \leq k-1$ then we cannot define $\sigma(i0j)$ because $i0$ and $0j$ are fixed. If $0 \leq i \leq k-1 < j \leq m-1$ then we cannot define $\sigma(ij0)$. So ij is fixed (we knew it already if $k \leq i, j$). □

From this claim we get that σ is the identity on the two lowest levels of the poset and so (by Lemma 22) on the whole poset, which proves part (ii) of the theorem. □

(iii) Now the word is of the following form $u_{m,n} = 01...(m-1)01...(m-1)...01...$ $...(m-1)01...(k-1)$ where $n = lm + k$. Let σ_0 an arbitrary automorphism. Clearly $\sigma_0(i) \neq j$ for $0 \leq i < k \leq j \leq m-1$ (we could not define $\sigma_0(i^{l+1})$). Similarly to the above train of thought we get that for every automorphism σ_0 and for $0 \leq i \leq k-1$ we have $\sigma_0(i) = i$ or $k-i-1$ considering the subwords $w_i = 01...(i-1)i^{l+1}(i+1)...(k-1)$ and here for $k \leq j \leq m-1$ we have $\sigma_0(j) = j$ or $k+m-j-1$ considering the subwords $u_j = k...(j-1)j^l(j+1)...(m-1)$ as well. Now every sequence $i(k-i-1)$ and $j(k+m-j-1)$ are fixed by every automorphism for $0 \leq i \leq k-1$ and for $k \leq j \leq m-1$.

Remember the automorphism $\sigma^*\nu_{k,m}$ defined before the proof. Let σ be $\sigma^*\nu_{k,m}\sigma_0$ if $\sigma_0(0) = (k-1)$ and let σ be σ_0 if $\sigma_0(0) = 0$. Now $\sigma(0) = 0$. By the previous paragraph, similarly to **(ii)**, one can see that σ is the identity on the two lowest levels of the poset and because of Lemma 22 on the whole poset.

The case $m = k$ is slightly simpler, then we have only $\sigma_0(i) = i$ or $k - i - 1$ for $0 \leq i \leq k - 1$ by any automorphism, etc. The proof is complete. ■

2.4 Open problems

Problem 28 *Determine the automorphism group of the matrix poset in the case of symmetric deletions.*

As we mentioned in the Introduction one can define a submatrix of a square matrix by deleting rows and columns either symmetrically or arbitrarily. In 2.2 we handled the second case. Regarding the other case, there are two obvious types of automorphisms: the one induced by a permutation on the alphabet, and the reflection to the diagonal, i.e. for the poset $\mathrm{sym}M_q^n$ the previous claims suggest that $\mathrm{Aut}(\mathrm{sym}M_q^n) = Sym_q \otimes Z_2$ for sufficiently large n. However, contrary to the non-symmetric case, the term "sufficiently large" means at least 4, since the symmetric analogue of Lemma 17 is not true for $n = 3$, which can be seen from the following example:

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Hence in order to prove the symmetric analogue of Theorem 18, one have to fix at least the three lowest level of the poset $\mathrm{sym}M_q^n$, which seems at least as technical as in the non-symmetric case, so we dispense with it.

Problem 29 *Further short proofs.*

Using the same tools as in 2.3 (i.e. with the help of the reconstruction from the $n-1$-deck) one can find shorter and new proofs of known results. One example can be the theorem of Carpi and deLuca [13], claiming that a word w is completely determined by its set of factors of length at most G_w+2, where G_w is the maximal length of a repeated factor of w (i.e. a factor of multiplicity at most two).

Chapter 3

Generalized bounds for reconstruction of words

3.1 Motivation and notation

Consider the t-element alphabet $\Sigma_t = \{0, 1, ..., t-1\}$ denoted by Σ_t throughout this section and the elements of the set Σ_t^n called *words*. We do not deal with the trivial case of $t = 1$, i.e. the alphabet consisting of one letter only. Remember that Simon [53] and Lothaire [40] proved the following bound:

Theorem 10 *Every word $w \in \Sigma_t^n$ is uniquely determined by its length and the $\lceil \frac{n+1}{2} \rceil$-deck consisting of its different subwords.*

Moreover, this result is sharp for the binary alphabet as the following example shows:

Example 30 *Consider the periodic words $w = 0101..01$ and $v = 1010..10$ of length $2n$. It is easy to see, that the n-deck consisting of the different subwords of v and also for w is Σ_2^n, i.e. all the binary words of length n.*

There are however several series of examples (see some in 3.3) showing that if one knows more about the structure of the word w then one can get an improvement of Theorem 10.

To describe a bit more this phenomenon we will specify some restrictions for the number of occurrences of letters in the words, which, in some cases, will result in better bound than Theorem 10.

Before we present our results let us recall two easy observations, which will be useful in what follows.

Remark 31 *We can reconstruct every $w \in \Sigma_t^n$ from the k-deck consisting of the multiset of $\binom{n}{k}$ subwords (or different subwords) for any $t \geq 2$ if and only if we can reconstruct every $w \in \Sigma_2^n$ from the k-deck consisting of the multiset of $\binom{n}{k}$ subwords (or different subwords resp.).*

Remark 32 *It is equivalent to know the set of subwords of length at most k instead of the set of subwords of length k.*

In this chapter we examine the case of different subwords only, hence from now on we use the term subwords for the set of different subwords.

3.2 General bounds for the number of letters

In the following we suppose that there are *common* lower and upper bounds for the number of occurrences of every letter in the words, i.e. for the examined words $w \in \Sigma_t^n$ we have

$$l \leq |w|_i \leq u$$

for $i = 0, \ldots, t-1$.

In the first Lemma we examine a slightly different problem of *reconstructing* the original word from its subwords. Note that this algorithmic point of view is a different problem, which is examined in several papers [16], [34], [54] in the case of the binary alphabet.

Lemma 33 *Let $w \in \Sigma_t^n$ such that $|w|_i \leq u$ for $i = 0, \ldots, t-1$. Then w is uniquely characterized by its length and by its subwords of length at most $u+1$.*

Proof: Let $\mathbb{S}$ denote the set of subwords of w of length at most $u+1$. Because of the assumption we get $i^{|w|_i} \in \mathbb{S}$, i.e. we know the exact number of each letter in w. In the next step we can find out the lengths of the 0-runs from the subwords $0^r j 0^{|w|_0 - r} \in \mathbb{S}$, moreover, we know which letters occur between two fixed consecutive 0-runs. Similarly, we can reconstruct the lengths of the runs of each letter i and the set of letters between two fixed consecutive i-runs. From these we can uniquely reconstruct the original word w by interleaving the runs. $\square$

Lemma 34 *Let $w \in \Sigma_t^n$ such that $|w|_i \geq l$ for $i = 0, \ldots, t-1$. Then w is uniquely characterized by its length and by its subwords of length at most $\lceil \frac{n-(t-2)\cdot l+1}{2} \rceil$.*

Proof: Let $\mathbb{S}$ denote the set of subwords of w of length at most $\lceil \frac{n-(t-2)\cdot l+1}{2} \rceil$ and suppose that there is a $v \in \Sigma_t^n, v \neq w$ with the same lower bound for the number of letters and the same subword-set $\mathbb{S}$. First we will show that $|w|_i = |v|_i$ for every i. Suppose to the contrary that $|w|_0 < |v|_0$, then the assumption yields $|v|_0 > \lceil \frac{n-(t-2)\cdot l+1}{2} \rceil$. Since $|w| = |v|$ we get that there is another letter, say, 1 such that $|w|_1 > |v|_1$ and from $|v_{\{0,1\}}| \leq n-(t-2)\cdot l$ we get $|v|_1 < \lceil \frac{n-(t-2)\cdot l+1}{2} \rceil$, which is a contradiction because $1^{|v|_1+1}$ is a subword of w and $1^{|v|_1+1}$ is not a subword of v.

From now on we suppose that $|w|_i = |v|_i$ for every i. Since $w \neq v$ there are two letters such that subwords of w and v builded up only from these letters are different. Take the two letters at a position where v and w differ. W.l.o.g. we can suppose that $w_{\{0,1\}} \neq v_{\{0,1\}}$. From the assumption we get $|w_{\{0,1\}}| = |v_{\{0,1\}}| \leq n-(t-2)\cdot l$. In this case we can use Theorem 10 for word length $n-(t-2)\cdot l$, so we find a subword of length at most $\lceil \frac{n-(t-2)\cdot l+1}{2} \rceil$ which is contained in one of them only. This completes the proof. $\square$

Theorem 35 *Every word $w \in \Sigma_t^n$ with $l \leq |w|_i \leq u$ for $i = 0, \ldots, t-1$ is uniquely characterized by its length and by its subwords of length at most $min\{u+1, \lceil \frac{n-(t-2)\cdot l+1}{2} \rceil\}$.*

Proof: Use Lemma 33 and Lemma 34. $\blacksquare$

The following remark is not surprising as we used Theorem 10 in the proof.

Remark 36 *For a binary alphabet Theorem 10 of Simon and Lothaire is a special case of Theorem 35 with the most general parameter values $l = 0, u = n$.*

Note that if there are different lower and upper bounds for the number of occurrences of every letter then we can prove similar bounds which are a bit stronger in some cases:

Remark 37 *Every word $w \in \Sigma_t^n$, with $l_i \leq |w|_i \leq u_i$ for $i = 0, \dots, t-1$ and $l_0 \leq \dots \leq l_{t-1}$, is uniquely characterized by its length and by its subwords of length at most* $\min\{\max_i u_i + 1, \lceil \frac{n - \sum_{i=0}^{t-3} l_i + 1}{2} \rceil\}$.

3.3 Examples

At the end of this chapter we present some natural examples where the two bounds of Theorem 35 coincide.

Example 38 *Let w be a permutation of order n, i.e. $l = u = 1, t = n$.*
Every permutation of order n can be reconstructed from its subwords of length 2. We get this from the bounds of Theorem 35 which are $u + 1 = \lceil \frac{n-(t-2)\cdot l+1}{2} \rceil = 2$.

More generally:

Example 39 *Let w and u two periodic words of t letters, i.e. $l = u = \frac{n}{t}$.*
$w = 01\dots(t-1)01\dots(t-1)\dots01\dots(t-1)$ *and* $v = 12\dots(t-1)012\dots(t-1)0\dots12\dots(t-1)0$. *Similarly to Example 30 it is easy to see, that the subwords of length $\frac{n}{t}$ of w and v are the same, however the subwords of length $\frac{n}{t}+1$ are not the same. We can get this from the bounds of Theorem 35 which are* $u + 1 = \lceil \frac{n-(t-2)\cdot l+1}{2} \rceil = \frac{n}{t} + 1$.

Chapter 4

Reconstruction of DNA-words

4.1 Motivation and notation

In this chapter we study another version of the reconstruction problem of words. Let us recall the basic notation of 2.1 concerning the DNA-words:

Let $\mathcal{A} = \{\{\alpha_1, \bar{\alpha}_1\}; \{\alpha_2, \bar{\alpha}_2\}; ...; \{\alpha_q, \bar{\alpha}_q\}\}$ be an alphabet of q pairs of symbols (called *complement pairs*), the finite sequences composed from $\mathcal{A}$ are called *DNA-words.* Define $\bar{\bar{\alpha}}_i = \alpha_i$; and for a DNA-word $f = x_1 x_2 ... x_t$ over $\mathcal{A}$ let $\widetilde{f} = \bar{x}_t \bar{x}_{t-1} ... \bar{x}_1$ the *reverse complement* of f. Note that $\widetilde{(\widetilde{f})} = f$. Since we want to keep the basic properties of the structure of the DNA, and we need a generalization of the ordinary subword-relation we identify every DNA-word with its reverse complement. Let $\mathcal{A}^n$ denote the DNA-words of length n composed from $\mathcal{A}$ considering this identification.

We will say that the DNA-word g *precedes* the DNA-word f or g is a *substrand* of f $(g \prec f)$ if g is a subword of f or it is a subword of its reverse complement $\widetilde{f}$. We suggest to the reader to keep the difference between the subword $(\leq)$ and the substrand $(\prec)$ relations in mind in this section. Let $S(m, f)$ denote the set of all *different* DNA-words of length at most m, which precede f. Our main goal in this chapter is to determine for a given n the smallest m such that for every $f \in \mathcal{A}^n$ the set $S(m, f)$ uniquely defines f, i.e. f can be reconstructed from its m-deck.

The problem and definitions have molecular biological motivations, two potential applications are Digital Velcro and DNA computing (for more details see D'yachkov et al. [19]). DNA typically exists as paired, reverse-complementary words or *strands*: the Watson-Crick double helix, with its four letters, A, C, G and T paired via $\bar{A} = T$ and $\bar{C} = G$. Respective DNA-codes could involve the insertion-deletion metric — with bounded *similarity* between two strands: the length of the longest subword common either to the strands or common to one strand and the reverse complement of the other.

Another common task is to decide fast and effectively whether a given DNA-word (for example an erroneous gene, which causes illness) is present in the sample. For that job microarrays are used: ten thousands of relatively short DNA-words (called *probes*) are fixed on a glass sheet. The sample reacts with the probes, and it is evaluated which probes bond material from the sample. See Figure 4.1 as an illustration.

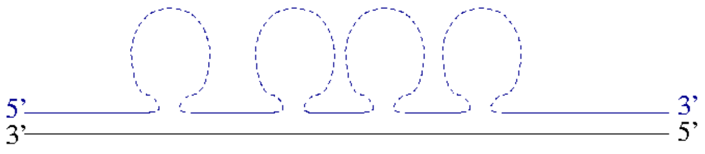


Figure 4.1: The bonding of a sample to a probe

On the figure we denote the probe with a black line and the sample with a blue line, such that the loops which are the substrands not bonding to the probe are denoted with dashed line. The blocks between two consecutive loops form a substrand of the sample bonding to the probe.

We want to model this process with our definition. One can argue that the physiochemical laws don't allow that each substrand of the long DNA-word can make the bond, but longer blocks are needed, and perhaps the number of blocks are also bounded. While these are perfectly legitimate objections, we try to make a step in that direction.

4.2 Main results

In this section we formulate our main results concerning the reconstruction of DNA-words. Since the proofs of the main theorems are rather technical and long, we present them in the following sections.

One can ask what is the difference between this and the original problems: here we may have more substrands but we do not know that the individual subwords belong to the DNA-word or to its reverse complement. This difference will be very clear already if our alphabet consists of one letter and its complement.

Let's consider the following example:

$$\mathcal{F}' = \bar{a}^{2k+\varepsilon}\, a^k \quad \text{and} \quad \mathcal{G}' = \bar{a}^{2k+\varepsilon-1}\, a^{k+1}, \tag{4.1}$$

where $\varepsilon \in \{0,1,2\}$ and $k \geq 1$ and $(k,\varepsilon) \neq (1,0)$. The length of both DNA-words are $3k+\varepsilon$. On the one hand the substrand $\bar{a}^{2k+\varepsilon}$ of $\mathcal{F}'$ satisfies $\bar{a}^{2k+\varepsilon} \not\prec \mathcal{G}'$. On the other hand it is easy to check that

$$S(2k+\varepsilon-1, \mathcal{F}') = S(2k+\varepsilon-1, \mathcal{G}').$$

This example suggests the following result:

Theorem 40 *Every DNA-word $f \in \{a, \bar{a}\}^*$ of length at most $3m-1$ is uniquely determined by its length and by the set*

$$D'(f) := S(2m, f).$$

The proof of this result can be found in Section 4.4.

The next example shows that if our DNA-words contain letters from more than one complement pairs then they are "easier" to determine. Consider the following DNA-words:

$$\mathcal{F} = \bar{a}^{2k+\varepsilon}\, \bar{b}\, b\, a^k \quad \text{and} \quad \mathcal{G} = \bar{a}^{2k+\varepsilon-1}\, \bar{b}\, b\, a^{k+1}, \tag{4.2}$$

where $\varepsilon \in \{0,1,2\}$ and $k \geq 1$ and $(k,\varepsilon) \neq (1,0)$. The length of both DNA-words are $3k+2+\varepsilon$. On the one hand the substrand $\bar{a}^{2k+\varepsilon}$ of $\mathcal{F}$ satisfies $\bar{a}^{2k+\varepsilon} \not\prec \mathcal{G}$. On the other hand it is easy to verify that

$$S(2k+\varepsilon-1, \mathcal{F}) = S(2k+\varepsilon-1, \mathcal{G}).$$

We have the following statement:

Theorem 41 *Every DNA-word $f \in \mathcal{A}^*$ of length at most $3m+1$ $(m > 1)$ containing both $(a$ or $\bar{a})$ and $(b$ or $\bar{b})$ is uniquely determined by its length and by the set*

$$D(f) := S(2m, f).$$

The examples *abab* and *abba* show that in case of $m = 1$ the statement is not true. The proof of this result can be found in Section 4.5.

We recall that, due to our definitions, the expression "uniquely determined" means "uniquely determined, up to reverse complementation". The statement deals with the case of $\varepsilon = 2$ in the examples.

4.3 Easy consequences and preliminary results

Applying the results in Section 4.2 we may derive some easy conclusions of them. For example, in the case when our DNA-words contain letters from one complement pair only, one may formulate the following result:

Corollary 42 *Every DNA-word $f \in \{a, \bar{a}\}^*$ of length at most n is uniquely determined by its length and by the set $S\left(\left\lceil \frac{2(n+2)}{3} \right\rceil, f\right)$.*

Proof: Let m be the smallest integer such that $n \leq 3m-1$. Then $\left\lceil \frac{2(n+2)}{3} \right\rceil \geq 2m$ and Theorem 40 applies. ∎

The situation is similar in the case of DNA-words containing letters from two complement pairs:

Corollary 43 *Every DNA-word $f \in \mathcal{A}^*$ of length at most n containing both $(a$ or $\bar{a})$ and $(b$ or $\bar{b})$ is uniquely determined by its length and by the set $S\left(\left\lfloor \frac{2(n+1)}{3} \right\rfloor, f\right)$.*

Proof: The statement is straightforward: let m be the smallest integer such that $n \leq 3m+1$. Then $\left\lfloor \frac{2(n+1)}{3} \right\rfloor \geq 2m$, therefore Theorem 41 applies. ∎

Let us recall the corresponding theorem for words:

Theorem 10 *Every word $w \in \Sigma_t^n$ is uniquely determined by its length and the $\lceil \frac{n+1}{2} \rceil$-deck consisting of its different subwords.*

It can be seen that Corollaries 42 and 43 are similar to Theorem 10. Indeed these corollaries can be considered as the analogues of Theorem 10 for DNA-words.

Perhaps the shortest proof of Theorem 10 is due to Sakarovitch and Simon (see [40], pp. 119–120.). Since in 4.4 and in 4.5 we were influenced by this nice proof, we present their proof here for a sake of completeness. They proved the following analogue of Theorems 40 and 41, from which Theorem 10 follows:

Theorem 44 *Every word $w \in \{a, b\}^*$ with at most $2m - 1$ letters is uniquely determined by its length and by the set of its different subwords of length at most m.*

Proof: Let us assume, to obtain a contradiction, that f and g are different words in $\{a, b\}^*$ such that $|f| = |g| \leq 2m - 1$ and let D denote the (common) set of their different subwords of length at most m. We further define $p = \max\{|s| : s \in D \cap a^*\}$ and $q = \max\{|s| : s \in D \cap b^*\}$.

We can suppose that $q \leq p$. Then clearly, $|f|_a \geq p$ and $|f|_b \geq q$, hence $2q \leq p+q \leq |f|_a+|f|_b = |f| \leq 2m-1$. Since q is an integer it follows that $q < m$, and this implies that $|f|_b = q$. Thus, there exist integers $i_0, i_1, ..., i_q, j_0, j_1, ..., j_q$, such that $f = a^{i_0}ba^{i_1}b...a^{i_{q-1}}ba^{i_q}$ and $g = a^{j_0}ba^{j_1}b...a^{j_{q-1}}ba^{j_q}$.

As $f \neq g$ by assumption, there exists a smallest k, such that $i_k \neq j_k$. W.l.o.g. we can assume that $i_k < j_k$. Since $|f| = |g|$, $i_l = j_l$ for $0 \leq l < k$ and $i_k < j_k$ it follows that $j_{k+1} + ... + j_q < i_{k+1} + ... + i_q$.

Let the words s and t be given by

$$s = a^{i_0+i_1+...+i_k+1}b^{q-k} \quad \text{and} \quad t = b^{k+1}a^{j_{k+1}+...+j_q+1}.$$

Clearly, s is a subword of g but not of f and t is a subword of f but not of g. Hence, due to the assumptions, it follows that $|s|, |t| > m$. Then $i_0+i_1+...+i_k+q-k \geq m$ and $k+1+j_{k+1}+...+j_q \geq m$. Summing these two inequalities and recalling that $i_l = j_l$ for $0 \leq l < k$, and that $i_k < j_k$ we have $j_0 + j_1 + ... + j_q + q = |g| \geq 2m$, a contradiction. Thus $i_k = j_k$ for every k, i.e. $f = g$. ∎

Previously we suspected that Corollaries 42 and 43 are not sharp and based on the examples above we thought the truth is the following two general statements:

- Each DNA-word of length at most $3m+\varepsilon$ containing only a or $\bar{a}$ is uniquely determined by its length and by the set $S(2m+\varepsilon, f)$.
- Each DNA-word of length at most $3m+2+\varepsilon$ containing both (a or $\bar{a}$) and (b or $\bar{b}$) is uniquely determined by its length and by the set $S(2m+\varepsilon, f)$.

As we mentioned above Theorem 40 and Theorem 41 are handling the case $\varepsilon = 2$ in presence of one or two complement pairs, respectively. However one can check it with simple calculations that in general case we cannot say better bound than the ones in Corollaries 42 and 43. The case $\varepsilon = 1$ gives the same bound in both cases, we can attain a slight (i.e. "1") improvements only in the case $\varepsilon = 0$. The analogue of Corollary 42 is the following:

Remark 45 *Every DNA-word $f \in \{a, \bar{a}\}^*$ of length $n = 3m$ is uniquely determined by its length and by the set $S\left(\left\lceil \frac{2(n+1)}{3} \right\rceil, f\right)$.*

The analogue of Corollary 43 is the following:

Remark 46 *Every DNA-word $f \in \mathcal{A}^*$ of length $n = 3m+2$ containing both (a or $\bar{a}$) and (b or $\bar{b}$) is uniquely determined by its length and by the set $S\left(\left\lfloor \frac{2n}{3} \right\rfloor, f\right)$.*

We will not prove both statements since they doesn't seem too relevant, however in 4.4.2 as a part of the proof of Theorem 40 we will get Remark 45.

If our DNA-words are self-reverse-complementary, then we are back to the original reconstruction problem of words :

Remark 47 *Let the DNA-words f and $g \in \mathcal{A}^*$ of length at most n be self-reverse-complementary, that is $f = \tilde{f}$ and $g = \tilde{g}$. Now if $S(\lceil \frac{n+1}{2} \rceil, f) = S(\lceil \frac{n+1}{2} \rceil, g)$ then $f = g$.*

Proof: If for the DNA-word w we have $w \prec f$ and $f = \tilde{f}$, then w is a subword of f and $\tilde{f}$ as well, i.e. $w \prec f$ if and only if $w \leq f = \tilde{f}$. Then we can apply Theorem 10 for ordinary words, which proves the statement. □

At the original reconstruction problem of words it was almost trivial, that if we know the result for the binary alphabet, then we have an answer at once for the case of k-element alphabets as well. The situation here is similar but the proof requires some work:

Theorem 48 *Theorem 41 remains valid if the DNA-word f contains letters from $k \geq 2$ different complement pairs.*

Proof: We use induction on the number k of different complement pairs present. The case of two pairs present is Theorem 41. Assume that the statement is valid for case of $k-1$ different pairs present. Let f and g be DNA-words with length $|f| = |g| \leq 3m+1$, and in both DNA-words there are k different complement pairs present. The alphabet is $\{\{a_1, \bar{a}_1\}, ..., \{a_k, \bar{a}_k\}\}$. Let $A_{1,2}, \bar{A}_{1,2}$ be a new pair of complement letters, and $f_{1,2}$ be the DNA-word derived from f by identifying all occurrences of a_1 and a_2 with $A_{1,2}$ and all occurrences of $\bar{a}_1$ and $\bar{a}_2$ with $\bar{A}_{1,2}$. The DNA-word $g_{1,2}$ is derived similarly. The new DNA-words contain letters from $k-1$ different pairs and $D(f_{1,2}) = D(g_{1,2})$. The induction hypothesis gives that $f_{1,2} = g_{1,2}$ (maybe we have to exchange the names of $g_{1,2}$ and $\widetilde{g}_{1,2}$). Furthermore, for the substrands $f^*_{1,2}$ and $g^*_{1,2}$ consisting of all occurrences of the letters $\{a_1, \bar{a}_1, a_2, \bar{a}_2\}$ we have $D(f^*_{1,2}) = D(g^*_{1,2})$ therefore we can apply Theorem 41, hence we have $f^*_{1,2} = g^*_{1,2}$ or $f^*_{1,2} = \widetilde{g}^*_{1,2}$.

In case of $f^*_{1,2} = g^*_{1,2}$ interleaving $f_{1,2}$ and $f^*_{1,2}$ we can determine f which is identical to g. In case of ($f_{1,2} = \widetilde{g}_{1,2}$ and $f^*_{1,2} = \widetilde{g}^*_{1,2}$) we can proceed similarly. However, it can happen that

$$\begin{aligned} f_{1,2} &= g_{1,2} \quad \text{but} \quad f_{1,2} \neq \widetilde{g}_{1,2} \quad \text{while} && (4.3)\\ f^*_{1,2} &\neq g^*_{1,2} \quad \text{but} \quad f^*_{1,2} = \widetilde{g}^*_{1,2}. && (4.4) \end{aligned}$$

The value $|f^*_{1,2}|$ cannot be odd, since otherwise $f_{1,2}(\frac{|f^*_{1,2}|+1}{2}) = g_{1,2}(\frac{|g^*_{1,2}|+1}{2})$, therefore $f^*_{1,2} = \widetilde{g}^*_{1,2}$ cannot occur. So let $|f^*_{1,2}| = \ell$ be even. From Condition (4.4) it follows that there is an index $j \leq \ell/2$ such that, say, $f^*_{1,2}(j) = a_1$, $g^*_{1,2}(j) = a_2$, while $f^*_{1,2}(\ell+1-j) = \bar{a}_2$ and $g^*_{1,2}(\ell+1-j) = \bar{a}_1$. From Condition (4.3) it follows that there is a subscript $i \leq (3m+1)/2$ such that, say, $f_{1,2}(i) = a_3$ (therefore $g_{1,2}(i) = a_3$ also holds) while $g_{1,2}(3m+2-i) = b$ where $b \neq \bar{a}_3$. If $b \in \{a_1, ..., a_k\}$ then introducing the new letters $B_1, \bar{B}_1, B_2, \bar{B}_2$, substitute all occurrences of a_1 and a_3 with B_1, all occurrences of $\bar{a}_1, \bar{a}_3$ with $\bar{B}_1$, all occurrences of the letters $a_2, a_4, ..., a_k$ with B_2, finally all occurrences of the letters $\bar{a}_2, \bar{a}_4, ..., \bar{a}_k$ with $\bar{B}_2$ in the original DNA-words. The result is the DNA-words f^B and g^B which satisfy the conditions of Theorem 41 while clearly $f^B \neq g^B$ and $f^B \neq \widetilde{g^B}$, a contradiction.

If, however, $b \in \{\bar{a}_1, \bar{a}_2, \bar{a}_4, ..., \bar{a}_k\}$ then we may define a bipartition of the alphabet, where letters b and a_3 belong to different classes, and letters a_1 and a_2 belong also to different classes. Then substitute all occurrences of the letters from the first class of the bipartition with $C_1, \bar{C}_1$ and the letters from the second class with $C_2, \bar{C}_2$, respectively. The new DNA-words clearly satisfy the conditions of Theorem 41, however the consequence of Theorem 41 does not hold. ■

This proof suggests that the existence of letters from more complement pairs decreases the necessary substrand length in the result.

Because our approach does not work for very short DNA-words, therefore we need the following help:

Remark 49 *Theorems 40 and 41 were tested by a computer program for short DNA-words (these are where $|f| \leq 15$ and if $|f| \leq 18$ and their structures are very special) and were found valid. Therefore in the proofs we can deal with long enough DNA-words only. That is important where our reasoning works above a (usually very small) bound.*

In the next two sections we prove our main results concerning the reconstruction of DNA-words. The general approach used is similar to the one in the proof of Theorem 48: find a subword of the DNA-word under investigation which distinguishes the DNA-word and its reverse complement from each other. Such a substrand in hand can identify the DNA-word itself. The greater the similarity between the DNA-word and its reverse complement, the harder to find such a substrand but, in exchange for this difficulty, we know more about the structure of the DNA-word.

4.4 The proof of Theorem 40 for DNA-words composed of one complement pair

Assume that f and g are DNA-words in $\{a, \bar{a}\}^*$ of the same length such that

$$|f| = |g| \leq 3m - 1 \quad \text{and} \quad D'(f) = D'(g) = D'.$$

Due to Remark 47 we may assume that f is not-self reverse complementary. Denote by $A(w)$ the number of a's in the DNA-word w, and define $\bar{A}(w)$ analogously. W.l.o.g. we may assume that both DNA-words f and g are written in the form where $A(f) \geq \bar{A}(f)$ and $A(g) \geq \bar{A}(g)$. At first assume that $A(f) > A(g)$, which also means that $\bar{A}(f) < \bar{A}(g)$. If $A(f) > 2m$ then take an arbitrary substrand g' of g such that $A(g'), \bar{A}(g') \geq \bar{A}(f)+1$. It is clear that $g' \not\prec f$. If, instead, $A(f) \leq 2m$ then take the substrand f' of f containing $A(g)+1$ a's. It is also clear that $f' \not\prec g$ and that $|f'|, |g'| \leq 2m$, which constitutes a contradiction. Therefore from now on in this proof we assume that we have

$$A := A(f) = A(g) \quad \text{and} \quad \bar{A} := \bar{A}(f) = \bar{A}(g). \tag{4.5}$$

Before we continue we recall one more notion: a DNA-word contains a *run* of length k when it contains k consecutive copies of a certain letter.

4.4.1 The case $\bar{A} < A$

In this case we know that $f \neq \tilde{f}$ and $g \neq \tilde{g}$, and each substrand of f or g containing at least $\bar{A}+1$ a's shows these inequalities. All substrands from $S(2m, f)$, containing at least $\bar{A}+1$ a's, are substrands of g, because they cannot be substrands of $\tilde{g}$ — and similarly for the substrands from $S(2m, g)$ the analogous statement holds.

Our DNA-words f and g can be written in the following form:

$$f := a^{I_0}\bar{a}\,a^{I_1}\bar{a}....\bar{a}\,a^{I_s} \quad \text{and} \quad g := a^{J_0}\bar{a}\,a^{J_1}\bar{a}....\bar{a}\,a^{J_s}$$

where $s = \bar{A}$, and any I_l or J_l can be zero. If $f \neq g$ then the subset

$$L := \Big\{ l \in \{0, ..., s\} \quad | \quad I_l \neq J_l \Big\}$$

has at least two elements. W.l.o.g. we may assume that $I_\ell = \min\{I_l, J_l \; : \; l \in L\}$, i.e. f contains a shortest run — of those indexed by L. Then consider the substrand g' of g containing all $\bar{a}$'s, in the ℓth run of a's containing at least $I_\ell + 1$ a's, finally it may contain other copies of a's so that altogether there are at least $\bar{A}+1$ a's. Then, due to the definition, g' is not a substrand of f, furthermore, by the number of a's, it is also clear that $\widetilde{g'}$ is also not a substrand of f. We know that

$$|g'| \leq \max\left\{\left(\left\lceil\frac{A}{2}\right\rceil - 1\right) + 1 + \bar{A}, \quad 2\bar{A} + 1\right\},$$

since the left argument of the maximum includes, within its parentheses, the largest possible value for I_k. If $|g'| \leq 2\bar{A} + 1 \leq 2m$ holds, then there is a contradiction. Therefore this method shows that $D'(f)$ and $D'(g)$ must be different while $\bar{A} + 1 \leq m$. Continuing the proof from now on (in this section) we assume that

$$\bar{A} > m - 1. \tag{4.6}$$

Hence, in this case

$$A = 3m - 1 - \bar{A} \leq 2m - 1. \tag{4.7}$$

Denote by $\bar{f}(a, \ell)$ the substrand of f containing all a's and the ℓth of $\bar{a}$'s. By our assumptions these are substrands of g, but, as we have just seen, not substrands of $\tilde{g}$. Therefore both f and g can be written in the following forms:

$$f = a^{r_0}\bar{a}^{s_1}a^{r_1}\bar{a}^{s_2}...\bar{a}^{s_t}a^{r_t} \quad \text{and} \quad g = a^{r_0}\bar{a}^{z_1}a^{r_1}\bar{a}^{z_2}...\bar{a}^{z_t}a^{r_t}, \tag{4.8}$$

where r_0 or r_t can be zero, while $r_1, ..., r_{t-1}$ and all s_i and z_i are non-zero.

Now we are going to show that for all i we also have $s_i = z_i$ (which, of course, implies that $f = g$).

Let $F \in \{x, y\}^*$ be an arbitrary word and assume it is written in the form

$$F = x^{r_0}y^{s_1}x^{r_1}y^{s_2}....y^{s_t}x^{r_t}, \tag{4.9}$$

where the runs are not empty (except, possibly, the very first and last). That is $r_0, r_t \geq 0$ and all other superscripts > 0.

Definition 50 *Let $F \in \{x, y\}^*$. A subword W of F is* well recognizable for the pair *x, y, or shortly well recognizable if one can reconstruct exactly which letter of W comes from which x- or y-runs of F.*

Note that reverse complementation is not taken into consideration now. Generally we will ensure separately that the well recognizable subword's reverse complement is not a subword of the original. It is clear that if the subword W' of F contains

W as a subword, then W' is also well recognizable. The subword F_1 containing one letter from each run is clearly well recognizable. Even better, if r_0 and r_t are both non-zero (or, oppositely, both zero), then the reverse complement of this subword is automatically not a subword of F. But when F has a big number of runs (say each run consists of one letter), then one can find much shorter well recognizable subwords.

Lemma 51 *Let $W(F)$ be the subword of F defined as follows:*
(I) *$W(F)$ retains at least one x from each x-run.*
(II) *If r_0 or $r_t > 1$ then $W(F)$ contains one x from the respective run and one y from the neighboring y-run.*
(III)*From all other x-runs with precisely two letters, let $W(F)$ contain both.*
(IV) *From all other x-runs with at least three letters $W(F)$ contains one x from the run and one y from both adjacent runs.*
(En1) *If between two previously chosen y's there are only two-letter x-runs, keep one x from each of these runs and take one element from each in-between y-run.*
(En2) *From every run of y's, remove all but one.*
Then the resulting $W(F)$ is a well recognizable subword of F for the pair x, y.

□

(The two last procedures make enhancement of the already constructed well recognizable subwords, that gives their different kind of names.) Lemma 51 may be thought of as an algorithm, whose six steps are applied sequentially in a single pass. Thus, its validity is evident. Let's remark that without operation (En1) the subword $W(F)$ would be still a well recognizable subword, but this operation decreases the number of letters with one with its every application. Note that $W(F)$ never has more letters than the total number of runs in f and it is never shorter than the number of x-runs. However, this construction is sensible for one-letter runs and in their presence it produces well recognizable subwords with fewer letters than the total number of runs.

Note also that any well recognizable subword of f in condition (4.8) is also a well recognizable subword of g.

Assume now that $f \neq g$, that is the series $s_1, ..., s_t$ and $z_1, ..., z_t$ are different. Then the set

$$L := \Big\{ l \in \{1, ..., t\} \quad \text{s. t.} \quad s_l \neq z_l \Big\}$$

has at least two elements, since the total number of $\bar{a}$'s are the same in both of our DNA-words. W.l.o.g. we may assume that $z_\ell = \min\{s_l, z_l \ : \ l \in L\}$. At first take the substrand f_1 of f containing all its a's and $z_\ell + 1$ $\bar{a}$'s from the ℓth $\bar{a}$-run:

$$\begin{array}{rcl} f &=& a^{r_0}\bar{a}^{s_1}a^{r_1}\dots\bar{a}^{s_\ell}\dots\bar{a}^{s_t}a^{r_t} \\ & & \downarrow \qquad \downarrow \ \dots \downarrow \ \dots \quad \downarrow \\ f_1 &=& a^{r_0+ \ \cdots \ +r_{\ell-1}}\bar{a}^{z_\ell+1}a^{r_\ell+ \ \cdots \ +r_t}. \end{array}$$

This DNA-word is clearly a well recognizable one, and, due to $A > \bar{A}$, its reverse complement is not a subword of f or g. Therefore, if $A + z_\ell + 1 \leq 2m$, then $f_1 \in D'(f)$ but $f_1 \notin D'(g)$, a contradiction.

If, however, this is not the case, then $|f_1| = 2m + \alpha$ and

$$\begin{aligned} A &= 2m + \alpha - (z_\ell + 1); \\ \bar{A} &= 3m - 1 - A = m - \alpha + z_\ell \end{aligned} \tag{4.10}$$

where $\alpha \geq 1$. By the minimality of z_ℓ there is another $\bar{a}$-run in f with at least z_ℓ elements. Therefore there are at most

$$t \leq 2 + \bar{A} - (2z_\ell + 1) = m + 1 - (z_\ell + \alpha) \tag{4.11}$$

$\bar{a}$-runs in the DNA-word f, and there is at most one more: that is, at most $m + 2 - (z_\ell + \alpha)$ a-runs in f.

Recall that the substrand f_1 is not in $D'(f)$ because it has α extra letters and $z_\ell \geq \alpha \geq 1$ (*viz.* (4.10)).

Assume at first that $r_0, r_t > 0$. Then consider the subword f_2 of the DNA-word f containing one letter from each run except the ℓth $\bar{a}$-run, which contains $z_\ell + 1$ $\bar{a}$'s:

$$\begin{array}{rcl} f &=& a^{r_0}\bar{a}^{s_1}a^{r_1}\dots\bar{a}^{s_\ell} \quad \dots\bar{a}^{s_t}a^{r_t} \\ & & \downarrow \ \downarrow \ \downarrow \ \ \dots \downarrow \quad \ \dots \downarrow \ \downarrow \\ f_2 &=& a \ \ \bar{a} \ \ a \ \ \dots\bar{a}^{z_\ell+1}\dots\bar{a} \ \ a. \end{array}$$

This DNA-word is well recognizable, and $\widetilde{f_2}$ is not a subword of f or g because there are not enough $\bar{a}$-runs in them. Furthermore, f_2 is also clearly not a subword of g, since in the ℓ-th $\bar{a}$-run there are too many letters. Due to (4.11) we know that

$$|f_2| \leq 1 + 2t + z_\ell \leq 1 + 2[m + 1 - (z_\ell + \alpha)] + z_\ell = 2m + 3 - 2\alpha - z_\ell \leq 2m,$$

since $z_\ell \geq \alpha \geq 1$. Therefore $f_2 \in D'(f)$ but $f_2 \notin D'(g)$, a contradiction.

If $r_0 = r_t = 0$ then we can repeat the previous reasoning since $\widetilde{f_2}$ is not a subword of f or g because there are not enough a-runs in them. If, say, $r_0 > 0$ and $r_t = 0$, then we cannot rule out that the reverse complement of f_2 is a subword of g. In this case there are precisely t $(\leq m + 1 - (z_\ell + \alpha))$ a-runs in f. We construct the subword f_3 of f as follows: it contains one letter from each run except the ℓth $\bar{a}$-run, which contains $z_\ell + 1$ $\bar{a}$'s. Then f_3 looks like f_2 but it has one less elements, due to $r_t = 0$. It is a well recognizable subword of f but not a subword of g. Its length is

$$|f_3| = 2t + z_\ell < |f_2|$$

therefore also $f_3 \in D'(f)$. In general it would yield a contradiction, but if $r_{t-\ell} > z_\ell$ then it still can happen that $\widetilde{f_3}$ is a subword of g. But then let f_4 be constructed from f_3 by adding z_ℓ more a letters to the $(t - z_\ell)$th a-run.

$$\begin{array}{rcccccccc}
f = & a^{r_0} & \bar{a}^{s_1} & a^{r_1} & \dots \bar{a}^{s_\ell} & \dots a^{s_{t-z_\ell}} & \dots & \bar{a}^{r_{t-1}} & a^{s_t} \\
 & \downarrow & \downarrow & \downarrow & \dots \downarrow & \dots \downarrow & & \downarrow & \downarrow \\
f_4 = & a & \bar{a} & a & \dots \bar{a}^{z_\ell+1} & \dots a^{z_\ell+1} & \dots & \bar{a} & a.
\end{array}$$

This f_4 is clearly a subword of f but not a subword of g or $\widetilde{g}$. Finally

$$|f_4| = |f_3| + z_\ell \leq 2m + 2 - 2\alpha \leq 2m.$$

Therefore $f_4 \in D'(f)$ but $\notin D'(g)$, a contradiction. The case $\bar{A} < A$ is proved.

4.4.2 The case $\bar{A} = A$

In this case we can prove a slightly stronger version of Theorem 40: we can suppose that $|f| \leq 3m$. Remember that this is the assumption of Remark 45 which will proven here at the same time.

Now $|f| = |g|$ is even, i.e. $m = 2k$ and the two DNA-words are of the form

$$f = a^{r_0}\bar{a}^{s_1}a^{r_1}\bar{a}^{s_2}....\bar{a}^{s_t}a^{r_t} \quad \text{and} \quad g = a^{R_0}\bar{a}^{z_1}a^{R_1}\bar{a}^{z_2}....\bar{a}^{z_T}a^{R_T}, \tag{4.12}$$

where $r_0 + \cdots + r_t = s_1 + \cdots + s_t = R_0 + \cdots + R_T = z_1 + \cdots + z_T = A = 3k$ and at least one from r_0, r_t and at least one from R_0, R_T is positive, otherwise we exchange the name of f and $\widetilde{f}$, and similarly for g as well. Now w.l.o.g. we may assume that $r_0 > 0$. Then in g we have $R_0 > 0$. Indeed, otherwise the subword $a\bar{a}^A$ of f does not precede g (since there are not enough $\bar{a}$'s after the first a in g, and not enough a's before the last $\bar{a}$ in $\widetilde{g}$).

If $r_t > 0$ also holds then consider the substrand $f_1 = \bar{a}^A a$. If $3k + 1 \leq 4k$ then $f_1 \in D'(f)$ but $\widetilde{f_1}$ is not a subword of g, since there are not enough a's after the first $\bar{a}$ in g. Therefore f_1 itself is a subword of g and we have $R_T > 0$, otherwise there are not enough $\bar{a}$'s before the last a in g. It also means that f_1 is a well recognizable subword of f and g as well. Therefore $r_t = 0 \Leftrightarrow R_T = 0$. (If, however, $|f| \leq 4$, then Remark 49 finishes the proof.)

Assume at first that

$$r_t, R_T > 0. \tag{4.13}$$

Denote by F_i the subword of f derived from f_1 by inserting one a from the ith a-run. If $A \geq 6$ then $F_i \in D'(f)$. These DNA-words together, for all i, describe the length of the $\bar{a}$-runs in f and all those runs are the complete union of some consecutive $\bar{a}$-runs in g. Repeating the process with g, yielding G_i's, we have the similar correspondence between the $\bar{a}$-runs of f and g. Therefore the $\bar{a}$-run structure of f and g are identical: $t = T$, and $s_i = z_i$ for $i = 1, ..., t$. (If $A \leq 5$ then Remark 49 finishes the proof.) Therefore our DNA-words are of the form

$$f = a^{r_0}\bar{a}^{s_1}a^{r_1}\bar{a}^{s_2}....\bar{a}^{s_t}a^{r_t} \quad \text{and} \quad g = a^{R_0}\bar{a}^{s_1}a^{R_1}\bar{a}^{z_2}....\bar{a}^{s_t}a^{R_t}. \tag{4.14}$$

Assume now that $f \neq g$, that is the series $r_0, ..., r_t$ and $R_0, ..., R_t$ are different. Then the set

$$L := \Big\{ l \in \{0, ..., t\} \quad \text{such that} \quad r_l \neq R_l \Big\}$$

has at least two elements, since the total number of a's is A in both DNA-words. W.l.o.g. we may assume that $R_\ell = \min\{r_l, R_l \ : \ l \in L\}$. Consider the subword $f_2 = \bar{a}^{s_1+\ldots+s_\ell}a^{R_\ell+1}\bar{a}^{s_{\ell+1}+\ldots+s_t}a$ of f This is clearly a subword neither of g nor of

$\widetilde{g}$. Therefore $A + R_\ell + 2 > 4k$, implying that $R_\ell \geq k - 1$. Due to the selection procedure for R_ℓ there is another a-run in f of length at least R_ℓ. Then all the other a-runs in f altogether contain at most $3k - (2R_\ell + 1)$ letters, hence the numbers of $\bar{a}$-runs are limited: $t \leq 3k - 2R_\ell$. Let the subword f_3 contain one letter from each different run in f, and contain R_ℓ more letters from the ℓth a-run. This DNA-word has at most $2(3k - 2R_\ell) + 1 + R_\ell = 6k - 3R_\ell + 1 \leq 3k + 4$ letters (here we used $R_\ell \geq k - 1$). Since f_3 is a substrand of f but does not precede g this is a contradiction (unless $k \leq 2$, when $|f| \leq 12$ and Remark 49 applies; or $k = 3$ and the length of DNA-word f's a-runs are $3, 2, 1, 1, 1, 1$ which allows again the use of Remark 49), proving Theorem 40 for this case.

From now on we assume that (4.13) does not hold: that is we have

$$r_t = R_T = 0. \tag{4.15}$$

(Let's recall that at that point we do not know whether the number of runs in f and g are equal or different.) Let $f(a;i)$ denote the subword of f containing all its a's and, furthermore, one $\bar{a}$ from the ith $\bar{a}$-run of f for $i = 1, ..., t$.

$$\begin{array}{rl} f = & a^{r_0}\bar{a}^{s_1}a^{r_1}\ldots\bar{a}^{s_i}\ldots\ \bar{a}^{r_{t-1}}a^{s_t} \\ & \downarrow \quad\ \downarrow \quad \ldots\downarrow \quad \ldots\downarrow \\ f(a;i) = & a^{r_0+\ \cdots\ +r_{i-1}}\ \bar{a}a^{r_i+\ \cdots\ +r_{t-1}}. \end{array}$$

Claim 52 *Every $f(a;i)$ is a subword of g or every $f(a;i)$ is a subword of $\widetilde{g}$ or both hold.*

Proof: Indeed, if every $f(a;i)$ is a subword of both DNA-words then there is nothing to prove. Therefore assume that there is an index i such that $f(a;i)$ is a subword of g but not of $\widetilde{g}$. Then for all indices $l \neq i$ the substrand $f(a;l)$ is also a subword of g. Indeed, if there is an index l, such that the substrand $f(a;l)$ was a subword of $\widetilde{g}$ but not of g, then consider the analogous subword $f(a;i,l)$ of f, containing altogether $A + 2$ letters (all a's and one letter from the ith and one from the lth $\bar{a}$-run). This would not be a subword either of g or $\widetilde{g}$, a contradiction, if $A \geq 6$ (if $A < 6$ then Remark 49 applies). The Claim is proved. □

Therefore we may assume that all $f(a;i)$ are subwords of g; therefore $t \leq T$, and one can make t groups $g_1^*, ..., g_t^*$ of consecutive a-runs in g such that the total length of a-runs within g_j^* is equal to s_j. Repeat the whole process for the subwords $g(a;i)$. It still can happen, that we must substitute $\widetilde{f}$ for f, but due to (4.15) this already implies that

$$t = T. \tag{4.16}$$

But from this equation it also follows that each $g(a;i)$ is a subword of f, since they are just the image in g of the subwords $f(a;i)$. Therefore we also have $r_i = R_i$ for all i.

Repeat the whole process now for the analogous subwords $f(\bar{a};i)$ of f. What we get is

$$\Big(s_i = z_i \text{ for all } i\Big) \quad \text{or} \quad \Big(s_i = R_{t-i} \text{ for all } i\Big).$$

In the first case we are done. Assume to the contrary that this is not the case. Then the second relation series holds. But repeating again the whole process for the analogous subwords $g(\bar{a},i)$ then we get that $z_i = r_{t-i}$ for all i, but since we have $r_i = R_i$ these imply that $s_i = z_i$ for all i. This contradicts our assumption and Theorem 40 is proved. ∎

4.5 The proof of Theorem 41 for DNA-words composed of two complement pairs

In this section, for the sake of simplicity, we will use the notation $\hat{a}$ for both a and $\bar{a}$ and $\hat{b}$ for both b and $\bar{b}$, when we don't care about the actual value of $\hat{a}$ or $\hat{b}$. With this notation every DNA-word of $\mathcal{A}^*$ can be considered as a DNA-word from $\{\hat{a},\hat{b}\}^*$ Assume that f and g are DNA-words in $\mathcal{A}^*$ of the same length such that

$$|f| = |g| \leq 3m + 1 \quad \text{and} \quad D(f) = D(g) = D. \tag{4.17}$$

Without loss of generality we also may assume, due to Remark 47, that at least one of the two DNA-words, say g, is not self-reverse complementary. Furthermore let

$$p = \max\Big\{|s| \ : \ s \in D \cap \hat{a}^*\Big\} \quad \text{and} \quad q = \max\Big\{|s| \ : \ s \in D \cap \hat{b}^*\Big\}.$$

W.l.o.g. we can assume that $q \leq p$. Let $f(a)$ denote the substrand of f consisting of all $\hat{a}$'s. The notations $f(b)$, $g(a)$, $g(b)$ are analogous. Then, by definition, $|f(a)| \geq p$ and $|f(b)| \geq q$, hence

$$2q \leq p + q \leq |f(a)| + |f(b)| = |f| \leq 3m + 1,$$

consequently $q \leq \frac{3m+1}{2} < 2m$ if $1 < m$. This implies that $|f(b)| = |g(b)| = q$. It also implies that $|f(a)| = |g(a)|$ holds. We remark that $|f(a)|$ can be bigger than p. (Note that if q is odd, then the substrands containing all $\hat{b}$'s are different from their reverse complements.)

Due to these properties there exist non-negative integers t, T; $i_0, ..., i_t$; $r_1, ..., r_t$; $j_0, ..., j_T$; and $R_1, ..., R_T$ such that

$$f = \hat{a}^{i_0}\hat{b}^{r_1}\hat{a}^{i_1}...\hat{b}^{r_t}\hat{a}^{i_t} \quad \text{and} \quad g = \hat{a}^{j_0}\hat{b}^{R_1}\hat{a}^{j_1}...\hat{b}^{R_T}\hat{a}^{j_T}, \tag{4.18}$$

where $t \neq T$ can happen and where i_0, i_t, j_0, j_T can be zero, while all other superscripts are nonnegative integers, furthermore $i_0 + ... + i_t = j_0 + ... + j_T = |f(a)|$ and $r_1 + \cdots + r_t = R_1 + \cdots + R_T = |f(b)|$. Since $q \leq 2m$, the substrands $f(b)$ and $g(b)$ belong to $S(2m, f) = D$; therefore $f(b) = g(b)$ or $f(b) = \widetilde{g}(b)$, or both. Let's remark that we have our general form (4.9) with letters $\hat{a}$ and $\hat{b}$; therefore Lemma 51 applies for these DNA-words.

For two DNA-words w and u denote by $w \simeq u$ that $w \prec u$ and $u \prec w$. The following observation will be useful later.

Lemma 53 *Let f and g DNA-words of form(4.18). Assume that $T = t$, $i_k = j_k$ for $k = 0, ..., t$ and $r_l = R_l$ for $l = 1, .., t$, furthermore $f(a) \simeq g(a)$ and $f(b) \simeq g(b)$. Then $f \simeq g$.*

Proof: Suppose to the contrary that $f \neq g$ and $f \neq \widetilde{g}$. We can obtain f by interleaving the runs of $f(a)$ and $f(b)$. Since $f \neq g$ it easy to see that we must get g from the runs of $\widetilde{f(a)}$ and $f(b)$. If at least one of $f(a)$ or $f(b)$ is self-reverse complementary, then we get $f = \widetilde{g}$ or $f = g$, a contradiction. Suppose now that $f(a) \neq \widetilde{f(a)}$ and $f(b) \neq \widetilde{f(b)}$. Then due to Theorem 10 there exists a subword a_*, of length at most $\lceil (|f(a)| + 1)/2 \rceil$, such that, say, $a_* \leq f(a)$, but $a_* \not\leq \widetilde{f(a)}$. We get b_* of length at most $\lceil (|f(b)| + 1)/2 \rceil$ similarly. Now let f_* be the DNA-word made by interleaving a_* and b_*. Clearly $f_* \prec f$ but $f_* \not\prec g$. Hence if $|f| > 7$, then $|f_*| \leq \lceil (|f(a)|+1)/2 \rceil + \lceil (|f(b)|+1)/2 \rceil = \lceil (f+2)/2 \rceil = \lceil (3m+3)/2 \rceil \leq 2m$, a contradiction. (The cases $|f| \leq 7$ are covered by Remark 49.) □

In the sequel we are going to show that the conditions of Lemma 53 hold.

At first we show that the run structures in $f(b)$ and in at least one of $g(b)$ and $\widetilde{g}(b)$ are identical. Denote by $f(b;\ell)$ the substrand consisting of all its $\hat{b}$'s and one letter from the ℓth $\hat{a}$-run:

$$\begin{array}{rl} f = & \hat{a}^{i_0}\hat{b}^{r_1}\hat{a}^{i_1} \ \dots \ \hat{a}^{i_\ell} \dots \hat{b}^{r_t}\hat{a}^{i_t} \\ & \quad\ \downarrow \qquad\ \dots \ \downarrow \ \dots \downarrow \\ f(b;\ell) = & \hat{b}^{r_1+\ \cdots\ +r_{i-1}}\hat{a}\hat{b}^{r_i+\ \cdots\ +r_t}. \end{array}$$

Since $|f(b;\ell)| \leq 2m,\ 1 < m$, this belongs to $D(f) = D(g)$.

Claim 54 *Every $f(b;\ell)$ is a subword of g or every $f(b;\ell)$ is a subword of $\widetilde{g}$ or both hold.*

Proof: Indeed, if every $f(b;\ell)$ is a subword of both DNA-words then there is nothing to prove. Therefore assume that for a particular k the DNA-word $f(b;k)$ is a subword of, say, g but not of $\widetilde{g}$. Then for all ℓ the DNA-words $f(b;\ell)$ are subwords of g as well. Indeed, if there is a $j \neq k$ such that $f(b;j)$ is a subword of $\widetilde{g}$ but not of g, then the f-subword $f(b;k,j)$, defined analogously, is not a subword of either g nor of $\widetilde{g}$.

Because $|f(b;k,j)| \leq (3m+1)/2+2$, this yields a contradiction for $5 \leq m$. (The cases $m \leq 5$ are covered by Remark 49.) The Claim is proved. □

So we can assume that every $f(b;\ell)$ is a subword of, say, g. Therefore $t \leq T$ and one can make t groups $g_1^*, \ldots, g_t^*$ of consecutive $\hat{b}$-runs in g such that the total length of the $\hat{b}$-runs within g_j^* is equal to r_j. Repeat the whole process for the subwords $g(a;i)$. It still can happen, that we had to substitute $\widetilde{f}$ for f, but this already implies that $t = T$. But from this equation it also follows that each $g(a;\ell)$ can be chosen to be a subword of f, since, as we know, the subwords $f(a;i)$ can be found in g. Therefore we also have $r_i = R_i$ for all i and

$$f = \hat{a}^{i_0}\hat{b}^{r_1}\hat{a}^{i_1}\ldots\hat{b}^{r_t}\hat{a}^{i_t} \quad \text{and} \quad g = \hat{a}^{j_0}\hat{b}^{r_1}\hat{a}^{j_1}\ldots\hat{b}^{r_t}\hat{a}^{j_t} \tag{4.19}$$

where the $\hat{b}$-runs with the same superscripts are identical. Furthermore, we also know that the number of non-empty $\hat{a}$-runs in f and g are equal as well. Indeed, if the multiset $\{i_0, i_r\}$ has no fewer non-zero elements than the multiset $\{j_0, j_r\}$ then the DNA-word containing one $\hat{a}$ from the nonempty runs indexed by the first multiset and $f(b)$ establishes this relation. Therefore the number of non-empty $\hat{a}$-runs in f and g are the same, say r', equal to $t-1$, t or $t+1$.

It remains to prove that $f(a) \simeq g(a)$ and g can be written in form that $i_k = j_k$ for all possible k. (Remark that if one must interchange g and $\widetilde{g}$ then we will show that in that case $f(b) = \widetilde{f(b)}$.)

4.5.1 The case $q = 1$

We start with the special case $q = 1$. Now w.l.o.g. we may assume that both DNA-words are written in form where $\hat{b} = b$ (otherwise we can take the reverse complement form of the DNA-word). Now any substrand of f containing the letter b should be contained in g in its original form because changing the substrand into its reverse complement would change b into $\bar{b}$. Since $|f(a)| = |g(a)|$, $i_0 + i_1 = j_0 + j_1$.

If the multisets $\{\{i_0, i_1\}\}$ and $\{\{j_0, j_1\}\}$ were different, then there would exist a unique smallest element within them, say, the i_1 : we have $i_0 > j_0, j_1 > i_1$. Take a subword u of g of the form

$$u = b\hat{a}^{i_1+1}.$$

This subword clearly does not precede f (there are not enough $\hat{a}$'s after b in the DNA-word f). Since $|u| \le (3m+1)/2 \le 2m,\ \ 1 < m$, therefore $D(f) \neq D(g)$, a contradiction. The ordered pairs (i_0, i_1) and (j_0, j_1) coincide. Denote by f_0 the longest simple subword of f finishing with b, and by f_1 the longest subword of f starting with b. The definitions of g_0 and g_1 are similar. Now f_0 and g_0 are DNA-words of the same length, and all their substrands of length $\le 2m$, ending with b coincide as well. Denote by f_0^* and g_0^* the same DNA-words without their b terminus. Then we know that all subwords of length $\lceil (|f_0^*| + 1)/2 \rceil$ of f_0^* and g_0^* are the same above the alphabet $a, \bar{a}$, in the simple subword relation. Application of Theorem 10 gives that $f_0^* = g_0^*$ in the original ordering. Furthermore, the same applies for f_1^* and g_1^*, therefore we have proven that $f = g$.

From now on we assume that $1 < q \leq (3m+1)/2$. Therefore $|f(a)| = 3m+1-q \leq 3m-1$. Now considering the elements $\hat{a}^k \in D$ and applying Theorem 40 we get that

$$f(a) \simeq g(a).$$

Our only remaining goal is to prove that the $\hat{a}$-structure of the DNA-words are the same, i.e. $i_k = j_k$ for all k. We will examine two cases concerning the size of q.

4.5.2 The case $1 < q \leq m+1$

Lemma 55 *If $1 < q \leq m+1$ and there are two indices $\ell \in \{0, ..., t\}$ for which*

$$q + i_\ell > 2m, \tag{4.20}$$

then we have $t = 2$, $q = m+1$, $i_0 = i_1 = j_0 = j_1 = m$.

Proof: Indeed, if $q \leq m$ and if there are two distinct indices $k \neq l$ satisfying (4.20) then

$$q + i_l + q + i_k \geq 2m + 1 + 2m + 1,$$

therefore

$$q + i_l + i_k \geq 4m + 2 - q \geq 3m + 2 > |f|,$$

a contradiction.

If, however, $q = m+1$ and $i_0 = i_1 = m$ then $j_0 = j_1$ as well. Otherwise we would have, say, $j_0 < i_1 < j_1$. Then a g-substrand consisting of one letter from the middle $\hat{b}$-run, and $i_1 + 1$ letters from the j_1-run is clearly shorter than $2m$ but does not precede f, a contradiction. Let's remark that in this case Lemma 53 is applicable directly, and Theorem 41 is proved. □

If there is precisely one index ℓ satisfying (4.20), then the corresponding run will be called a *long* run, while the other runs are called *short*. Denote by $f^*(b; k)$ the f-substrand consisting of all its $\hat{b}$'s and the complete kth $\hat{a}$-run. For short runs the length of these substrands is at most $2m$, therefore these belong to $D(f) = D(g)$. Assume for a moment that $f(b) = g(b) \neq \widetilde{g(b)}$. Then $f^*(b; k)$ is not a subword of $\widetilde{g}$ for any short run, therefore we can find equality of the lengths of the short runs, i.e. $i_k = j_k$ for short runs. Furthermore, because of Lemma 55 (i) there is only one $\hat{a}$-run (the ℓ-th), whose length can not be ascertained from the substrands, but then $|i_\ell| = (3m+1-q) - \sum_{k\neq\ell} |i_k| = (3m+1-q) - \sum_{k\neq\ell} |j_k| = |j_\ell|$, which completes the proof in this case. Therefore from now on we assume that

$$f(b) = g(b) = \widetilde{g(b)}$$

holds as well.

Now we analyze two cases based on the presence of a long run.

Case 1: Assume at first that there is a long run in the DNA-word f and this is the ℓth one. Then g also has at least one long run. Indeed, let u_1 denote an $(2m-q)$-letter substrand of the long run. Then the f-substrand $f(b) \cup u_1$ belongs to $D(g)$, and the image of u_1 is contained in a long $\hat{a}$-run of g. However, g cannot contain two long runs, otherwise Lemma 55 would apply, a contradiction. Therefore g contains exactly one long run and we may assume that f and g contain their respective long runs at the same index ℓ. Let's assume now that $\ell \neq t-\ell$. Then denote f^*_ℓ the substrand containing everything except the ℓth and $(t-\ell)$th $\hat{a}$-runs. This has at most $2m$ letters, and therefore belongs to $D(f)$: that is it precedes the analogously defined g-substrand g^*_ℓ. Similarly g^*_ℓ precedes f^*_ℓ. Consequently we know that $f^*_\ell \simeq g^*_\ell$. That can mean that **(a)** $f^*_\ell = g^*_\ell$ or **(b)** $f^*_\ell = \widetilde{g^*_\ell}$ or both. But all three possibilities show us that $i_\ell + i_{t-\ell} = j_\ell + j_{t-\ell}$. If **(b)** does not hold then there is a $k \neq \ell, t-\ell$ such that $\widetilde{f}(b;k)$ is not a subword of $g(b;t-k)$. But since $i_{t-k} \neq 0$ therefore the substrand $f(b;k,t-\ell)$ (it consists of all $\hat{b}$'s and one element of the kth and one element of the $(t-\ell)$th $\hat{a}$-runs each) which is not longer than $2m$, is a substrand of $g(b;k,t-\ell)$, and vice versa, which shows that Lemma 53 is applicable. If, however, **(b)** holds but **(a)** does not, then there is a k such that $f(b;k)$ is not a substrand of $g(b;k)$. Then let u denote an $2m-q-i_k$ element substrand of the long run in f. Let f' be the DNA-word consisting of u and $f(b;k)$. This is not a subword of g but also not a subword of $\widetilde{g}(b;t-k,t-\ell)$ unless q is very close to m and $j_{t-\ell}$ is also close to m. But then we have a small run-number r and then there is a well recognizable subword of f with at most $2r+1$ letters and repeating the previous reasoning we get the contradiction.

We came now to the case when $\ell = t - \ell$ and t is odd. But then if f_ℓ^* has at most $2m$ letters, which allows us to show as before that $f_\ell^* \simeq g_\ell^*$, and then we can apply Lemma 53 again. If this is not the case then we have $q = m+1$ and $i_\ell = m$. If we have at least four non-empty $\hat{a}$-runs then for all $k \neq \ell$ we have $f(b; k, t-k) \simeq g(b; k, t-k)$, showing that $i_\ell = j_\ell$. Furthermore, it is impossible, as usual, that for k_1, k_2 we have $f(b; k_1, t-k_1) = g(b; k_1, t-k_1)$ while $f(b; k_2, t-k_2) = g(b; k_2, t-k_2)$. (We can use the previous technique again.) So Lemma 53 is applicable again.

Case 2: Next suppose that there is no long run. Then all $f(b; k) \in D(f) = D(g)$. Assume that for all k the substrand $f(b; k, t-k)$ has length $\leq 2m$. Then for all k we have $f(b; k, t-k) \simeq g(b; k, t-k)$. Even more, as usual, we can show that if there is a k such that $f(b; k, t-k)$ is equal to $g(b; k, t-k)$ but not to its reverse complement, then for all other $l \neq k$ we also have $f(b; l, t-l) = g(b; l, t-l)$. Indeed, if this is not the case then there is a substrand f_1 of $f(b; k, t-k)$ with at most $\lceil (i_k + i_{t-k})/2 \rceil$ letters from its $\hat{a}$-runs showing that $f(b; l, t-l) \neq \widetilde{g}(b; l, t-l)$. Similarly, there is a substrand f_2 of $f(b; l, t-l)$ with at most $\lceil (i_l + i_{t-l})/2 \rceil$ letters from its $\hat{a}$-runs showing that $f(b; l, t-l) \neq g(b; l, t-l)$. Putting together these two substrands we get a DNA-word from $D(f)$ which does not belong to $D(g)$, a contradiction, except if $q = m+1$ and both $\hat{a}$-run pairs contain exactly $m-1$ letters, where m is odd. But again, then we can find a well recognizable DNA-word with ten letters, and repeating the whole process we are done.

So what remains is that we have an ℓ such that $q + i_\ell + i_{t-\ell} > 2m$. Then for all other $k \neq \ell, t-\ell$ we have $f(b; k, t-k) \simeq g(b; k, t-k)$. (Otherwise again we have four non-empty $\hat{a}$-runs, and finding a well recognizable DNA-word with eight letters finishes the proof.) Now again we can show that, say, $f(b; k, t-k)$ is equal to $g(b; k, t-k)$. Of course, we also get from it that $i_\ell + i_{t-\ell} = j_\ell + j_{t-\ell}$. Then the multisets $\{i_\ell, i_{t-\ell}\}$ and $\{j_\ell, j_{t-\ell}\}$ are the same. Otherwise there would be a clear maximum, say i_ℓ and then $f(b; i_\ell)$ does not precede g, a contradiction. So we are done except if $i_\ell = j_{t-\ell} \neq j_\ell = i_{t-\ell}$. But then if for all $k \neq \ell, t-\ell$ we have $f(b; k, t-k) = \widetilde{g}(b; k, t-k)$ then we can apply Lemma 53 getting that $f = \widetilde{g}$ or there is a k which does not satisfy this. But then, as usual, we can construct a substrand of f with $\lceil (i_k + i_{t-k})/2 \rceil + \lceil (i_\ell + i_{t-\ell})/2 \rceil$ letters from the respective $\hat{a}$-runs which does not precede g, a contradiction; except if again those four runs contain all the $\hat{a}$'s. But then again we can construct a well recognizable DNA-word of length at most, say, 10, repeating the reasoning. So the case $1 < q \leq m+1$ is solved.

4.5.3 The case $q > m + 1$

In this case we have $p = |f(a)| \leq 2m - 1$. Therefore any substrand f_k consisting of $f(a)$ and an arbitrary letter from the kth $\hat{b}$-run belongs to $D(f)$. If $f(a) \neq \widetilde{f(a)}$ then it also means that for all k the substrand f_k is a subword of g, therefore for all k we have $i_k = j_k$. Lemma 53 finishes the proof.

So we may assume that $f(a) = \widetilde{f(a)}$. Suppose that there is a k such that f_k is a subword of g but not of $\widetilde{g}$. Assume furthermore that there is an ℓ such that f_ℓ is a subword of $\widetilde{g}$ but not of g. (If this second substrand does not exist then we already have that the lengths of the $\hat{a}$-runs in f and g are identical.) Let $f_{k,\ell}$ denote the union of the former two subwords, then it is a subword of f but not a subword either of g nor of $\widetilde{g}$. If $q > m + 2$ then $f_{k,\ell} \in D(f)$ therefore it is a contradiction and we are done. But $q = m + 2$ can not be true, otherwise $p = 2m - 1$ would hold, therefore $f(a) \neq \widetilde{f(a)}$, a contradiction. Theorem 41 is fully proved. ■

4.6 Open problems

Problem 56 *Reconstruction from the k-deck of the multiset of substrands.*

As we seen in the Introduction, in the case of reconstruction problem of words the exact bounds for multiset-reconstruction are still unknown. One can ask the similar question for DNA-words as well. However, it seems less relevant from the practical point of view, it is rather a problem of theoretical mathematics than of applied mathematics or bioinformatics.

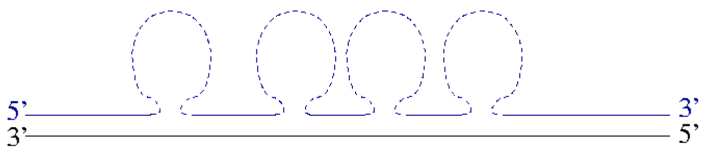
Furthermore one idiosyncrasy of the substrand relation (i.e. the identification of every DNA-word with its reverse complement) complicate the problem, since to handle the multiplicities of the substrands becomes a nontrivial problem.

Problem 57 *Finding a polynomial (or being optimistical, a linear) algorithm.*

Although, we use the terminology "reconstruction“, the results of this chapter tell when a DNA-word is uniquely determined by its substrands. Though our method contains a few algorithmic elements, a fast polynomial algorithm for reconstruction of DNA-words will be most welcome as well.

Problem 58 *Approximate the mathematical model to the real world.*

Let us recall Figure 4.1 to illustrate this problem:



As we mentioned above, the physiochemical laws of the DNA double-helix make restrictions to the length and the number of the blocks (i.e. the lines between two consecutive dashed loops). Hence in a more exact mathematical model there are two parameters L and N, and our aim is to reconstruct a DNA-word from its m-deck such that the elements of the m-deck have to satisfy that the length of they blocks is at most L and the number of the blocks is at least N.

Unfortunately these assumptions make the problem very hard to handle. Some examples and computer tests made by the authors gave nothing useful until this moment.

Chapter 5

Reconstruction of matrices

5.1 Motivation and notation

Let Σ be a finite alphabet and $\Sigma^{n\times n}$ the set of all $n \times n$ matrices over Σ. For $B \in \Sigma^{k\times k}$ and $A \in \Sigma^{n\times n}$ we say that B is a *symmetric submatrix* of A if we can get B by deleting $n-k$ rows and columns of A *symmetrically.* Denote by $\mathrm{sym}M_k(A)$ the multiset of all the $\binom{n}{k}$ submatrices of A of size $k \times k$.

We can consider the case of *non-symmetric* deleting of rows and columns as well, then let $M_k(A)$ denote the multiset of all the $\binom{n}{k}^2$ submatrices of A of size $k \times k$ with this submatrix relation. Let us define the *sum-matrix* of the matrix A as the sum of the elements of the k-deck of the respective multisets in both cases: $\mathrm{sym}\Sigma_k(A) = \sum_{B\in \mathrm{sym}M_k(A)} B$ and $\Sigma_k(A) = \sum_{B\in M_k(A)} B$.

In this chapter we consider the following reconstruction problems:

Problem 59 *For a given n what is the smallest k such that every $A \in \Sigma^{n\times n}$ is uniquely determined by* $\mathrm{sym}M_k(A)$ *or by $M_k(A)$, i.e. the k-deck consisting of the multiset of its submatrices of size $k \times k$.*

Problem 60 *For a given n what is the smallest k such that every $A \in \Sigma^{n\times n}$ is uniquely determined by* $\mathrm{sym}\Sigma_k(A)$ *or by $\Sigma_k(A)$, i.e. its sum-matrix of order k.*

We will use the term *k-equivalent* for matrices A and B in the case of symmetric or non-symmetric deletions if $\mathrm{sym}M_k(A) = \mathrm{sym}M_k(B)$ or $M_k(A) = M_k(B)$, resp. Similarly we say that matrices A and B are *additively k-equivalent* in the case of symmetric or non-symmetric deletions if $\mathrm{sym}\Sigma_k(A) = \mathrm{sym}\Sigma_k(B)$ or $\Sigma_k(A) = \Sigma_k(B)$, resp.

Beyond its theoretical interest, Problem 59 has a connection to the famous (vertex) graph reconstruction problem of Kelly [31] and Ulam [60]: it is equivalent with the special case of ordered graphs or ordered bipartite graphs, respectively.

An *ordered graph* on n vertices is a graph with a linear ordering on its vertices which is inherited by its subgraphs. A bipartite graph is ordered if both of the the color-classes of the graph are ordered independently. Using 0-1 matrices and ordered graphs, Tardos [58] settled a series of conjectures and gave new proofs for several problems.

The vertex-reconstruction of ordered (bipartite) graphs is a special case of reconstruction of square matrices with (non-)symmetric deletions by considering the adjacency matrix of the (bipartite) graph.

By the following lemma, the symmetric analogue of Lemma 16, claims that it is enough to examine the problems over the binary alphabet.

Lemma 61 *Every $A \in \Sigma^{n\times n}$ is uniquely determined by $\mathrm{sym}M_k(A)$ for $|\Sigma| \geq 2$ iff every $A \in \{0,1\}^{n\times n}$ is uniquely determined by $\mathrm{sym}M_k(A)$.* □

Contrary to the case of words, the reconstruction problem of matrices was not examined widely. The only one result is due to Manvel and Stockmeyer [43] proving the reconstruction from $\mathrm{sym}M_{n-1}(A)$, i.e. the reconstruction from the $n-1$-deck consisting of its symmetric submatrices.

5.2 The limitation of the method

First we present negative results concerning the reconstruction from the sum-matrix (i.e. Problem 60), which show the limitation of our method via proving the existence of additively k-equivalent matrices:

Theorem 62 *(a) If $k < \dfrac{n^{2/3}}{\sqrt[3]{2\log_2(n+1)}}$ then there exist matrices $A, B \subset \{0,1\}^{n\times n}$ such that $A \neq B$ but $\Sigma_k(A) = \Sigma_k(B)$.*

(b) If $k < \dfrac{n^{2/3}}{\sqrt[3]{\log_2(n+1)}}$ *then there exist matrices* $A, B \subset \{0,1\}^{n\times n}$ *such that* $A \neq B$ *but* $\mathrm{sym}\Sigma_k(A) = \mathrm{sym}\Sigma_k(B)$*.*

Proof: (a) For an arbitrary 0-1 matrix A of size $n \times n$, the matrix $\Sigma_k(A)$ is the sum of $\binom{n}{k}^2$ submatrices of A so each entry in $\Sigma_k(A)$ is a nonnegative integer not exceeding $\binom{n}{k}^2$. Hence,

$$\left|\left\{\Sigma_k(A) :\ H \in \{0,1\}^{n\times n}\right\}\right| \leq \left(\binom{n}{k}^2 + 1\right)^{k^2} \leq (n+1)^{2k^3} < 2^{n^2} = \left|\{0,1\}^{n\times n}\right|.$$

So the number of possible values of $\Sigma_k(A)$ is less than the number of 0-1 matrices.

(b) Similarly to the nonsymmetric case, each entry of $\mathrm{sym}\Sigma_k(A)$ is at most $\binom{n}{k}$ and therefore

$$\left|\left\{\mathrm{sym}\Sigma_k(A) :\ H \in \{0,1\}^{n\times n}\right\}\right| \leq \left(\binom{n}{k} + 1\right)^{k^2} \leq (n+1)^{k^3} < 2^{n^2} = \left|\{0,1\}^{n\times n}\right|.$$

■

Remark 63 *With more careful computation the conditions can be improved to* $k < \left(\sqrt[3]{\dfrac{3}{2}} - \varepsilon\right)\dfrac{n^{2/3}}{\sqrt[3]{\log_2 n}}$ *and* $k < \left(\sqrt[3]{3} - \varepsilon\right)\dfrac{n^{2/3}}{\sqrt[3]{\log_2 n}}$*, respectively.*

Since this statement shows the limitation of the method only, we do not deal with the determination of the exact constants.

5.3 Rephrasing the reconstruction problem

From now on our aim is to give an upper bound for the reconstruction Problems 59 and 60. In this section we give a generalization of the ideas of Krasikov and Roditty [32].

5.3.1 The symmetric case

We will index the rows and the columns of the matrices from 0 to $n-1$ and let $A = (a_{ij})$. Note that in the case of symmetric deletions an element of the lower(or upper)-triangle of a matrix remains in the lower(or upper)-triangle of its submartix and the elements of the diagonal also remain in the diagonal, hence we handle these cases separately. It is easy to see that the case of the diagonal leads back to the case of reconstruction of words examined by Krasikov and Roditty [32] (see Theorem 8), therefore we eliminate the formulas concerning the diagonal here. The case of lower and the upper triangle can be handled similarly, here we consider the upper triangle only.

Lemma 64 *Let $A \in \{0,1\}^{n\times n}$ and let $S_{ij}(A)$ denote the total number of occurrences of 1's in the position of the i-th row and j-th column of the elements of* $\mathrm{sym}M_k(A)$*. (Clearly $S_{ij}(A) = \mathrm{sym}\sum_k(A)_{ij}$.) Then for $i,j = 0,\dots,k-1$ and $i<j$*

$$S_{ij}(A) = \sum_{l=0}^{n-1}\sum_{m=l+1}^{n-1}\binom{l}{i}\binom{m-l-1}{j-i-1}\binom{n-m-1}{k-j-1}a_{lm}.$$

Proof: Suppose that $a_{lm} = 1$, then we need to count the cases when this element becomes a 1 of the ith row and jth column in a matrix of $\mathrm{sym}M_k(A)$. Note that we need to handle the case $l < m$ only because an element of the lower-triangle of a matrix remains in the lower-triangle of its submartix. (If we allow non-symmetric deleting of rows and columns this statement is not true.) We attain such a case if we choose row and column indices by the following way: first choose i indices from the first l ones, then $j-i-1$ indices from the indices from $l+1,\dots,m-1$, and the remaining $k-j-1$ indices from the indices $m+1\dots n-1$. This ensures $\binom{l}{i}\binom{m-l-1}{j-i-1}\binom{n-m-1}{k-j-1}$ 1's in $\mathrm{sym}M_k(A)$ for a given pair l,m, summing up all these expressions for every $l<m$ finishes the proof.

□

Let A and B be two k-equivalent matrices and let $\Delta = (\delta_{ij})$ be a matrix over the alphabet $\{-1,0,1\}$ where $\delta_{ij} = a_{ij} - b_{ij}$. From Lemma 64 we get the following:

Lemma 65

$$\sum_{l=0}^{n-1}\sum_{m=l+1}^{n-1}\binom{l}{i}\binom{m-l-1}{j-i-1}\binom{n-m-1}{k-j-1}\delta_{lm}=0, \qquad 0\le i<j\le k-1,$$

□

Consider now the following polynomials of two variables:

$$p_{ij}(x,y)=\binom{x}{i}\binom{y-x-1}{j-i-1}\binom{n-y-1}{k-j-1}$$

for $i<j$ and $i,j=0,\dots,k-1$. It is easy to see that every p_{ij} is a polynomial of total degree $k-2$, furthermore $p_{ij}(x,y)=0$ for $0\le x<i$, $n+j-k<y\le n-1$ and $0<y-x<j-i$.

Lemma 66 *For every fixed pair of integers n and k, the set $\{p_{ij}(x,y) : i<j,\ i,j=0,\dots,k-1\}$ forms a basis of the vectorspace of polynomials of total degree at most $k-2$ in two variables.*

Proof: Consider

$$r(x,y)=\sum_{i=0}^{k-1}\sum_{j=i+1}^{k-1}\lambda_{ij}p_{ij}(x,y)=\sum_{i=0}^{k-1}\sum_{j=i+1}^{k-1}\lambda_{ij}\binom{x}{i}\binom{y-x-1}{j-i-1}\binom{n-y-1}{k-j-1}.$$

Now suppose that $r(x,y)\equiv 0$, we will prove that $\lambda_{ij}=0$ for all $i<j$. Suppose to the contrary that $\lambda_{\alpha\beta}\neq 0$ and this coefficient is the first one in lexicographical order.

Then we get $0=r(\alpha,\beta)=\lambda_{\alpha\beta}\binom{n-\beta-1}{k-\beta-1}$, since $\binom{\alpha}{i}\binom{\beta-\alpha-1}{j-i-1}\binom{n-\beta-1}{k-j-1}=0$ for $i=\alpha, j>\beta$, and for $i>\alpha$ because of the second and the first binomial term respectively, furthermore $\binom{\alpha}{i}\binom{n-\beta-1}{k-i-1}=0$ for $i>\alpha$ which is a contradiction.

Then $p_{i,j}(x,y)$'s are $\frac{k(k-1)}{2}$ linearly independent polynomials, which completes the proof. □

From Lemma 65 and Lemma 66 we get the following necessary condition of k-equivalence of matrices of order n:

Theorem 67 *If A and B are k-equivalent then for every polynomial $r(x,y)$ of total degree at most $k-2$ in two variables*

$$\sum_{x=0}^{n-1}\sum_{y=x+1}^{n-1}\delta_{xy}r(x,y)=0.$$

Proof:

$$\sum_{x=0}^{n-1}\sum_{y=x+1}^{n-1}\delta_{xy}r(x,y)=\sum_{x=0}^{n-1}\sum_{y=x+1}^{n-1}\delta_{xy}\sum_{i=0}^{k-1}\sum_{j=i+1}^{k-1}\lambda_{ij}p_{ij}(x,y)=$$

$$=\sum_{i=0}^{k-1}\sum_{j=i+1}^{k-1}\lambda_{ij}\sum_{x=0}^{n-1}\sum_{y=x+1}^{n-1}\delta_{xy}p_{ij}(x,y)=0.$$

■

5.3.2 The non-symmetric case

Suppose now that for $A\in\Sigma^{k\times k}$ and $B\in\Sigma^{n\times n}$ that A is a *submatrix* of B if we can get A by deleting $n-k$ rows and columns of B arbitrarily. In this case we can get similar statements as above, there are however some little differences, e.g. let $M_k(A)$ be the multiset of all the $\binom{n}{k}^2$ submatrices of A of size $k\times k$, and the diagonal and the lower/upper-triangles are not inherited by the submatrices. This results in similar and simpler proofs, hence we don't discuss them in detail here.

Lemma 68 *Let $A\in\{0,1\}^{n\times n}$ and let $S_{ij}(A)$ be the total number of occurrences of 1's in the i-th row and j-th column of the elements of $M_k(A)$. (Clearly $S_{ij}(A)=\sum_k(A)_{ij}$.) Then*

$$S_{ij}(A)=\sum_{l=0}^{n-1}\sum_{m=0}^{n-1}\binom{l}{i}\binom{n-l-1}{k-i-1}\binom{m}{j}\binom{n-m-1}{k-j-1}a_{lm},\qquad i,j=0,\ldots,k-1.$$

□

Lemma 69

$$\sum_{l=0}^{n-1}\sum_{m=0}^{n-1}\binom{l}{i}\binom{n-l-1}{k-i-1}\binom{m}{j}\binom{n-m-1}{k-j-1}\delta_{lm}=0,\qquad i,j=0,\ldots k-1.$$

□

Consider now the following polynomials of two variables:

$$p_{ij}(x,y) = \binom{x}{i}\binom{n-x-1}{k-i-1}\binom{y}{j}\binom{n-y-1}{k-j-1}$$

for $i, j = 0, \ldots, k-1$. It is easy to see that every p_{ij} is a polynomial of degree $k-1$ in both of the variables and that $p_{ij}(x,y) = 0$ for $0 \le x < i$, $n+i-k < x \le n-1$ and $0 \le y < j$, $n+j-k < y \le n-1$.

Lemma 70 *For every fixed pair of integers n and k, the set $\{p_{ij}(x,y) : i, j = 0, \ldots, k-1\}$ forms a basis of the vectorspace of polynomials of of degree $k-1$ in each variable.* □

Theorem 71 *If A and B are k-equivalent then for every polynomial $q(x,y)$ of of degree $k-1$ in each variable*

$$\sum_{x=0}^{n-1}\sum_{y=0}^{n-1} \delta_{xy} q(x,y) = 0.$$

■

Theorems 67 and 71 show that the case of symmetric and non-symmetric deletions are essentially the same in the sense that the order of magnitude of the bounds resulting from the method will be the same.

Let us mention, that from now on we will index the rows and the columns of the matrices from 1 to n for clearer formulas.

Consider the subsets of the discrete grid of size $n \times n$ as the indices of the 1-entries of the square matrices of order n. Then the main theorems of this chapter suggest the following notion: let $H_1, H_2 \subset \{1,2,\ldots,n\} \times \{1,2,\ldots,n\} = \{1,2,\ldots,n\}^2$ be two arbitrary sets (in the symmetric case we restrict to the upper triangle, more precisely $H_1, H_2 \subset \{(i,j) : 1 \le i < j \le n\}$) and let k be positive integer and define sets H_1 and H_2 *k-distinguishable* if there exists a polynomial $p(x,y)$ with real coefficients such that $\deg p < k$ and

$$\sum_{(x,y)\in H_1} p(x,y) \ne \sum_{(x,y)\in H_2} p(x,y).$$

5.4 Construction of the polynomial

5.4.1 Main results

First we present the main result concerning the existence of k-distinguishable sets without proof. Let us mention that this theorem is the generalization of the result of Borwein et al. [11] for polynomials of two variables.

The following asymptotic result is the best possible in the sense that it approximates the lower bound of Theorem 62 with an $O(\sqrt[3]{\log n})$ factor:

Theorem 72 *If n is sufficiently large and $k \geq 38n^{2/3}$ then any two different sets $H_1, H_2 \subset \{1, 2, \ldots, n\}^2$ are k-distinguishable.*

The proof of this result can be found in Section 5.4.2.

As a consequence of this result and of the theorems of the previous sections we get the following bounds for reconstruction of matrices:

Corollary 73 *For sufficiently large n every $A \in \Sigma^{n \times n}$ is uniquely determined by* $\mathrm{sym}M_k(A)$ *if $k \geq 38n^{2/3} + 1$.*

For sufficiently large n every $A \in \Sigma^{n \times n}$ is uniquely determined by $M_k(A)$ if $k \geq 38n^{2/3} + 1$.

Proof: Suppose that $k \geq 38n^{2/3} + 1$, $A \neq B$ are binary matrices of order n and let $H_1 \neq H_2 \subset \{1, 2, \ldots, n\}^2$ be the set of indices of the 1-entries of A and B, resp. First suppose that the diagonals of the matrices are different. Then we are back to the case of words and we can have a better bound applying Theorem 8, i.e. the matrices are uniquely determined by its k-deck if $k \geq \lfloor \frac{16}{7}\sqrt{n} \rfloor + 5$. If the diagonals of $A \neq B$ are the same then its lower or upper-triangle must be different. Then for sufficiently large n from Theorem 67 and Theorem 72 we get that A and B cannot be k-equivalent, which proves the first part of the corollary.

Using Theorem 71 instead of Theorem 67 in the non-symmetric case implies the second part of the statement. ∎

Theorem 62 shows that one cannot get essentially better bound for reconstruction of matrices using this method.

5.4.2 Proof of the upper bound

Lemma 74 *For arbitrary real numbers $A, M > 0$ there exists a polynomial $f(x)$ with real coefficients with the following properties:*

(a) $f(0) = M$,

(b) $|f(x)| \le \min\left(M, \frac{1}{x^2}\right)$ *for all* $x \in (0, A]$ *and*

(c) $\deg f < \sqrt{\pi}\sqrt{A}\sqrt[4]{M} + 2$.

Proof: Let $k = \left\lceil \frac{\sqrt{\pi}}{2}\sqrt{A}\sqrt[4]{M} \right\rceil + 1$ and consider the Chebishev polynomial $T_k(x)$. Let $u_0 = \cos\frac{\pi}{2k}$ which is the largest root and $u_1 = \cos\frac{\pi}{k}$ which is the largest local minimum (see Figure 5.1).

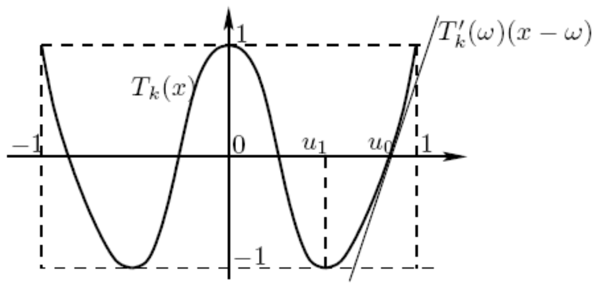


Figure 5.1: Construction of polynomial in Lemma 74

The requested polynomial will be constructed as

$$f(x) = cg^2(x); \qquad g(x) = \frac{-T_k\left(u_0 - \frac{1+u_0}{A}x\right)}{x}, \qquad c = \frac{M}{g^2(0)}.$$

Obviously, $f(0) = M$ and $\deg f = 2(k-1) < \sqrt{\pi}\sqrt{A}\sqrt[4]{M} + 2$, so properties (a) and (c) hold.

To estimate $g(0)$, notice that

$$T_k'(\cos t) = -\frac{\left(T_k(\cos t)\right)'}{\sin t} = -\frac{\left(\cos kt\right)'}{\sin t} = \frac{k \sin kt}{\sin t} \tag{5.1}$$

for all $0 < t < \pi$. Then

$$g(0) = \frac{1+u_0}{A}T_k'(u_0) = \frac{1+\cos\frac{\pi}{2k}}{A} \cdot \frac{k\sin\frac{\pi}{2}}{\sin\frac{\pi}{2k}} > \frac{2 - \frac{1}{2}\left(\frac{\pi}{2k}\right)^2}{A} \cdot \frac{k}{\frac{\pi}{2k}} = \frac{\frac{4}{\pi}k^2 - \frac{\pi}{4}}{A} > \sqrt{M},$$

therefore $c = \frac{M}{g^2(0)} < 1$.

For all $x \in (0, A]$, we have $u_0 - \frac{1+u_0}{A}x \in [-1, 1]$ and $|T_k\big(u_0 - \frac{1+u_0}{A}x\big)| \leq 1$. Hence,

$$|f(x)| = c\left(\frac{T_k\big(u_0 - \frac{1+u_0}{A}x\big)}{x}\right)^2 < \frac{1}{x^2}. \tag{5.2}$$

The function $T_k(x)$ is convex in the interval $[u_1, u_0]$, and thus $|T_k(x)| \leq T_k'(u_0)(x - u_0)$. For $x \in [-1, u_1]$ we have $T_k'(u_0)(x - u_0) < -1$. Therefore $|T_k(x)| \leq |T_k'(u_0)| \cdot (u_0 - x)$ holds in the entire interval $[-1, u_0]$. Then, for all $x \in (0, A]$,

$$|f(x)| = c\left(\frac{T_k\big(u_0 - \frac{1+u_0}{A}x\big)}{x}\right)^2 \leq c\left(\frac{|T_k'(u_0)| \cdot \frac{1+u_0}{A}x}{x}\right)^2 = cg^2(0) = M. \tag{5.3}$$

Estimates (5.2) and (5.3) together provide property (b). □

Remark 75 *This lemma and this polynomial come from an earlier paper [11], but the proof has been re-arranged in a different way to make generalizations easier, as it can be seen in Lemma 76.*

Lemma 76 *For arbitrary real numbers $A, B, M \geq 1$ there exists a polynomial $f(x)$ with real coefficients such that*

(a) $f(0) = M$,

(b) $|f(x)| < \min\left(4M, \frac{1}{x^2}\right)$ for all $x \in [-A, B]$, $x \neq 0$ and

(c) $\deg f < 7\sqrt{ABM} + 2$.

Proof: Without loss of generality we can assume $A \geq B$. Let k be the smallest odd integer which is not less than $\frac{7}{2}\sqrt{ABM}$ and consider the Chebishev polynomial $T_k(x)$. Let $\omega = \text{arc cos}\,\frac{A-B}{A+B}$ and let $u_0 = \cos\omega_0$ be the greatest root of $T_k(x)$ in the interval $[-1, \frac{A-B}{A+B}]$. Since k is odd, $u_0 \geq 0$. Similarly to Lemma 74, the requested polynomial will be constructed as

$$f(x) = cg^2(x); \qquad g(x) = \frac{T_k\left(u_0 + \dfrac{1+u_0}{A}x\right)}{x}, \qquad c = \frac{M}{g^2(0)}.$$

Again, properties (a) and (c) are obvious. For all $x \in [-A, B]$ we have $u_0 + \frac{1+u_0}{A} x \in [-1, 1]$ and therefore $|g(x)| \le \frac{1}{|x|}$.

Since $\omega_0 \le \min\left(\omega + \frac{\pi}{k}, \frac{\pi}{2}\right)$,

$$\sin\omega_0 < \sin\omega + \frac{\pi}{k} \le \sqrt{1 - \left(\frac{A-B}{A+B}\right)^2} + \frac{\pi}{\frac{7}{2}\sqrt{ABM}} = \frac{2\sqrt{AB}}{A+B} + \frac{\frac{2}{7}\pi}{\sqrt{ABM}} < 3\sqrt{\frac{B}{A}},$$

$$|g(0)| = \frac{1+u_0}{A}|T_k'(u_0)| \ge \frac{1}{A} \cdot \frac{k}{\sin\omega_0} > \frac{\frac{7}{2}\sqrt{ABM}}{A \cdot 3\sqrt{\frac{B}{A}}} > \sqrt{M}$$

and

$$|f(x)| = \frac{M}{g^2(0)} g^2(x) < 1 \cdot \left(\frac{1}{x}\right)^2 = \frac{1}{x^2}. \tag{5.4}$$

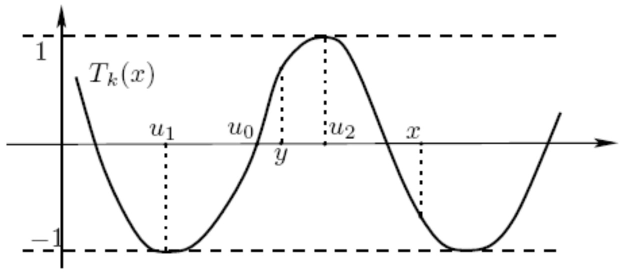


Figure 5.2: Construction of polynomial in Lemma 76

To finish proving property (b) we show that

$$\left|\frac{T_k(x)}{x - u_0}\right| < 2|T_k'(u_0)| \tag{5.5}$$

for all $x \in [-1, 1]$, $x \ne u_0$. Let let $u_1 = \cos\omega_1$ and $u_2 = \cos\omega_2$ be the two neighboring local extrema of $T_k(x)$ around u_0 (see Figure 5.2). Consider an arbitrary point $x \in [-1, 1]$, $x \ne u_0$. If $T_k(x) = 0$ then inequality (5.5) is trivial. Otherwise, choose the point $y = \cos\vartheta \in [u_1, u_2]$ such that x and y lie on the same side of u_0 and $|T_k(y)| = |T_k(x)|$. Then $0 < |y - u_0| \le |x - u_0|$ and by Cauchy's mean value theorem, there exists a $\xi \in (\omega_2, \omega_1)$ such that

$$\left|\frac{T_k(x)}{x-u_0}\right| \le \left|\frac{T_k(y) - T_k(u_0)}{y - u_0}\right| = \left|\frac{\cos k\vartheta - \cos k\omega_0}{\cos\vartheta - \cos\omega_0}\right| = \left|\frac{-k\sin k\xi}{-\sin\xi}\right| < \frac{k}{\sin\omega_2} =$$

$$= \frac{\sin\omega_0}{\sin\omega_2} \cdot |T_k'(u_0)|.$$

Since $\omega \le \omega_0 \le \frac{\pi}{2}$ and $\omega_2 = \omega_0 - \frac{\pi}{2k}$,

$$\frac{\sin\omega_2}{\sin\omega_0} > \frac{\sin\omega_0 - \frac{\pi}{2k}}{\sin\omega_0} = 1 - \frac{\frac{\pi}{2k}}{\sin\omega_0} \ge 1 - \frac{\frac{\pi}{2k}}{\sin\omega} \ge 1 - \frac{\frac{\pi}{7\sqrt{ABM}}}{\frac{2\sqrt{AB}}{A+B}} > \frac{1}{2}$$

and inequality (5.5) follows.

Applying inequality (5.5) on polynomial $g(x)$,

$$|g(x)| = \frac{1+u_0}{A} \cdot \left| \frac{T_k\left(u_0 + \frac{1+u_0}{A}x\right)}{\frac{1+u_0}{A}x} \right| < \frac{1+u_0}{A} \cdot 2|T_k'(u_0)| = 2g(0)$$

and

$$|f(x)| = M\left(\frac{g(x)}{g(0)}\right)^2 < 4M. \tag{5.6}$$

Inequalities (5.4) and (5.6) prove property (b). □

Lemma 77 *For sufficiently large n there exist a convex lattice polygon P_n with the following properties.*

(a) P_n contains a square of size $n \times n$ in its interior, with horizontal and vertical sides;

(b) The side lengths of P_n lie in the interval $[n^{1/3}, 2n^{1/3}]$;

(c) The sides of P_n do not contain any lattice point other than the vertices;

Proof: Denote by $N(R)$ be the number of lattice points (x, y) in the circle $x^2 + y^2 < R^2$ which are visible from the origin (i.e. x and y are relatively prime). It is well-known that

$$\lim_{R\to\infty} \frac{N(R)}{R^2} = \frac{6}{\pi}.$$

Let $R_1 = n^{1/3}$ and $R_2 = 2n^{1/3}$ and consider the lattice vectors (x, y) where x and y are relatively prime integers and $R_1^2 \le x^2 + y^2 < R_2^2$. Choose these vectors to be the sides of P_n; i.e. sort the vectors by direction and merge them to obtain the convex polygon. Obviously, properties (b) and (c) hold.

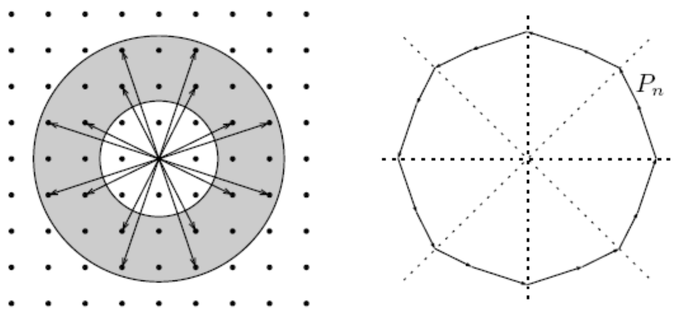


Figure 5.3: Construction of P_n

The perimeter of P_n is at least

$$\big(N(R_2) - N(R_1)\big) \cdot R_1 > \left(\frac{6}{\pi} - \varepsilon\right) R_2^2 R_1 - \left(\frac{6}{\pi} + \varepsilon\right) R_1^3 = \left(\frac{18}{\pi} - 5\varepsilon\right) n > 4\sqrt{2}n.$$

By the symmetry of P_n, property (a) follows. □

Lemma 78 *Let ℓ be an arbitrary line intersecting P_n and let ℓ_1 and ℓ_2 be the two supporting lines of P_n, parallel to ℓ; denote the distance between ℓ and ℓ_i by d_i $(i = 1, 2)$. Assume that ℓ has a common point with a side S of P_n such that the angle between ℓ and S is $\varphi = \text{arc }\sin n^{-1/3}$ (see Figure 5.4). Then*

$$\min(d_1, d_2) < 15n^{1/3}.$$

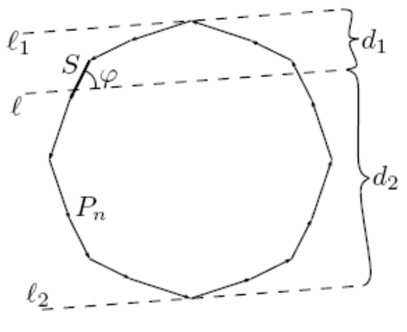


Figure 5.4: Estimate for $\min(d_1, d_2)$

Proof: Without loss of generality, we can assume $d_1 \le d_2$. Consider the side vectors of P_n which lie completely or partially between the lines ℓ and ℓ_1. Translating these vectors to start from the origin, the end-points lie in a region D which is bounded by two concentric circular arcs of radii $R_1 = n^{1/3}$ and $R_2 = 2n^{1/3}$ and two radii of the same circles. The central angle of the arcs is 2φ (see Figure 5.5).

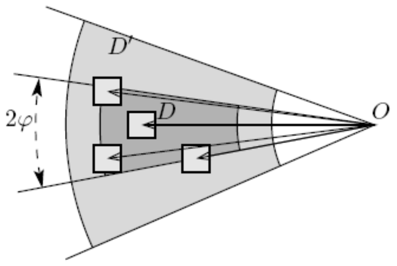


Figure 5.5: Regions D and D'

Drawing a unit square around the endpoints of the vectors, these squares do not overlap and they lie in a region denoted by D' in the Figure. The central angle of this region is less than 4φ and its area is less than $\big((R_2+1)^2-(R_1-1)^2\big)\cdot 4\varphi < 15n^{1/3}$. Therefore, the number of sides of P_n which have at least one end-point between the lines ℓ and ℓ_1 is less than $15n^{1/3}$. Since the side lengths of P_n do not exceed $2n^{1/3}$ and the angles between ℓ and the mentioned sides do no exceed arc $\sin n^{-1/3}$, this implies $d_1 < 15n^{1/3}$. □

Lemma 79 *For sufficiently large n, for an arbitrary nonempty set $H \subset \{1, 2, \ldots$ $\ldots, n\}^2$ there exists a point $\mathbf{a} = (a_1, a_2) \in H$ and a polynomial $p(x, y)$ such that* $\deg p < 38n^{2/3}$ *and*

$$p(a_1, a_2) > \sum_{(x,y)\in H,\quad (x,y)\neq(a_1,a_2)} |p(x, y)|. \tag{5.7}$$

Proof: Translate the polygon P_n, provided by Lemma 77, to polygon P_n' such that the set H is contained in P_n' and at least one point of H lies on the boundary of P_n'. By the choice of the side vectors, any side of P_n' may contain at most two lattice points; if a side contains two lattice points, they must be the two endpoints. Since set H cannot contain all vertices of the polygon P'_n, there is a side S which contains exactly one element of H. Let $\mathbf{a} = (a_1, a_2)$ be this element.

The desired polynomial will be constructed as a product of two polynomials p_1 and p_2. To construct the first polynomial, rotate the side vector of S by 90 degrees such that it points inside P'_n; let this vector be $\mathbf{u} = (u_1, u_2)$; by the construction of P_n, the coordinates u_1 and u_2 are relatively prime integers and $n^{1/3} \le |\mathbf{u}| \le 2n^{1/3}$ (see Fig. 5.6).

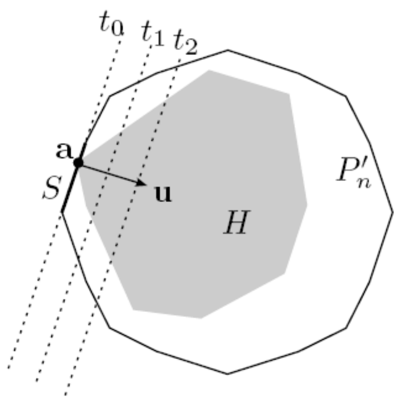


Figure 5.6: Construction of the first polynomial in Lemma 79

Let $f_1(t)$ be the polynomial provided by Lemma 74 for $M = 19$ and $A = 2n^{4/3}$ and define

$$g_1(x, y) = u_1(x - a_1) + u_2(y - a_2), \qquad p_1(x, y) = f_1\big(g_1(x, y)\big).$$

For each integer k, let t_k be the line where $g_1(x, y) = k$. Line t_0 is the extension of side S and the distance between lines t_k and t_{k+1} is $1/|\mathbf{u}|$ for every k. Since the diameter of set H is at most $\sqrt{2}n$ and $|\mathbf{u}| \le 2n^{1/3}$, we have $g_1(H) \subset \big\{0, 1, 2, \dots, \big[2\sqrt{2}n^{4/3}\big]\big\}$.

To construct the second polynomial, take a unit vector $\mathbf{v}$ which encloses an angle $\varphi = \text{arc}\sin n^{-1/3}$ with u. Let ℓ be the line through (a_1, a_2) which is perpendicular to v and let ℓ_1 and ℓ_2 be the two supporting lines of the set H, parallel to ℓ. Let d_i be the distance between ℓ and ℓ_i $(i = 1, 2)$. We can assume

$d_1 \le d_2$. Moreover, by Lemma 78, we have $d_1 < 15n^{1/3}$ and $d_2 \le \sqrt{2}n$ since the diameter of H is at most $\sqrt{2}n$ (Fig. 5.7).

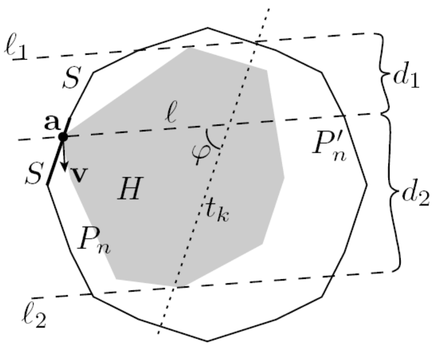


Figure 5.7: Construction of the second polynomial in Lemma 79

Let $f_2(t)$ be the polynomial by Lemma 76 for parameters $A = \max(d_1, 1)$, $B = \max(d_2, 1)$ and $M = 1$ and define

$$g_2(x, y) = v_1(x - a_1) + v_2(y - a_2), \qquad p_2(x, y) = f_2\big(g_2(x, y)\big).$$

For an arbitrary integer k, consider the lattice points on line t_k. The lattice points are distributed uniformly; the distance between the consecutive pairs is $|u|$. Hence, the values $g_2(x, y)$ on these points form an arithmetic progression lying in the interval $[-d_1, d_2]$ with difference $|u|/\sin\varphi \ge 1$.

Since $|f_2(t)| \le \min(4, 1/t^2)$ in the interval $[-d_1, d_2]$, this implies

$$\sum_{(x,y)\in H\cap t_k} |p_2(x, y)| < 2 \sum_{h=0}^{[\max(d_1,d_2)]} \min\left(4, \frac{1}{h^2}\right) < 8 + \frac{\pi^2}{3}.$$

Now let

$$p(x, y) = p_1(x, y) \cdot p_2(x, y).$$

Then

$$p(a_1, a_2) = f_1(0) \cdot f_2(0) = 19 \cdot 1 = 19$$

and

$$\sum_{(x,y)\in H,\quad (x,y)\neq(a_1,a_2)} |p(x,y)| = \sum_{k=1}^{[2\sqrt{2}n^{4/3}]} \sum_{(x,y)\in H\cap t_k} |p_1(x,y)|\cdot|p_2(x,y)| =$$

$$= \sum_{k=1}^{[2\sqrt{2}n^{4/3}]} |f_1(k)| \sum_{(x,y)\in H\cap t_k} |p_2(x,y)| < \sum_{k=1}^{[2\sqrt{2}n^{4/3}]} \frac{1}{k^2}\left(8+\frac{\pi^2}{3}\right) < \frac{\pi^2}{6}\left(8+\frac{\pi^2}{3}\right) <$$

$$< 19 = p(a_1,a_2)$$

so the polynomial $p(x,y)$ satisfies (5.7).

The degree of the polynomial is

$$\deg p = \deg p_1 + \deg p_2 < \left(\sqrt{\pi}\sqrt{2n^{4/3}}\sqrt[4]{19}+2\right)+\left(7\sqrt{15n^{1/3}\cdot\sqrt{2}n}+2\right) < 38n^{2/3}.$$

□

Proof: [Proof for Theorem 72] Let $D_1 = H_1 \setminus H_2$ and $D_2 = H_2 \setminus H_1$. By Lemma 79, there exists a point $(a_1,a_2) \in D_1 \cup D_2$ and a polynomial $p(x,y)$ such that $\deg p < 38n^{2/3} \leq k$ and

$$p(a_1,a_2) > \sum_{(x,y)\in D_1\cup D_2,\quad (x,y)\neq(a_1,a_2)} |p(x,y)|.$$

Without loss of generality we can assume that $(a_1,a_2) \in D_1$. Then

$$\sum_{(x,y)\in H_1} p(x,y) - \sum_{(x,y)\in H_2} p(x,y) = \sum_{(x,y)\in D_1} p(x,y) - \sum_{(x,y)\in D_2} p(x,y) =$$

$$= p(a_1,a_2) + \sum_{(x,y)\in D_1,\quad (x,y)\neq(a_1,a_2)} p(x,y) - \sum_{(x,y)\in D_2} p(x,y) \geq$$

$$\geq p(a_1,a_2) - \sum_{(x,y)\in D_1\cup D_2,\quad (x,y)\neq(a_1,a_2)} |p(x,y)| > 0.$$

Therefore

$$\sum_{(x,y)\in H_1} p(x,y) > \sum_{(x,y)\in H_2} p(x,y)$$

and the sets H_1 and H_2 are k-distinguishable. ■

5.5 Conclusion

Here we collect all of our results concerning the reconstruction from the k-deck consisting of the multiset of its submatrices (Problem 59) and from the sum-matrix (Problem 60) both. Let denote the minimal k's in Problem 59 by symk_{min} or k_{min}; and in Problem 60 by symk^+_{min} or k^+_{min}, resp.

Corollary 80 *There exist positive constants c_1, c_2, c_3 such that:*

$$c_1 \cdot \frac{n^{2/3}}{\sqrt[3]{\log n}} \leq (\mathrm{sym})k^+_{min} \leq c_2 \cdot n^{2/3};$$

$$(\mathrm{sym})k_{min} \leq c_3 \cdot n^{2/3}.$$

Clearly $(\mathrm{sym})k_{min} \leq (\mathrm{sym})k^+_{min}$, yielding the second inequality, however the question $(\mathrm{sym})k_{min} \lneqq (\mathrm{sym})k^+_{min}$ is not a trivial one neither for words. The authors made computer tests for small word and subword-lengths (i.e. $n \leq 20$), but did not find words which are additively k-equivalent but not k-equivalent (the equivalences can be defined analogously to the 2-dimension case).

5.6 Open problems

Problem 81 *Reconstruction from the k-deck consisting of different submatrices.*

Finding the smallest k such that every square matrix can be reconstructed from the k-deck consisting of its different submatrices is very difficult in the symmetric and the non-symmetric case both. (Remember that the reconstruction from the $n-1$-deck in the non-symmetric case is proved in Lemma 17.)

In the symmetric case $k > \lceil \frac{n+1}{2} \rceil$ as the following pair of matrices of size $2k+1$ show (the kth and the $k+1$th element of the last rows is 1):

$$\begin{pmatrix} 0 & 0 & & \dots & & & 0 \\ 0 & 0 & & \dots & & & 0 \\ \vdots & \vdots & & \dots & & & \vdots \\ 0 & 0 & \dots & 1 & 0 & \dots & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & & \dots & & & 0 \\ 0 & 0 & & \dots & & & 0 \\ \vdots & \vdots & & \dots & & & \vdots \\ 0 & 0 & \dots & 0 & 1 & \dots & 0 \end{pmatrix}$$

In the non-symmetric case the situation is slightly better: it is easy to prove that $k \leq \lceil \frac{n}{2} \rceil$ in a special case of matrices with different number of 1 entries using simple graph-theoretical arguments and pigeonhole principle.

Chapter 6

An algorithm for words and its application to genome rearrangement

6.1 Motivation and notation

The differences between the order of genes in two genomes have been used as a measurement of evolutionary distance already more than six decades ago [56]. The rediscovery of inversion distance is dated back to the eighties [50], and since then a large set of papers on optimization methods for genome rearrangement problems has been published. However, except the case of sorting signed permutations by inversions [5; 8; 25; 29; 52; 57] or by translocations [24], only approximations [7; 9; 21; 23; 30] and heuristics [10] exist. Most of the methods concerning with more types of mutations either penalize all the mutations with the same weight [23], or exclude a whole set of possible mutations due to a special choice of weights [21]. (A nice exception can be found in [6].)

Among the numerous parsimony approaches that try to obtain the shortest sequence of rearrangement operations sorting one genome into the other, Bayesian Markov chain Monte Carlo methods have been introduced a few years ago. They define different models where genomes can evolve by reversals [33; 61], reversals and translocations [18] or reversals, transpositions and inverted transpositions

[46; 48]. It has been shown that transpositions and inverted transpositions could happen in unichromosomal genomes [47], therefore it is natural to incorporate such events into the Bayesian model. So far the available computer program for the model accommodating transpositions and inverted transpositions used $O(n^3)$ memory and had $O(n^4)$ running time per MCMC step [48] where n is the length of the genome. Though this memory usage and running time allowed the analysis of short genomes (for example, Metazoan mithochondrial genomes), the program suffered memory problems with large genomes containing hundreds of genes.

We introduce an algorithm for characterizing and sampling transpositions and inverted transpositions. The algorithm characterizes the mutations by the change in the number of cycles in the graph of desire and reality and samples from a distribution in which cycle-increasing mutations are preferred, which is equivalent with searching fixed 3-long signed permutations starting in every positions of a signed permutation of length n. The algorithm runs in $O(n)$ time and has $O(n)$ memory usage. Since linear running time algorithms for characterizing and sampling reversals have already been developed earlier [47; 48], an MCMC step in the reversals, transpositions and inverted transpositions accommodating model takes only $O(n^2)$ running time (the sampling algorithm might be repeated $O(n)$ times in an MCMC step), and needs only linear memory with this algorithm.

Let us remark that one can make other kind of linear algorithm which characterizes the mutations by the number of breakpoints in the graph of desire and reality and samples from a distribution in which breakpoint-removing mutations are preferred. We disregard the full discussion of this algorithm because it is not related to the subject of the thesis, we present however the comparison of the two approaches when we think it is necessary.

6.2 Preliminaries

6.2.1 Mathematical description of genome rearrangement

Let us remember that a genome can be considered as a signed permutation π_i, and the evolutionary distance between two genomes π_1, π_2 is the minimal number of mutations transforming π_1 to π_2. There are different types of mutations acting on genomes, here we present the ones which will take into consideration in our mathematical model, there are however other types (e.g. translocations, etc.) which we eliminate in the following. Let's denote the genes by γ_i-s, the examined mutations are:

- *inversions*: this type of mutations reverses a consecutive block of the genome and changes the reading directions of all genes in the block, i.e.:

$$\dots \gamma_i \overrightarrow{\gamma_{i+1} \dots \gamma_j} \gamma_{j+1} \dots \longrightarrow \dots \gamma_i \overleftarrow{-\gamma_j \cdots - \gamma_{i+1}} \gamma_{j+1} \dots$$

- *transpositions*: this type of mutations interchanges two consecutive blocks of the genome, i.e.:

$$\dots \gamma_i \overrightarrow{\gamma_{i+1} \dots \gamma_j} \overrightarrow{\gamma_{j+1} \dots \gamma_k} \gamma_{k+1} \dots \longrightarrow \dots \gamma_i \overrightarrow{\gamma_{j+1} \dots \gamma_k} \overrightarrow{\gamma_{i+1} \dots \gamma_j} \gamma_{k+1} \dots$$

- *inverted transpositions*: this type of mutations can be considered as a concatenation of a transposition and an inversion on one block of the transposition performed, i.e.:

$$\dots \gamma_i \overrightarrow{\gamma_{i+1} \dots \gamma_j} \overrightarrow{\gamma_{j+1} \dots \gamma_k} \gamma_{k+1} \dots \longrightarrow \dots \gamma_i \overleftarrow{-\gamma_k \cdots - \gamma_{j+1}} \overrightarrow{\gamma_{i+1} \dots \gamma_j} \gamma_{k+1} \dots$$

Note that an inversion can be considered as an inverted transposition where the second block is empty, since we will deal with transpositions and inverted transpositions only.

Since mutations are actions on the group of signed permutations, transforming a genome π_1 to a genome π_2 is equivalent with sorting $\pi_2^{-1}\pi_1$ to the identical permutation, and thus, we are going to talk about sorting permutations instead of transforming one into another.

By following the convention, a signed permutation of length n is represented as an unsigned permutation of length $2n + 2$, $+i$ is replaced by $2i - 1, 2i$, and $-i$ is replaced by $2i, 2i - 1$. This unsigned permutation is then framed to 0 and $2n + 1$. Here we present a short example:

$$(3, -2, -1, 4, -5) \rightarrow (0, 5, 6, 4, 3, 2, 1, 7, 8, 10, 9, 11). \tag{6.1}$$

To properly mimic the signed permutation case, only segments $[2i + 1, 2j]$ are allowed to mutate in the unsigned representation.

6.2.2 The graph of reality and desire

The graph representation of a signed permutation is called *graph of reality and desire*, whose vertices are the numbers from 0 to $2n + 1$, and edges are the reality and desire edges. The *reality edges* connect every second position in the permutation starting with 0. Mutations act on the reality edges, a reversal acts on two reality edges, while a transposition or an inverted transposition on three ones. The *desire edges* are arcs connecting $2i$ with $2i + 1$ for each i. In the present model all edges are unoriented, however the method based on the change of breakpoints used oriented edges as well: a desire edge is unoriented if it spans even number of points otherwise it is oriented (we will need this notion to an open problem in 6.5). Since each vertex has a degree of 2, the graph of desire and reality can be unequivocally decomposed into cycles. A reality edge is a *breakpoint* if its cycle is longer than 2. The graph of reality and desire of the example seen in 6.1 is the following (in the figure the desire edges are the blue ones and the reality edges are the black ones):

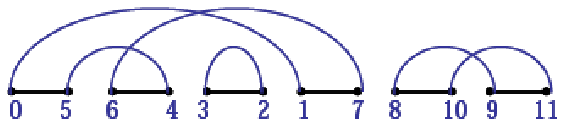


Figure 6.1: The graph of reality and desire of the signed permutation $(3, -2, -1, 4, -5)$

The identity permutation has 0 breakpoints and $n + 1$ cycles (see Figure 6.2 on the next page), all other mutations have more breakpoints and less cycles . Therefore the sorting of a permutation is equivalent with increasing the number of cycles to $n + 1$ or decreasing the number of breakpoints to 0.

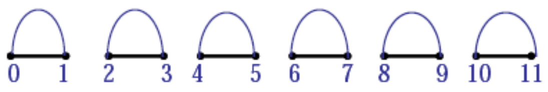


Figure 6.2: The graph of reality and desire of the identity permutation $(1, 2, 3, 4, 5)$

Mutations can be characterized by the number of breakpoints they remove or the change in the number of cycles. We will talk about e.g. -3-b-transpositions meaning that they remove 3 breakpoints or +1-c-inversions, which increase the number of cycles by 1.

6.2.3 Stochastic modeling and Bayesian MCMC

Time-continuous Markov models have been the standard approaches for stochastic modeling of molecular evolution. Unlike the case of nucleic acid substitution models, modeling genome rearrangements is computationally demanding and no analytical solutions is known for transition probabilities. What we can calculate is the likelihood of a trajectory, which is the probability that a given sequence of mutations happened in a time span conditional on a set of parameters describing the model [46; 47; 48].

To sample trajectories from the posterior distribution, we apply Bayesian *Markov chain Monte Carlo* (MCMC) [39; 45] which is a random walk on the possible trajectories, and whose stationary distribution is the posterior distribution of trajectories. The random walk is constructed in two steps. In the first step, a new trajectory is drawn from a proposal distribution, and in the second step, the discrepancy between the proposal and the target distribution is corrected by accepting the proposal with probability

$$\min\left\{1\ ,\ \frac{P(X|Y)\pi(Y)}{P(Y|X)\pi(X)}\right\} \tag{6.2}$$

where P is the proposal distribution, π is the target one, X is the actual state of the chain, and Y is the proposal, and the chain remains in state X with the complement probability [27; 45]. The proposal step replaces a part of the trajectory. The new sub-trajectory is obtained step by step, each mutation is drawn from a distribution that mimics the target distribution we would like to sample from, and the new proposal is independent from the old sub-trajectory.

The mixing of the Markov chain depends on how well the proposal distribution can mimic the target distribution. When proposing a new sub-trajectory step by step, published methods measure the departure of the actual rearrangement from the rearrangement where the sub-trajectory must arrive to, and propose mutations decreasing the measurement of the departure ('good' mutations) with high probability and propose other ones ('bad' mutations) with low probability. This philosophy seems to be essential since random mutations would reach the target rearrangement with a very small probability.

Since there are $3\binom{n+1}{3}$ transpositions and inverted transpositions and $\binom{n+1}{2}$ reversals, an algorithm that spends only constant time with each possible mutation to decide its goodness will already run in $\Omega(n^3)$ time. Therefore it is not a trivial problem how to characterize and sample mutations in less time. Below we show an algorithm characterizing and sampling transpositions and inverted transpositions in linear time, for cycles, but it is easy to construct a simpler one for breakpoints.

6.3 Characterizing and sampling transpositions and inverted transpositions

Figure 6.3 **a)** and **b)** shows the two decision trees that the two algorithms (based on the change of breakpoints and on the change of the cycles resp.) use to sample random mutations. At an internal node, a random decision is made only if both subtrees are non-empty. If one of the subtrees is empty, then the algorithm chooses the other subtree with probability 1. For example, on Figure 6.3 **a)**, if there is no transposition or inverted transposition decreasing the number of breakpoints by 3, and there is no reversal decreasing the number of breakpoints by two, then there is no random decision at the root of the tree, the algorithm will go to the right subtree with probability 1.

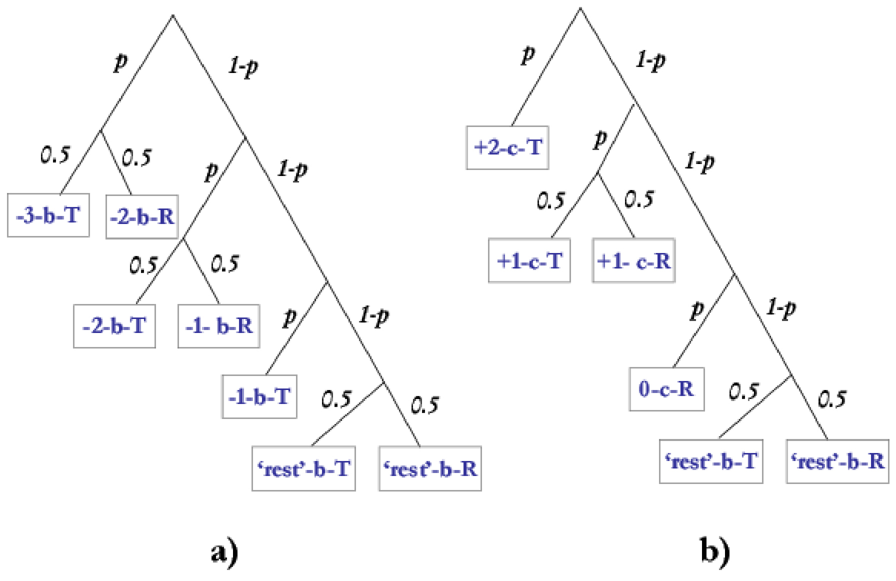


Figure 6.3: Decision trees used by the introduced algorithms

T stands for transpositions and inverted transpositions, R stands for reversals. Numbers on the edges means probabilities, p is between 0.5 and 1. In practice, $p = 0.8$ gives a proposal distribution which is reasonably close to the target distribution, acceptance ratio is about $20 - 30\%$.

6.3.1 Sampler based on the change of cycles

The proposed algorithm characterizes the mutations by the change in the number of cycles. Though this algorithm does not tell the exact number of mutations falling into a given class, it does tell for each category and for each reality edge whether or not there exists a mutation that falls into the given category and its leftmost edge is the given one. This is enough for using the decision tree on Figure 6.3. **b)** and for sampling from a distribution for which the sampling probabilities can be calculated. (We would like to mention for non-experts that the ability of sampling from a distribution does not imply that sampling probabilities can be calculated, see for example [39] and [47].)

It is easy to show that cycle-increasing mutations act on one cycle. If three reality edges are in one cycle, they are in one of the eight possible configurations on Table 6.1. Dotted arcs on the table are not necessarily reality edges but alternating paths of reality end desire edges.

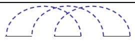
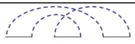
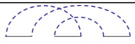
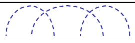
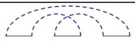
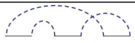
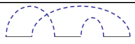
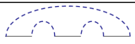
Configuration	transpos.	inv. tr. to the left	inv. tr. to the right
	+2-c	+1-c	+1-c
	+1-c	+2-c	"rest"
	+1-c	"rest"	+2-c
	+1-c	"rest"	"rest"
	"rest"	+1-c	+1-c
	"rest"	+1-c	"rest"
	"rest"	"rest"	+1-c
	"rest"	"rest"	"rest"

Table 6.1: The possible configurations of three reality edges in a cycle and the category of mutations acting on them.

The idea of the algorithm is that for each configuration and reality edge, the algorithm decides whether or not there are other two reality edges to the right being in the given configuration with the third edge. If so, then the reality edge goes to a set from which the algorithm chooses a random leftmost reality edge. Once the algorithm has chosen the mutation type and the leftmost reality edge, it decides for each reality edge on the right hand side of the leftmost edge whether or not it can be together with a rightmost reality edge in a configuration that is good for the given mutation type. After choosing a random middle edge from the ensemble of possible middle edges, the algorithm finally chooses a random good leftmost edge. This method also takes only $O(n)$ time and memory.

6.3.1.1 Preprocessing

The algorithm works on each cycle independently. Starting with the leftmost edge of the cycle, the algorithm traverses the cycle and stores the visiting order of reality edges, as well as the direction of the reality edges on the cycle-traversing. $\pi(i)$ tells the visit order of the reality edge in the ith position, and $pos(i)$ tells the position of the edge which was the ith in the cycle tour and $sign(i)$ tells the direction of the edge.(We will denote by plus sign the left to right direction and by minus sign the right to left direction.) These arrays can be trivially calculated in $O(n)$ time.

The pseudocode of this stage can be found in Algorithm 1.

Algorithm 1 `Preprocessing1` {*calculating the arrays* $\pi(), sign(), pos()$}

```
for $i = 1$ to $n$ do
    $\pi(i) \leftarrow j_i$ if the edge $i$ is the $j_i$-th in the cycle tour
    if the edge $i$ has left to right direction in the cycle tour then
        $sign(i) \leftarrow +$
    else
        $sign(i) \leftarrow -$
    $i \leftarrow i + 1$
for $j = 1$ to $n$ do
    $pos(j) \leftarrow \pi^{-1}(j)$
    $j \leftarrow j + 1$
```

After this, the algorithm traverses the reality edges in reverse position order (namely, from right to left), and calculates $s\max(i) = \max_{j \geq i}\{\pi(j)|sign(j) = s\}$ and $s\min(i) = \min_{j \geq i}\{\pi(j)|sign(j) = s\}$ both for positive and negative signs.

The pseudocode of this stage can be found in Algorithm 2.

Algorithm 2 `Preprocessing2` {*calculating the arrays* $s\max(), s\min()$}

```
for $s \in \{+,-\}$ do
  $s\max(n) \leftarrow null$, $s\min(n) \leftarrow null$
  for $i = n$ to 1 do
    if $sign(i) = s$ then
      if $s\max(i) = null \wedge s\min(i) = null$ then
        $s\max(i) = s\min(i) \leftarrow \pi(i)$
      else
        if $\pi(i) \in [s\min(i), s\max(i)]$ then
          $s\max(i) \leftarrow s\max(i+1)$, $s\min(i) \leftarrow s\min(i+1)$
        else
          if $\pi(i) < s\min(i)$ then
            $s\min(i) \leftarrow \pi(i)$
          else
            $s\max(i) \leftarrow \pi(i)$
    $i \leftarrow i - 1$
```

6.3.1.2 Existence of mutations

Each configuration on Table 6.1 can be traversed in six possible ways, see for example on Figure 6.4 how the first configuration on Table 6.1 can be traversed. Eight configuration times the six possible traversing gives 48 cases, and this is the 3! possible permutations of the visiting order of the three edges multiplied by the 2^3 possible signs of the three edges. Instead of configurations and traversing, we will talk about visiting permutations and signs, there is a one-to-one correspondence between them. Therefore the problem is to tell in constant time for each permutation, sign pattern and reality edge whether or not there are other two reality edges to the right being in the given permutation and sign pattern. Any sign pattern can be discussed in a general way, the three signs will be denoted by s_1, s_2 and s_3 from left to right.

Another observation is that it is enough to give algorithms for the $1,2,3$, the $2,1,3$ and $1,3,2$ permutations since the cycle can be traversed with starting the tour on the leftmost edge in the other direction. This will cause a change in the

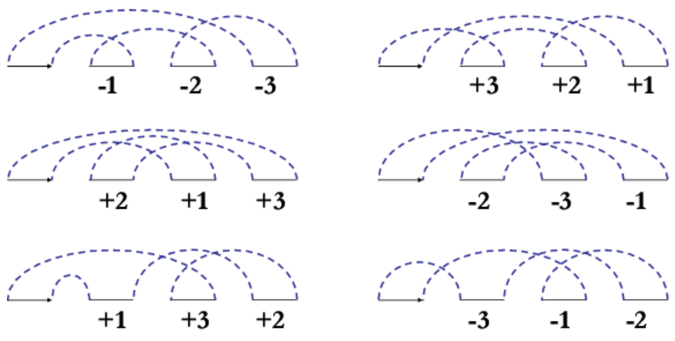


Figure 6.4: The possible visiting order of three reality edges on which a transposition increases the number of cycles by two

permutation such that 3 and 1 will be swapped, and all signs will change to the other sign. For example, on Figure 6.4 the cases on the right column will turn to the cases on the left column if the cycle is traversed in a reverse order. Dotted arcs are not necessarily reality edges but alternating paths of reality and desire edges.

6.3.1.3 The $1, 2, 3$ case

The $1, 2, 3$ permutation is the easy case for any signs. The algorithm traverses again the reality edges in a reverse position order, and calculates

$$s_2 \max s_3 \max(i) = \max_{j \geq i}\{\pi(j) | \pi(j) \leq s_3 \max(j) \ \& \ sign(j) = s_2\} \qquad (6.3)$$

There is a $1, 2, 3$ permutation with a good sign pattern for a position i if $sign(i) = s_1$ and $\pi(i) \leq s_2 \max s_3 \max(i)$.

The pseudocode of the $1, 2, 3$ case can be found in Algorithm 3.

Algorithm 3 The $1,2,3$ case

```
execute Preprocessing1, Preprocessing2
$s_2 \max s_3 \max(n) \leftarrow null$
for $i = n$ to 1 do
  if $sign(i) = s_2$ then
    if $\pi(i) \leq s_3 \max(i)$ then
      if $s_2 \max s_3 \max(i) = null$ then
        $s_2 \max s_3 \max(i) \leftarrow \pi(i)$
      else
        if $\pi(i) > s_2 \max s_3 \max(i)$ then
          $s_2 \max s_3 \max(i) \leftarrow \pi(i)$
        else
          $s_2 \max s_3 \max(i) \leftarrow s_2 \max s_3 \max(i+1)$
  $i \leftarrow i - 1$
for $i = 1$ to $n$ do
  if $sign(i) = s_1 \wedge \pi(i) < s_2 \max s_3 \max(i)$ then
    return There is a requested signed permutation starting in $pos(i)$.
  else
    return There is no requested signed permutation starting in $pos(i)$.
  $i \leftarrow i + 1$
```

6.3.1.4 The $2,1,3$ case

The algorithm runs an index i from 1 to n and is in the rightmost position j for which $\pi(j) < i$, $sign(j) = s_2$ and $s_3 \max(j) > i$. If $pos(i) < j$ and $sign(i) = s_1$, then there is a $2,1,3$ case with proper signs starting in position $pos(i)$, otherwise such configuration does not exist in that position. Knowing the $pos()$ and $s_3 \max()$ arrays, it is easy to jump to the proper rightmost position until $i > s_3 \max(j)$. Then the algorithm must go back to the proper position j for which $\pi(j) < i < s_3 \max(j)$. Directly traversing back the positions would take $O(n)$ time and such traversing back might be necessary $O(n)$ times, giving the algorithm an $O(n^2)$ running time. Therefore some preprocessing is necessary.

In the preprocessing, the algorithm marks the anchor points of the $s_3 \max$

threshold function (rectangles on Figure 6.5). Then for each interval between two consecutive anchor points, it traverses backwards the interval, and creates the chained list of the local s_2 min anchor points (black circles on Figure 6.5, locality also means that it checks only points which are smaller than the right anchor s_3 max value). For the local minimum, it finds on the previous chained list the first anchor point which is smaller than the actual local minimum, traversing the chain from up to down. The actual list is then augmented with the rest of the list, however, with up-to-down search, each anchor point is visited only once while searching, providing the $O(n)$ running time of the preprocessing algorithm.

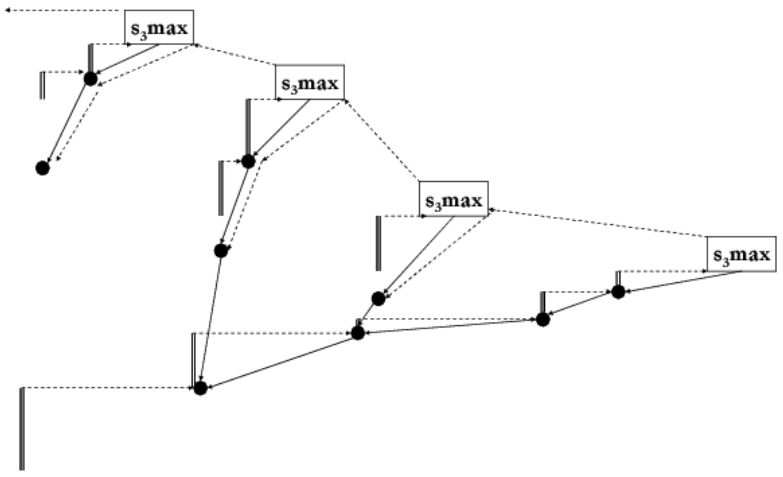


Figure 6.5: Explanatory figure for the $2, 1, 3$ algorithm

Increasement of i is indicated with a double line on Figure 6.5, jumping in positions is indicated with a dashed line. While there is no j for which $\pi(j) < i < s_3 \max(j)$ and $sign(j) = s_2$, the algorithm remains in position 1 and marks all $pos(i)$ having no good $2, 1, 3$ configuration. The algorithm jumps positions toward the right end of the permutation whenever a good position j appears, until $i > s_3 \max(j)$. Then it jumps to the next s_3 max anchor point to the left, and slides down on the s_2 min chained list until for the current position j, $\pi(j) < i$. Each edge of the s_2 min anchor chains is used at most once for back-traversing. To see this suppose, that we are in the position j of an s_2 min anchor chain, such that $\pi(j) < i < s_3 \max(j)$. Increasement of i ensures the validity of condition $\pi(j) < i$, but it can indicate that $i > s_3\max(j)$. However in this case the algorithm jumps the next $s_3\max$ anchor point to the left, and in the following it uses only the edges of the $s_2\min$ anchor chain starting in this $s_3\max$ anchor point, which are unused edges all. Since the total size of the chained $s_2\min$ anchor list is $O(n)$, the algorithm spends only $O(n)$ time with back-traversing, and hence, has only $O(n)$ running time altogether

The pseudocode of the $2, 1, 3$ case can be found in Algorithm 4.

Algorithm 4 `The` $2, 1, 3$ `case`

```
execute Preprocessing1, Preprocessing2
create ChainedList($s_3\max$Anchor) and ChainedList(Local$s_2\min$Anchor)
$j \leftarrow 1$
for $i = 1$ to $n$ do
  if $\pi(j) < i < s_3\max(j)$ then
    if $pos(i) < j \wedge sign(i) = s_1$ then
      return There is a requested signed permutation starting in $pos(i)$.
    else
      return There is no requested signed permutation starting in $pos(i)$.
  else
    increase $j$ in ChainedList($s_3\max$Anchor) until $i > s_3\max(j)$
    jump to Prev$s_3\max$Anchor
    decrease $j$ in ChainedList(Local$s_2\min$Anchor) until $\pi(j) < i$
    if $pos(i) < j \wedge sign(i) = s_1$ then
      return There is a requested signed permutation starting in $pos(i)$.
    else
      return There is no requested signed permutation starting in $pos(i)$.
  $i \leftarrow i + 1$
```

6.3.1.5 The $1, 3, 2$ case

For this case, the preprocessing creates a double chained list of the numbers form 1 to n, and if $s_2 \neq s_3$, then the the signs of the numbers in the permutation are also denoted on the chained structure, and there are pointers to the next and previous numbers both with the same and with the other signs. Then the preprocessing traverses the positions of the permutation from left to right, and pulls out the visited numbers from the chained list. Before pulling out a number i having sign s_2, it checks which number is the next number in the chained list with sign s_3. This is the biggest number on the permutation with sign s_3 which is smaller than i and whose position is bigger than $pos(i)$. These numbers are stored in an additional array $maxmin(i)$. After the preprocessing, the algorithm traverses the permutation in reverse position order, and notes the maximum of the $maxmin$ values, denoted by M. There is a proper configuration for position i if $sign(i) = s_1$ and $\pi(i) < M$.

The pseudocode of the $1, 3, 2$ case can be found in Algorithm 5.

Algorithm 5 The $1, 3, 2$ case

```
execute Preprocessing1, Preprocessing2
$maxmin(n) \leftarrow -1$, $maxmin(n-1) \leftarrow -1$, $M \leftarrow -1$
create DoubleChainedList($\pi()$) with SignPointers($s_3$Prev, $s_3$Next)
for $i = 1$ to $n - 2$ do
  if $sign(i) = s_2$ then
    read $s_3$Next from DoubleChainedList($\pi()$)
    $maxmin(i) \leftarrow s_3$Next
  else
    $maxmin(i) \leftarrow -1$
  pull out $\pi(i)$ from DoubleChainedList($\pi()$)
  $i \leftarrow i + 1$
for $i = n - 2$ to $1$ do
  while $maxmin(i) = -1$ do
    $i \leftarrow i - 1$
  if $maxmin(i) > M$ then
    $M \leftarrow maxmin(i)$
  if $sign(i) = s_1 \wedge \pi(i) < M$ then
    return There is a requested signed permutation starting in $pos(i)$.
  else
    return There is no requested signed permutation starting in $pos(i)$.
  $i \leftarrow i - 1$
```

6.3.1.6 Mutations with leftmost reality edge of position 1, and sampling the middle and rightmost edges

The abovementioned algorithms work for the reality edge in position 1, with the notation that the given permutation patterns must be compared with the configurations in Table 6.1.

Once we choose a rightmost edge in position i and the type of the mutation, deciding whether or not a reality edge can be in a pattern being good for the prescribed mutation is very easy, one should only check the s_3 min and s_3 max values with the possible restriction they might not be bigger or smaller than $\pi(i)$, depending on the searched permutation pattern. Similarly, once the rightmost and middle edge have been chosen, it is very easy to find the list of possible leftmost reality edges.

6.3.1.7 Weighting the reality edges

Sampling from the uniform list of possible rightmost edges might lead to a very skewed distribution where mutations on the right ends of cycles are preferred. This is because at the left end of a cycle, there might be significantly more mutations of a category with a leftmost reality edge than at the right end of a cycle. Therefore some sophisticated weighting yields better distribution also in terms of acceptance ratios. However one can get algorithms of running-time $O(n \log n)$ arising this purpose using some divide-and-conquer arguments.

6.3.1.8 “Rest” mutations

We must mention that mutations acting on more than one cycle all fall into the “rest” category. Knowing whether or not there are reality edges being in other cycles, it is trivial to decide whether or not mutations acting on different cycles and having the current reality edges as leftmost edge exists is a trivial problem.

6.4 Discussion

We introduced a new strategy for efficient sampling of transpositions and inverted transpositions. The algorithm runs in $O(n)$ time and memory, and can be used in Bayesian MCMC. With this sampling algorithm, one MCMC step can be performed in $O(n^2)$ time and in linear memory, which is a significant improvement to the so far available algorithm having $O(n^4)$ running time and $O(n^3)$ memory.

We hope that we could convince the readers that designing Markov chain Monte Carlo methods in bioinformatics is not only a statistical problem but an at least as important algorithmic problem, too.

6.5 Open problems

Problem 82 *Other approaches of genome rearrangement.*

As a finishing of the thesis we present a problem of theory of graph-algorithms which is equivalent with a statistical problem of genome rearrangement.

Suppose that there is a connected graph with two kinds of vertices: the black ones and the white ones, furthermore there is an operation on the black vertices. The operation changes an arbitrary black vertex to white and alters both the colors and the adjacencies of all of its neighbors. Bergeron [8] showed that every graph with at least on black vertex can be transformed into isolated white vertices only via a series of operations above and determined the minimal number of such operations.

The problem is to determine the number of the different operation-series being of minimal.

Surprisingly, this question is motivated by genome rearrangement. To see this consider *the breakpoint graph* G of the permutation π with oriented desire edges and presence of transpositions only: every vertex of G corresponds to a desire edge, which is black if the edge is oriented and white otherwise. Two vertices are connected in G if the corresponding desire edges intersect. It is easy to see, that an operation toward isolated white vertices only in G can never increase the number of breakpoints in the graph of desire and reality, hence it is equivalent to sorting the permutation π.

As we mentioned in 6.3.1.7, the number of different sortings of a permutation is a necessary quantity in the statistical examination of genome rearrangement which equals to the number of different minimal operation-series in a graph G.

The results of the thesis are based on the following papers:

[L1] P.L. Erdős, P. Ligeti, P. Sziklai, and D.C. Torney. Subwords in reverse-complement order. *Annals of Combinatorics*, **10**:415–430, 2006.

[L2] G. Kós, P. Ligeti, and P. Sziklai. Reconstructing matrices from submatrices. Manuscript.

[L3] P. Ligeti. Matrix-posets and automorphisms. Manuscript.

[L4] P. Ligeti and P. Sziklai. Automorphisms of subword-posets. *Discrete Math.*, **305**(1-3):372–378, 2003.

[L5] P. Ligeti and P. Sziklai. Generalized bounds for reconstruction of words. In *5th Hungarian-Japanese Symposium on Discrete Mathematics and Its Applications*, pages 252–256, 2007.

[L6] I. Miklós, T.P. Brooks, and P. Ligeti. Efficient sampling of transpositions and inverted transpositions for Bayesian MCMC. In *Proceedings of WABI2006*, **4175** of *Lecture Notes in Computer Science*, pages 174–185, 2006.

Further papers of the author not related to the thesis:

[N1] M. Bárász, P. Ligeti, L. Mérai, and D.A. Nagy. Boardroom voting using devices with limited computational resources. Submitted to Financial Cryptography and Data Security '08.

[N2] M. Bárász, P. Ligeti, L. Mérai, D.A. Nagy, and P. Sziklai. *Hungarian Patent Submission* **Nr.P0700548**: Szavazatszámláló rendszer, 2007.

[N3] M. Bárász, P. Ligeti, K. Lója, L. Mérai, and D.A. Nagy. Anonymous sealed bid auction protocol. Submitted to EUROCRYPT 2008.

[N4] M. Bárász, P. Ligeti, K. Lója, L. Mérai, D.A. Nagy, and P. Sziklai. *Hungarian Patent Submission* **Nr. P0700596**: Árverező rendszer, 2007.

[N5] Sz. L. Fancsali and P. Ligeti. Some applications of finite geometry for secure network coding. Submitted to Journal of Math. Cryptology.

Bibliography

[1] P.G. Aleksanjan. The reconstruction of vectors by their fragments. *Akad. Nauk. Armyan. SSR Inst. Mat.*, **68**:39–41, 1979. 11

[2] J. Allouche and J. Shallit. The ubiquitous Prouhet-Thue-Morse sequence. In *Sequences and their applications, Proceedings of SETA'98*, **2**, pages 1–16. Springer, 199. 12

[3] N. Alon, Y. Caro, I. Krasikov, and Y. Roditty. Combinatorial reconstruction problems. *J. Comb. Theory Ser. B*, **47**(2):153–161, 1989. 10

[4] L. Babai. Automorphism groups, isomorphism, reconstruction. In *Handbook of combinatorics (vol. 2)*, pages 1447–1540. MIT Press, Cambridge, MA, USA, 1995. 9

[5] D.A. Bader, B.M.E. Moret, and M. Yan. A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comp. Biol.*, **8**(5):483–491, 2001. 78

[6] M. Bader and E. Ohlebusch. Sorting by weighted reversals, transpositions and inverted transpositions. In *Proceedings of RECOMB2006*, **3909** of *Lecture Notes in Bioinformatics*, pages 563–577, 2006. 78

[7] V. Bafna and A. Pevzner. Sorting by transpositions. *SIAM J. Disc. Math.*, **11**(2):224–240, 1998. 78

[8] A. Bergeron. A very elementary presentation of the Hannenhalli-Pevzner theory. In *Proceedings of CPM2001*, pages 106–117, 2001. 78, 94

[9] P. Berman, S. Hannenhalli, and M. Karpinski. 1.375-approximation algorithm for sorting by reversals. In *Proceedings of ESA2002*, pages 200–210, 2002. 78

[10] M. Blanchette, T. Kunisawa, and D. Sankoff. Parametric genome rearrangement. *Gene*, **172**:11–17, 1996. 78

[11] P. Borwein, T. Erdélyi, and G. Kós. Littlewood-type problems on [0,1]. *Proceedings of the London Mathematical Society*, **79**(1):22–46, 1999. 13, 67, 69

[12] G. Burosch, H-D. O.F. Gronau, and J-M. Laborde. The automorphism group of the subsequence poset $B_{m,n}$. *Order*, **12**:179–194, 2000. 6, 27

[13] A. Carpi and A. deLuca. Words and special factors. *Theoret. Comp. Sci.*, **259**:145–182, 2001. 31

[14] P. Chase. Subsequence numbers and logarithmic concavity. *Discrete Math.*, **16**:123–140, 1976. 27

[15] C. Choffrut and J. Karhumäki. Combinatorics of words. In G. Rozenberg and A. Salomaa, editors, *Handbook on Formal Languages*, **I**. Springer, 1997. 12

[16] A. Dress and P.L. Erdős. Reconstructing words from subwords in linear time. *Ann. Comb.*, **8**(4):457–462, 2004. 14, 33

[17] M. Dudík and L.J. Schulman. Reconstruction from subsequences. *J. Comb. Theory Ser. A*, **103**(2):337–348, 2003. 12

[18] R. Durrett, R. Nielsen, and T.L. York. Bayesian estimation of genomic distance. *Genetics*, **166**:621–629, 2004. 78

[19] A.G. D'yachkov, P.L. Erdős, A.J. Macula, V.V. Rykov, D.C. Torney, C.-S. Tung, P.A. Vilenkin, and P. S. White. Exordium for DNA codes. *J. Comb. Opt.*, **7**:369–379, 2003. 4, 37

[20] P.L. Erdős, P. Sziklai, and D. Torney. The word poset and insertion-deletion codes. *Electronic Journal of Combinatorics*, **8**:10 pp., 2001. 27

[21] N. Eriksen. (1+ε)-approximation of sorting by reversals and transpositions. In *Proceedings of WABI2001*, **2149** of *Lecture Notes in Computer Science*, pages 227–237, 2001. 78

[22] W.H. Foster and I. Krasikov. An improvement of a Borwein-Erdélyi-Kós result. *Methods Appl. Anal.*, **7**(4):605–614, 2000. 13

[23] Q-P. Gu, S. Peng, and H.I. Sudborough. A 2-approximation algorithm for genome rearrangements by reversals and transpositions. *Theor. Comp. Sci.*, **210**(2):327–339, 1999. 78

[24] S. Hannenhalli. Polynomial algorithm for computing translocation distance between genomes. In *Proceedings of CPM1996*, pages 168–185, 1996. 78

[25] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of ACM*, **46**(1):1–27, 1999. 78

[26] F. Harary. On the reconstruction of a graph from a collection of subgraphs. In *Theory of Graphs and its Applications (Proc. Sympos. Smolenice, 1963)*, pages 47–52. Publ. House Czechoslovak Acad. Sci., Prague, 1964. 10

[27] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, **57**(1):97–109, 1970. 82

[28] L.O. Kalashnik. The reconstruction of a word from fragments. *Numerical Mathematics and Computer Technology, Akad. Nauk. Ukrain. SSR Inst. Mat.*, **Preprint IV**:56–57, 1973. 3, 11

[29] H. Kaplan, R. Shamir, and R. Tarjan. A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.*, **29**(3):880–892, 1999. 78

[30] J.D. Kececioglu and D. Sankoff. Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. *Algorithmica*, **13**(1/2):180–210, 1995. 78

[31] P.J. Kelly. *On Isometric Transformations.* PhD thesis, University of Wisconsin, 1942. 5, 8, 61

[32] I. Krasikov and Y. Roditty. On a reconstruction problem for sequences. *J. Combin. Theory Ser. A*, **77**(2):344–348, 1997. 12, 62, 63

[33] B. Larget, D.L. Simon, and B.J. Kadane. Bayesian phylogenetic inference from animal mitochondrial genome arrangements. *J. Roy. Stat. Soc. B.*, **64**(4):681–695, 2002. 78

[34] V.K. Leont'ev and Y.G. Smetanin. Problems of information on the set of words. *Journal of Mathematical Sciences*, **108**(1):49–70, 2002. 13, 14, 33

[35] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *J. Soviet Phys.-Doklady*, **10**:707–710, 1966. 4

[36] V. I. Levenshtein. Efficient reconstruction of sequences from their subsequences or supersequences. *J. Combin. Theory Ser. A*, **93**(2):310–332, 2001. 14

[37] V.I. Levenshtein. On perfect codes in deletion and insertion metric. *Discrete Math. Appl.*, **2**:241–258, 1992. 14

[38] V.I. Levenshtein. Efficient reconstruction of sequences. *IEEE Tr. Inf. Theory*, **47**(1):2–22, 2001. 14

[39] J. S. Liu. *Monte Carlo Strategies in Scientific Computing.* Springer, 2001. 82, 84

[40] M. Lothaire. *Combinatorics on Words*, **17** of *Encylopedia of Mathematics and its Applications.* Addison-Wesley, Reading, 1985. 1, 6, 13, 32, 40, 104, 105

[41] M. Lothaire. *Applied Combinatorics on Words*, **105** of *Encylopedia of Mathematics and its Applications*. Cambridge University Press, 2005. 1, 5

[42] B. Manvel, A. Meyerowitz, A. Schwenk, and K. Smith. P. Stockmeyer. Reconstruction of sequences. *Discrete Math.*, **94**(3):209–219, 1994. 11

[43] B. Manvel and P.K. Stockmeyer. On reconstruction of matrices. *Mathematics Magazine*, **44**(4):218–221, 1971. 5, 61

[44] A. Marcus and G. Tardos. Excluded permutation matrices and the Stanley–Wilf conjecture. *J. Combin. Theory Ser. A*, **107**(1):153–160, 2004. 10

[45] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *J. Chem. Phys.*, **21**(6):1087–1091, 1953. 82

[46] I. Miklós. MCMC genome rearrangement. *Bioinformatics*, **19**:130–137, 2003. 79, 82

[47] I. Miklós and J. Hein. Genome rearrangement in mitochondria and its computational biology. In *Proceedings of the 2nd RECOMB Satellite Workshop on Computational Genomics*, **3388** of *Lecture Notes in Computer Science*, pages 85–96, 2004. 79, 82, 84

[48] I. Miklós, P. Ittzés, and J. Hein. ParIS genome rearrangement server. *Bioinformatics*, **21**(6):817–820, 2005. 79, 82

[49] J. Pach and G. Tardos. Forbidden patterns and unit distances. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 1–9, New York, NY, USA, 2005. ACM Press. 10

[50] J.D. Palmer and L.A. Herbon. Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *J. Mol. Evol.*, **28**:87–97, 1988. 78

[51] A.D. Scott. Reconstructing sequences. *Discrete Math.*, **175**(1):231–238, 1997. 12

[52] A. Siepel. An algorithm to find all sorting reversals. In *Proceedings of RECOMB2002*, pages 281–290, 2002. 78

[53] I. Simon. Piecewise testable events. In *Proceedings of the 2nd GI Conference on Automata Theory and Formal Languages*, **33** of *Lecture Notes in Computer Science*, pages 214–222, 1975. 6, 13, 32, 104, 105

[54] I. Simon. Words distinguished by their subwords. In *Proceedings of WORDS'03, the 4th International Conference on Combinatorics on Words*, pages 6–13, 2003. 14, 33

[55] J. Spencer, E. Szemerédi, and W.T. Trotter. Unit distances in the Euclidean plane. In *Graph Theory and Combinatorics (B. Bollobás ed.)*, pages 293–303. Academic Press, New York, 1984. 10

[56] A.H. Sturtevant and E. Novitski. The homologies of chromosome elements in the genus drosophila. *Genetics*, **26**:517–541, 1941. 78

[57] E. Tannier and M.-F. Sagot. Sorting by reversals in subquadratic time. In *Proccedings of the 15th CPM*, **3109** of *Lecture Notes in Computer Science*, pages 1–13, 2004. 78

[58] G. Tardos. Extremal problems for finite point configurations and 0-1 matrices. Doctoral Thesis, Alfréd Rényi Institute of Mathematics, Hungarian Academy of Sciences, 2005. 10, 61

[59] G. Tardos. On 0-1 matrices and small excluded submatrices. *J. Comb. Theory Ser. A*, **111**(2):266–288, 2005. 10

[60] S. M. Ulam. *A collection of mathematical problems.* Interscience Tracts in Pure and Applied Mathematics. Interscience, New York, 1960. 5, 8, 61

[61] T.L. York, R. Durrett, and R. Nielsen. Bayesian estimation of inversions in the history of two chromosomes. *J. Comp. Biol.*, **9**:808–818, 2002. 78

[62] A.I. Zenkin and V.K. Leont'ev. On a nonclassical recognition problem. *USSR Comput. Maths Math. Phys.*, **24**(3):189–193, 1984. 11

Summary

Combinatorics on words is a relatively new research area of discrete mathematics partly inspired by problems in theoretical computer science and other fields of mathematics, such as number theory, group theory and probability theory.

In mathematics, there is a notable number of problems that deal with reconstruction either of an object by some incomplete information about it or of a whole by its parts. In a substantial part of this thesis we will examine different generalizations and applications of the following reconstruction problem:

Basic problem *Let the length n of a word and an alphabet Σ be given. Determine the smallest k such that every word $w \in \Sigma^n$ can be reconstructed from the k-deck of its subwords.*

In Chapter 2 we examine the most simple version of reconstruction from different subwords in the case of DNA-words and matrices (i.e. the reconstruction from the $n-1$-deck) and as an application of these kind of results we determine the automorphism groups of posets consisting of DNA-words and matrices partially ordered by the substrand and the submatrix relation, resp.

In Chapter 3 we give an improvement of the result of Simon [53] and Lothaire [40] (i.e. the reconstruction from different subwords of length at most $\lceil \frac{n+1}{2} \rceil$) for a general alphabet and with lower and upper bounds for the number of occurrences of letters in the words.

In Chapter 4 we examine the reconstruction of DNA-words from set of its different substrands. We prove that every DNA-word of length n can be reconstructed from its different substrands of length at most $\lfloor \frac{2(n+1)}{3} \rfloor$.

In Chapter 5 we consider the reconstruction of square matrices from the multiset of its square submatrices and from its sum-matrix. The main results of this chapter are an asymptotical upper bound of order of magnitude $O(n^{2/3})$ and a lower bound which differs in a factor $O(\sqrt[3]{\log n})$ only from the upper bound.

In Chapter 6 we present an algorithm on words which has an application to bioinformatics which determines the evolutionary distance between two organisms. The algorithm runs in $O(n)$ time and uses $O(n)$ memory, where n is the size of the genome. This is a significant improvement compared with the so far available brute force method with $O(n^3)$ running time and memory usage.

Összefoglaló

A szavak kombinatorikája egy viszonylag fiatal kutatási ága a diszkrét matematikának, melyet részben számelméleti, csoportelméleti, valószínűségszámítási, valamint elméleti számítástudományi problémák ihlettek.

Matematikai problémák sokasága foglalkozik egy objektum rekonstruálásával annak részeiből, vagy a rá vonatkozó információ-töredékekből. Az értekezés legnagyobb részében az alábbi rekonstrukciós feladat általánosításaival és alkalmazásaival foglalkozunk:

Alapprobléma *Legyen adva egy Σ abécé és egy n szóhossz. Határozzuk meg a legkisebb k értéket, amire tetszőleges Σ feletti n hosszú szót rekonstruálni lehet a k-hosszú részszavaiból!*

Az 2. fejezetben a különböző részszavakból rekonstruálásnak a lehető legegyszerűbb esetének - rekonstruálás az $n-1$-hosszú részszavakból - általánosításait vizsgáljuk DNS-szavak es mátrixok esetén. Ezen állítások alkalmazásaiként különböző részbenrendezett halmazok (röviden poset-ek) automorfizmus csoportjait határozzuk meg.

A 3. fejezetben a különböző részszavakból történő rekonstruálásnál a Simon [53] és Lothaire [40] által adott $\lceil \frac{n+1}{2} \rceil$-es korlátot javítjuk meg tetszőleges méretű ábécé, valamint a szavakat alkotó betűk számára előírt (alsó és felső) korlátok esetén.

A 4. fejezetben DNS-szavaknak különböző részszálaiból történő rekonstruálását vizsgáljuk. Megmutatjuk, hogy minden n hosszú DNS-szó rekonstruálható a legfeljebb $\lfloor \frac{2(n+1)}{3} \rfloor$-hosszú részszálaiból.

Az 5. fejezetben egy négyzetes mátrixot a sorainak és oszlopainak tetszőleges-, vagy szimmetrikus törlésével keletkező részmátrixaiból, illetve ezek összegéből kívánunk rekonstruálni. A fejezet fő eredményei egy aszimptotikusan $O(n^{2/3})$-os felső korlát, valamint az összeg-mátrixból való rekonstrukcióra adott alsó korlát, mely ettől mindössze egy $O(\sqrt[3]{\log n})$-es szorzóban tér el.

A 6. fejezetben célunk két faj közötti evolúciós távolság meghatározása egy szó-algoritmus segítségével. Fő eredményünk egy, a genomok hosszában lineáris időt és tárhelyet használó algoritmus, melynek segítségével az evolúciós távolságot meghatározó MCMC módszer egy lépéséhez szükséges futási idő $O(n^2)$-re, a tárhely pedig $O(n)$-re csökken, az eddigi $O(n^4)$, illetve $O(n^3)$-ről.