Genome Sequencing: Algorithms for New Approaches

Sekwencjonowanie genomu: algorytmy dla nowych podejść

Aleksandra Świercz

Ph.D. Thesis

Promotor: dr hab. inż. Marta Kasprzak

Poznan University of Technology

Institute of Computing Science

Poznań, 2007

I would like to thank:
My promotor Marta Kasprzak for patience, helpful comments and discussions,
My Father for being an example and for motivation to work,
Karol for forbearance and being with me in hard moments,
My Mother, who is continuously giving me a helping hand,
Lukasz Szajkowski and Piotr Gawron, whose help was invaluable.

CONTENTS

1. INTRODUCTION

1.1 Genome sequencing and the role of bioinformatics

For many ages people were attempting to find the structure of living organisms. They were looking for their origins and the rules for the evolution of the organisms. First simple experiments were limited to observations how individuals paired. In next steps one discovered the connections between features of the offspring and their parents. Soon many entirely new problems arose which are interdisciplinary in nature. The growing amount of information that was coming from different laboratories all over the world and need of an integration and exploration of all these data brought up to a creation of a new field *computational molecular biology*. The construction of computer science tools allowed for an analysis of huge amount of biochemical data and a development of mathematical and computer science methods expanded the range of experiments. Owing to that, determining the structure of organism genomes was possible. Revealing the sequence of nucleic acids and the structure of proteins led to learning their functions and to discovering origins of diseases, development of vaccines and drugs, and also obtaining different species of plants or cattle, e.g. more resistant to diseases or more fertile.

The analysis of molecules was firstly focused on deoxyribonucleic acids (DNA), as a main carrier of genetic information. Sequence analysis of DNA, in particular, has become an increasingly essential issue over the past few years, as we can see that nearly every research project in molecular biology involves sequence analysis and comparison [SK95, AH04]. Furthermore, the ultimate goal of the Human Genome Project [HGP03] was to sequence accurately the entire human genome. The basic approach to accomplish this task includes, from the computational point of view, three main parts: *mapping*, *assembling* and *sequencing* [Wat95, SM97, Gus97, Pev00,

FC03]. In the sequencing stage one can obtain the sequences of a length equal to a few hundreds of nucleotides. Next, during the assembling step, already sequenced chains are merged together into contigs, which are often whole genes. In the last step – mapping – the already assembled sequences are placed into proper place on a chromosome. In the case of sequencing of smaller genomes the last step is omitted and only the sequencing (called *small scale sequencing*) and assembling steps (called *large scale sequencing*) are performed.

Popular method of reading the DNA sequence in the small scale is *gel electrophoresis* (chain termination method) [MG77, SNC77], which does not need any computer analysis and generates not very long sequences, which are not resistant to experimental errors. The contaminants of the experiment may be caused by primers which stick to a second site of the sequence or by other molecules disturbing the PCR reaction and lead to wrong image of the gel. To increase the confidence of the results obtained in the experiment one can repeat many times the chain termination reaction or provide greater overlap of each nucleotide. The second option can be done by another approach, *sequencing by hybridization*, which reads longer sequences, but needs algorithmic computations due to large amount of data.

Sequencing by hybridization (SBH) was proposed in 1988 by Ed Southern [Sou88]. During the hybridization experiment the set of all fragments (*oligonucleotides*) that compose the original sequence is detected. In the standard approach all the fragments have the same length. The aim of the sequencing process is to reconstruct an original sequence of a known length from the fragments. Finding the solution can be done in polynomial time if there are no errors in the hybridization experiment [Pev89]. Unfortunately, two types of errors can appear in the input set of fragments: a *negative error*, if there is the lack of an oligonucleotide in the set despite of the presence of the fragment in the original sequence, and a *positive error*, if in the set an extra oligonucleotide appears, absent in the original sequence. In the case of errors in the input set, the problem of sequence reconstruction is computationally hard. Several approaches were devoted in recent years to overcome these difficulties [Pev89, BS88, LFK$^+$88, BKK$^+$97, BFK$^+$99, BFK$^+$00, BKK02, ZWZ03, BGK04] but still a great effort is needed to fulfill demands of biologists.

Recently, a novel isothermic approach to SBH was proposed [BFKM99]. This new method uses oligonucleotide libraries of equal melting temperatures (thus, differing

by oligonucleotide lengths) and, as explained later, may lead to overcoming one of the drawbacks of the standard SBH approach (with equal length oligonucleotide library). The latter sometimes cannot handle excessive rate of errors being a result of the biochemical phase of the SBH approach. Solving the isothermic sequencing by hybridization problem is similar to its standard counterpart, polynomially solvable in the case when there are no errors in the input set and computationally hard in the case of errors [BFKM00, BFKM04].

The next step of recognizing the sequence of genome – assembling – is also computationally hard. The assembling problem is much more complicated than the sequencing process because of larger amount of data it has to operate on and it must allow an inexact matches of neighboring fragments in the solution. Apart from this, the fragments can come from two sequences: the original sequence and its reverse complementary one, which was not the case in the sequencing step, where all the oligonucleotides had the same orientation (were coming from the same sequence).

A novel approach to large scale sequencing was proposed by 454 Life Sciences company [MEA+05]. In the classical approach DNA sequence assembling rely on merging the sequences coming from the sequencing step into a chain of a length up to a few millions of nucleotides. Input sequences are overlapping neighbors with a great shift and the average coverage, which is the average value of the number of fragments that cover each nucleotide in the sequence, is eight. In case of the large scale sequencing proposed by 454 Life Sciences company, the sequence recovery is divided into two stages. First one, which is done automatically by a Genome Sequencer 20 System, can obtain sequences of length about 100 nucleotides. These medium size chains are strongly overlapping every position and each place of the sequence is covered 30 times on average. Moreover, as the input for the second stage we get (together with the sequences) rates of confidence for a position of each nucleotide in the sequence. The greater the variance of nucleotides in the sequence, the greater the confidence that nucleotides were correctly obtained.

From the above overview one may see the great importance and influence the computational methods have on the genome sequencing projects. Without a development of the appropriate computational approaches, genome sequencing and recognition would be impossible in general.

1.2 Aims and scope

Approaches known from literature (described deeper in Chapter 3) leave a room for a further improvement to meet the demands of practitioners. The general goal of this thesis is to meet these requirements and propose new efficient and robust computational approaches for the first (sequencing) and the second (assembling) stages of the genome discovery process.

The sequencing in the small and large scales are the problems prone to experimental errors. New biochemical approaches (described in the following chapters) were recently proposed, which are supposed to diminish the number of errors. In the small scale sequencing the novel isothermic approach was developed, which decreases the number of negative errors by equalizing conditions of forming duplexes for all the fragments used in the experiment. In the large scale sequencing (assembling) the 454 genome sequencing was applied which increases the coverage of nucleotides and decreases the cost of the experiment.

For isothermic sequencing by hybridization and 454 genome sequencing problems none effective algorithms exist. These new approaches seem to be promising and thus there is a need to develop new methods and compare them to the already existing ones. These problems are computationally hard and exact methods solving instances of these problems may need too much time of computations or would never finish.

The aim of this work is to analyze the sequencing problem and to propose heuristics, which are time-cost effective and construct solutions close to the optimum ones. Next, the novel algorithms are compared in a computational experiment with the previous ones.

The organization of this work is as follows. Chapter 2 consists of basic terms of computer science and biochemistry, and of theoretical formulations of introduced problems. In Chapter 3 the already existing in the literature methods are described for different biochemical approaches, among them for the standard sequencing by hybridization (with equal-length oligonucleotides) and for the one with universal bases. Next, a new biochemical approach to isothermic SBH and the algorithm for the ideal case of SBH with isothermic libraries of oligonucleotides are presented. In the next section, the classical approach to large scale sequencing is presented, which is more commonly called the assembling. In the last section of this chapter the existing

approaches are summarized.

In Chapter 4 two new heuristic algorithms for the isothermic sequencing problem are presented. The tabu search one, which is a local search heuristic, and the genetic algorithm one, an evolutionary based method, both consider the specificity of the data – oligonucleotides of different length. These approaches may be also of value when designing new algorithms for the assembling stage of genome reading. In the genetic algorithm, the proposed new crossover operator that inherits some features of the structured weighted combinations [Glo94] may also be of value for some other combinatorial optimization problems for which the solution can be represented as a permutation, Traveling Salesman Problem (TSP) being one of the examples. It is in detailed described in a separate section (4.3). These methods are next compared in an extensive computational experiment, preceded by preliminary tests for tuning algorithms' parameters. The instances used for the comparison were prepared from real DNA sequences coding human proteins, taken from GenBank (http://www.ncbi.nlm.nih.gov/Genbank/index.html). The way of their preparation is also described in the chapter.

In Chapter 5 the isothermic approach to SBH is compared to standard SBH. The newly proposed algorithm for the standard sequencing has been based on the genetic algorithm for the isothermic sequencing (presented in Chapter 4). Such corresponding construction of the algorithms assures a credible comparison of the approaches, independent of program implementation. Next, these two algorithms have been compared also with other algorithms dedicated to the standard SBH problem. A particular type of errors coming from biochemical experiment, repetitions in the original sequences, is considered in more details. The extensive computational tests proved that the new proposition outperforms other existing algorithms in the literature and handles in a good way the case of repetitions (notorious hard case for DNA sequencing by hybridization).

A WWW bio-portal was created as a part of this work, where one can download the spectra for standard and isothermic DNA sequencing by hybridization, which were used in the computational experiments. There, also all the details referring to sequencing by hybridization are available: description of the approaches, literature and the program solving the isothermic sequencing problem with the tabu search algorithm. The details of the WWW bio-portal are put into Appendix A while the

portal can be found at http://bio.cs.put.poznan.pl.

Chapter 6 begins with description of the assembling problem. Next, novel algorithms for genome assembling are proposed which search for such a path in a graph that pass through all vertices. Finally, the results of computational experiment are presented. Data used for testing come from real biochemical experiments.

Chapter 7 concludes the results of the dissertation.

The thesis is complemented by 3 Appendices which contain, respectively, the details concerned with the WWW bio-portal (Appendix A), the accession numbers of the sequences from the GenBank which were used in the computational tests (Appendix B), and the detailed results of the tests of the hybrid genetic algorithm dedicated to isothermic and standard sequencing by hybridization problems (Appendix C).

2. BASIC DEFINITIONS

In this chapter basic notions from computer science and biochemistry, necessary to understand approaches and algorithms presented later on, are recalled. Since the formulated problems are of combinatorial nature, complexity issues arising in the context of such problems are coming first. Next, basic definitions from graph theory used for modeling the DNA sequencing problems in question and present basic algorithmic approaches are discussed. The chapter is complemented by necessary concepts of biochemistry allowing for a more formal presentation of the problem.

2.1 Basic notions of computer science

2.1.1 Computational complexity

The DNA sequencing problem in its formal setting belongs to the class of *combinatorial problems*. Without going into unnecessary details the latter can be thought of as a domain of problems which are characterized by a finite set of parameters taking values from a finite sets [GJ79, Bła88].

Combinatorial problems can be divided into two classes depending on a form of the solution: decision and optimization ones [GJ79, Bła88]. Note that both classes of problems can be represented as the class of *search* problems. The latter is bigger than only a union of the two classes and includes also "pure" search problems, as for example the problem of sorting. *Decision problems* are in the form of a question for which the answer is "yes" or "no". On the other hand, *optimization problems* require a complete solution reached by maximizing or minimizing a criterion function. It is clear that for any optimization problems there exists a corresponding decision version. Moreover, the decision version of a problem is not harder than its optimization version. Consider for example the knapsack problem. We have n elements in set $A = \{a_1, a_2, ..., a_n\}$ and for each element $a_i \in A$ the size $s(a_i)$ and the profit $w(a_i)$ is

defined, as well as constants b, the size of a knapsack and y, the threshold value of the profit.

In the decision version of the problem we ask: *Is there a subset* $A' \subseteq A$ *such that* $\sum_{a_i \in A'} s(a_i) \leqslant b$ *and* $\sum_{a_i \in A'} w(a_i) \geqslant y$*?*

In the optimization version we *look for a subset* $A' \subseteq A$ *with a maximum profit and such that* $\sum_{a_i \in A'} s(a_i) \leqslant b$*.*

The set of all instances of decision problem Π is denoted by D_Π and the subset $Y_\Pi \subseteq D_\Pi$ contains all the instances for which the answer is "yes". An *instance* I of problem Π is obtained by assigning values to all the problem parameters. It can be written as a string of the parameters decoded with the use of symbols over a finite alphabet. A similar definition can be also given in case of an optimization problem. The length of this string is the *size of the instance*. If numbers occurring in the problem instance are represented in binary or in any other base greater than 1, and in the symbol string coding data there are no unnecessary symbols, such an *encoding scheme* is *reasonable*. In the following, the reasonable scheme of encoding instances and the realistic computer model (like e.g. deterministic Turing machine) is assumed. Computational time of an algorithm solving a problem depends on the instance size of the problem. If the time function of an algorithm can be bounded by a polynomial depending on the instance size, we call such an algorithm *polynomial* in time. Other algorithms, for which the time function cannot be polynomially bounded are called *exponential* in time. If the computational time of an algorithm can be expressed by a polynomial function dependent both on the instance size and on the greatest number from the instance, the algorithm is *pseudo-polynomial* in time.

Unfortunately, not for all combinatorial problems polynomial algorithms exist. This can be analyzed by studying complexity of the problem in question. From what we said it follows that when studying the hardness of an optimization problem we can limit ourselves to its decision counterpart being not harder than the original problem. Thus, further complexity will be presented for decision problems only.

If the correctness of a solution of a problem can be verified in time polynomially dependent on the instance size, such a problem belongs to *class NP*. However, the verification of the solution of the problem in polynomial time does not mean that the problem is polynomially solvable. If the problem can be solved in polynomial time it belongs to *class P*, where P$\subseteq$NP. Till now it has not been proven that P$\neq$NP.

Computationally hardest problems in class NP are classified as *NP-complete* problems. They are solved with exponential-time or pseudo-polynomial-time algorithms. An NP-complete problem for which any pseudo-polynomial-time algorithm does not exist is *strongly NP-complete*.

A proof that a decision problem Π_1 is NP-complete consists of a construction of a *polynomial transformation* $f : D_{\Pi_2} \rightarrow D_{\Pi_1}$ of a known NP-complete problem Π_2 to the considered one in such a way:

- for every instance $I_2 \in D_{\Pi_2}$ the answer is "yes" if and only if the answer for $f(I_2) \in D_{\Pi_1}$ is also "yes"
- the time of calculating the function f for every instance $I_2 \in D_{\Pi_2}$ is bounded from above by a polynomial in the size of I_2.

It is not possible to solve a problem proven to be *NP-complete* with a polynomial-time algorithm on condition P≠NP. Proving that a problem is *strongly NP-complete* consists in construction of *pseudo-polynomial transformation* of some strongly NP-complete problem to the considered one. The pseudo-polynomial transformation proceeds in time having as upper bound a polynomial dependent on the instance size and on the greatest number from the instance. Moreover, the greatest number of the instance cannot increase exponentially, the instance size cannot decrease exponentially, and the answer for the instance before the transformation is "yes" if and only if the answer for the instance after the transformation is also "yes".

Optimization problems for which their decision counterparts are proven to be NP-complete belong to the class of *NP-hard* problems. Similarly, if a decision version of a problem is strongly NP-complete, its optimization version belongs to the class of *strongly NP-hard* problems. However it is possible that the decision problem is solvable by a polynomial-time algorithm while its optimization counterpart is NP-hard/strongly NP-hard.

One of the examples of NP-complete problems is the knapsack problem, which was already defined in this section [GJ79].

2.1.2 Graph theory

Graphs are extremely useful in formulating and solving a broad class of problems with a discrete set of parameters. We will use them in solving DNA sequencing problems.

Basic notions are recalled below.

A *graph* can be *directed* or *undirected* [CLRS01, CL86]. A *directed graph* or *digraph* G is an ordered pair $G = (V, E)$, where V is a finite set of *vertices* (sometimes called *nodes*) in G and E is a set of ordered pairs of elements from the set V, called *arcs* or *directed edges*. An arc $(u, v) \in E$ is a directed edge from u to v; vertex u is a direct *predecessor* of v and vertex v is a direct *successor* of u. In the directed graph the arc which starts and ends at the same vertex is called *loop*. We can define the *out-degree* in the digraph, which is the number of arcs starting from this vertex, and the *in-degree* which is the number of arcs, that finish in this vertex.

An *undirected graph* G is an ordered pair (V, E), where V is a finite set of vertices in G and E is a set of unordered pairs of vertices, called *edges*. In an undirected graph an edge (u, v) is equal to an edge (v, u). A *degree* of a vertex in the undirected graph is the number of adjacent edges.

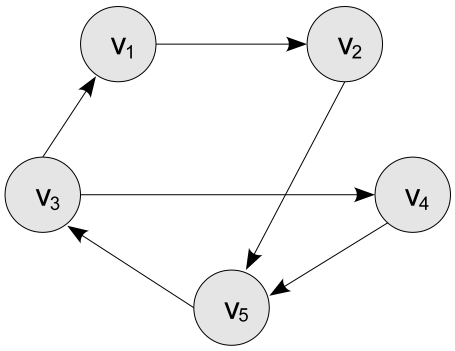
A *path of length* k from vertex u to vertex v in graph $G = (V, E)$ is a series of vertices $\langle v_0, v_1, v_2, ..., v_k \rangle$ such that $u = v_0$, $v = v_k$ and $(v_{i-1}, v_i) \in E$ for $i = 1,2,...k$. If all the vertices in the path are different, the path is called *simple*. A *cycle* in a graph is a path $\langle v_0, v_1, v_2, ..., v_k \rangle$, where $v_0 = v_k$ and it contains at least two arcs/edges. Moreover if the vertices $v_1, v_2, ..., v_k$ are different, the cycle is called *simple*. A graph without any cycles is called *acyclic*. A simple cycle, which contains all the vertices from V, is called a *Hamiltonian cycle*. A simple path containing all the vertices from V is called *Hamiltonian path*. If a path passes by all the arcs/edges from graph G exactly once then it is called an *Eulerian path*. Moreover, if the path form a cycle then, such a *cycle* is called *Eulerian*. An example of a directed graph, Hamiltonian and Eulerian (directed) paths is presented in Figure 2.1.

A digraph is bipartite if set V can be partitioned into two sets V_1 and V_2 in such way that every arc from set E connects two vertices, one from set V_1 and the other from set V_2. Hereafter, it is assumed that an arc starts from a vertex in V_1 and enters a vertex from V_2. A complete bipartite graph is denoted $K_{m,n}$, where $m = |V_1|$ and $n = |V_2|$.

A *network* is a digraph without loops and has two specified vertices: source s with the in-degree equal to zero and sink t with the out-degree equal to zero.

A graph is a *p-graph* if for any ordered pair of vertices (x, y) there exist at most p parallel arcs from x to y. The *adjoint* G' of a graph $G = (X, V)$ is the 1-graph

Fig. 2.1: An example of a digraph $G = (V, E)$. The set of vertices is $V = \{v_1, v_2, v_3, v_4, v_5\}$. The set of arcs: $E = \{(v_1, v_2), (v_2, v_5), (v_3, v_1), (v_3, v_4), (v_4, v_5), (v_5, v_3)\}$. A Hamiltonian path (one of the possible paths) is a series of vertices: $\langle v_1, v_2, v_5, v_3, v_4\rangle$. There exist more than one Eulerian path and one of them is: $\langle v_3, v_1, v_2, v_5, v_3, v_4, v_5\rangle$. In graph G neither Hamiltonian nor Eulerian cycle exist.



whose vertices represent arcs of G and which has an arc from x to y if and only if the terminal endpoint of the arc x is the initial endpoint of the arc y in the graph G. A graph is a directed line-graph if and only if it is an adjoint of a 1-graph.

A digraph is *connected* if for every pair of its vertices u and w we can find a sequence of vertices $\langle v_1, v_2, ..., v_k\rangle$ such that $v_1 = u$ and $v_k = w$, and $\{(v_i, v_{i+1}) \in E \vee (v_{i+1}, v_i) \in E\}$, $i = 1, 2, ..., k-1$. A graph is *complete* if for every two distinct vertices x and y at least one of the arcs (x, y) or (y, x) is present in the graph. A *complete symmetric* graph has both arcs (x, y) and (y, x) for every two distinct vertices x and y. A *multigraph* is a graph that allows parallel arcs and loops.

A *tree* is a graph for which its underlying undirected graph is acyclic and connected. *Root* r in directed tree T is a vertex, such that T contains a path from r to any other vertex v and there is no path from any vertex $v \neq r$ to vertex r. Vertices of T having out-degree equal to 0 are called *leaves*.

The Traveling Salesman Problem (TSP) is defined for a graph in which vertices correspond to the cities which the salesman has to visit. All pairs of vertices are connected by arcs and a cost is assigned to every arc, which corresponds to the distance between the two cities. The salesman has to visit all the cities traversing the smallest total distance, which is equivalent to finding a Hamiltonian cycle in the graph with the smallest total cost. It is also possible to assign another cost with an

arc which stands for example for a road toll. In such a graph a Hamiltonian cycle is looked for with the minimization of both costs (the criteria can be weighted – depending on a problem). Yet another is defined by a graph where, besides costs on the arcs, profits are assigned to nodes and the salesman has to maximize the total profit on visited nodes (here, not all vertices must be included in the path), while not exceeding the assumed cost of the cycle (Selective Traveling Salesman Problem).

2.1.3 Basic algorithms

In this section basic algorithmic approaches will be described, used to solve combinatorial problems, which will be further on applied and customized for the DNA sequencing problems.

Genetic algorithm is an algorithm that simulates natural evolution process of a population. The method was first proposed by Holland in [Hol75] and has been applied to many different combinatorial problems [AL97, VMOR98]. The algorithm starts with an initial population of *individuals*. All individuals are evaluated by the *fitness function*. Next, the individuals are *selected* to the pool of parents from which the next generation will be created. The selection process, based on the fitness value, rewards the individuals with the highest fitness value while unfit members of the population die away. Such procedure (treatment, conduct) ensures that individuals from the consecutive populations are more fit, and finally the algorithm will converge to the local or global optimum. The common selection procedures can be stochastic (roulette wheel selection, stochastic remainder without replacement selection, stochastic remainder with replacement selection), deterministic (deterministic selection, tournament selection) or a mixture of both above approaches (part sum selection).

To the fittest individuals from the parent population *genetic operators* are applied and this results in the creation of a new *offspring* population. The genetic operators are *crossover* operator, which inherits some features from parents, and *mutation*, which occurs very seldom and is changing randomly one of the feature in aparent. The goal of the crossover is to keep good characteristics while the objective of the mutation is to introduce diversity in the population. The individuals from the offspring are next selected for a new generation. The process of the reproduction is repeated until a *stopping criterion* is met, usually it is a number of iterations without improvement

of the objective function.

Tabu search is the algorithm based on local search, proposed by Glover [GL97]. The method searches deterministically for local optimum, but on the other hand, allows jumping into another place in the solution space. It can be reached by searching in a local neighborhood and choosing the best move. The general idea of the algorithm is as follows. For each solution x, its *neighborhood* $N(x)$ can be determined, which is composed of all the solutions one move apart from x. Starting from an initial solution, the method chooses the best solution in the consecutive neighborhoods usually increasing a value of the objective function at the same time. In order to extend the search in the feasible solution space, beyond the local optimum area, the algorithm can *restart* from a different, randomly selected point of the space.

Tabu search allows also for such moves which do not increase the objective function value. Considering the maximization problem $max\{\gamma(x) \mid x \in \mathcal{S}\}$, where γ is the objective function and $\mathcal{S}$ is the solution space, the transition from y to x can result in a negative increment $\Delta = \gamma(x) - \gamma(y)$. Then, in the next iteration there is the tendency to improve the objective function value by returning to the local optimum – the solution y. For preventing such cycles the *tabu list* is introduced, which stores the list of forbidden moves. The goal of the tabu list is not to eliminate all the moves from the list but to make all such moves less probable. Each move deteriorating the objective function is put into the tabu list for a particular number of iterations. It means that obtaining the solution from which we "came", is hardly probable. Since keeping the transformations (moves) may prevent from acceptance legal solutions which have not been considered yet, an additional rule based on an aspiration criterion was introduced, which is an 'exception from the rule'. The additional rule defines that forbidden moves can be accepted if an aspiration criterion is fulfilled. Usually, the aspiration criterion is fulfilled when the solution found is the best among the solutions found so far or when a neighborhood is empty.

2.2 Basic concepts of biochemistry

This section contains basic issues of biochemistry necessary to formulate in an exact manner the problem of DNA sequencing.

Deoxyribonucleic acid (DNA) is a double helix: two chains are twisted together

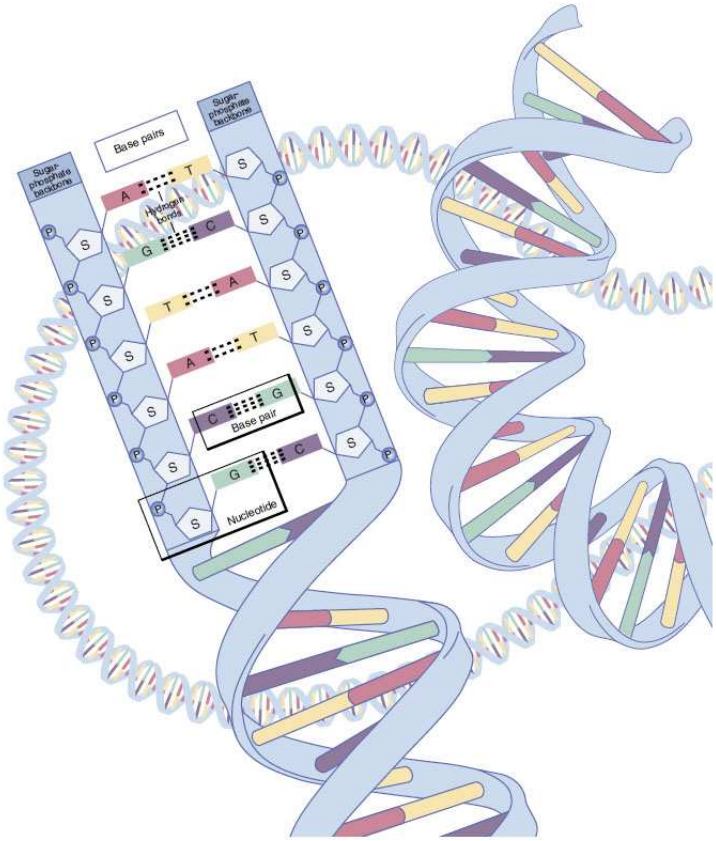
and joined by hydrogen bonds. Each chain in the double helix is composed of molecules called nucleotides. Nucleotides contain sugar (deoxyribose), phosphoric acid and nitrogenous base (Figure 2.2). They differ only in their nitrogenous base (adenine – A, cytosine – C, guanine – G, thymine – T). The order of bases codes genetic information, symbolically written as a string of letters A, C, G, and T. The double helix is joined by double or triple hydrogen bonds between bases in nucleotides. 'A' from one chain is paired with 'T' from the other chain and respectively 'G' with 'C' (Figure 2.3). This rule of pairing is called *complementarity* (sometimes also called Watson-Crick principle, following Chargaff's rule, cf. [Wat95]) and it ensures that knowing a sequence of nucleotides in one chain one can always determine a complementary sequence of bases in the other chain. The length of a DNA fragment is expressed in the number of nucleotides or bases and is denoted as *base pairs* (*bp*). For example, the length of the human genome (*Homo sapiens*) is about $3 * 10^9$ bp , mouse (*Mus musculus*) – $3,3 * 10^9$ bp, fly (*Drosophila melanogaster*) – $1,4 * 10^8$ bp , bacteria (*Escherichia coli*) – $4,64 * 10^6$ bp. A short fragment of a DNA sequence is called an *oligonucleotide*.

As it was pointed in Introduction, the process of genome recognition may be divided into three stages: mapping, assembling and sequencing. *Mapping* is the step of reading the DNA sequence where the restriction map is build. The whole DNA chain is digested with the restriction enzymes in particular sites, which are characteristic for the enzyme used in this experiment. The sites where the enzymes cut the sequence are next marked on the restriction map. This map is used at the end to build the sequence from the fragments of length up to 10^6 nucleotides, coming from the assembling step.

In the *assembling* step the long fragments of the DNA sequence (even 10^6 nucleotides) are constructed from sequences (of length up to a few hundreds of nucleotides) coming from the sequencing stage. Here, computational methods are used to combine the short sequences into longer fragment or fragments, called *contigs*. Depending on the method which was chosen for the sequencing, the input sequences can contain different rate of errors (wrongly read nucleotides) and may come from both strands of the DNA helix. Thus, the problem is hard to solve and there is need for developing efficient algorithms for the assembling problem.

The *sequencing* step is the stage of reading the DNA sequence on the lowest level.

Fig. 2.2: The structure of DNA [HGR]

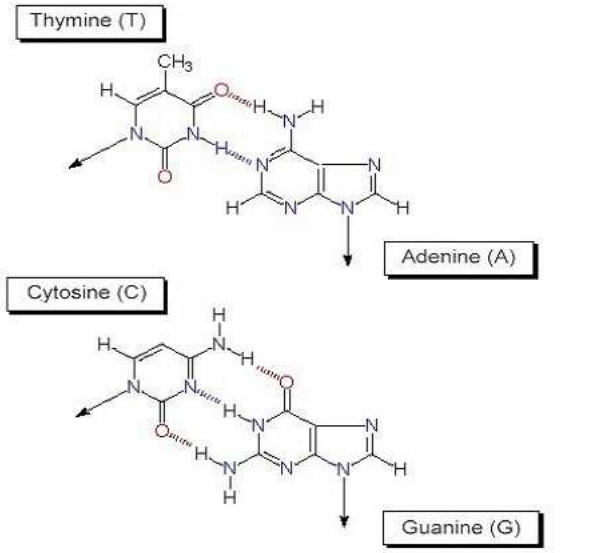


Before the sequencing, there are a few steps required for preparation of a fragment of DNA. First, a DNA sequence of an organism is isolated. Next, it is replicated (copied) with the use of special molecules (called *vectors*), like for example plasmids or BACs. After the replication, the copied chains are disjoined into single strands of the helix. These strands are cut into shorter fragments (of length up to a few hundred nucleotides) with a shotgun method, and some of the fragments are sequenced.

There are a few approaches used for reading the order of the nucleotides in the fragments, among them gel electrophoresis, pyrosequencing (which will be described later on), and *sequencing by hybridization (SBH)*. The latter is composed of two phases: biochemical (the hybridization) and computational ones. Let us note, that the computational phase of SBH is similar to the assembling stage, thus, the importance of designing good algorithms for SBH exceeds by far this particular approach.

The *hybridization experiment* is based on the principle that a one-strand DNA fragment hybridizes (joins) with the complementary fragment of the other strand. Its

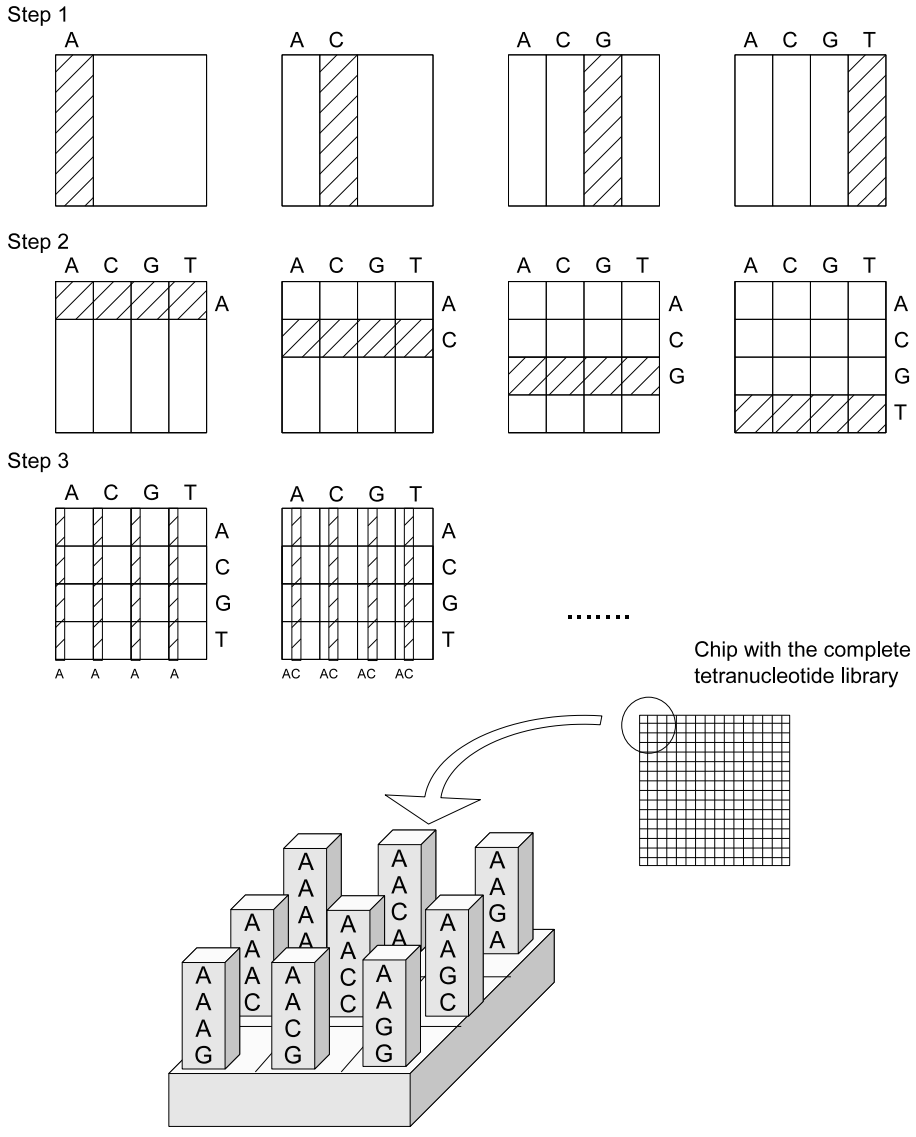
Fig. 2.3: The hydrogen bonds between pairs of complementary bases



aim in the *standard* DNA sequencing is to detect all subsequences (oligonucleotides) of a given length l (usually l is equal 8-10 nucleotides) that are contained in the *target* sequence of length n. For this purpose a library of all possible 4^l short DNA sequences of length l is prepared. The large number of oligonucleotides that has to be placed on a square plate, made of glass or silicon, needs an advanced chip (microarray) technology [Sou88, FRP$^+$91].

The DNA chip consists of as many cells as many oligonucleotides in the library are. The library of oligonucleotides is constructed on the plate. Depending on the technology, length of the oligonucleotides, and the quality of the prepared microarrays [MLB$^+$96, Agi06], for each oligonucleotide the square with the side of size 10 - 65 μm is dedicated, and the size of the plate depends on the size of the cells. For example, to prepare a chip with all tetranucleotides (oligonucleotides of length 4) one must prepare a plate composed of 256 cells arranged in the square 16 $\times$ 16 (Figure 2.4). Each cell is filled with a different oligonucleotide according to a special procedure. In the first step 1/4 of the cells are provided with adenine, and the other quarters with cytosine, guanine, and thymine, respectively. These nucleotides are sticked to the surface and they are the beginning of the oligonucleotides. In the next steps each oligonucleotide is extended by 1 nucleotide. At the end the plate is filled with all possible oligonucleotides of a given length, each of them is sticked to a different place

Fig. 2.4: The construction of a microarray chip.



on the plate. A position of an oligonucleotide can be determined on the basis of the coordinates of the chip.

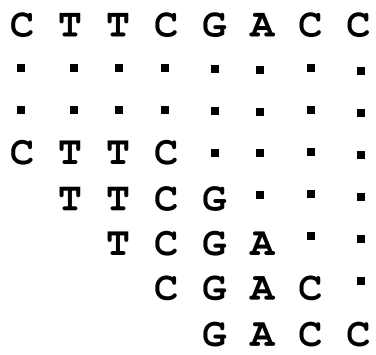
The chip with the library of oligonucleotides that are capable to take part in the hybridization experiment, is introduced to a solution (in a chemical sense) with many copies of the examined DNA sequence (single stranded), which are labeled fluorescently or radioactively. Under favorable hybridization conditions these DNA sequences hybridize with oligonucleotides from the array complementary to fragments of the sequence. (Note, that we have many copies of the sequence, each hybridizing with a different oligonucleotide.) After the reaction, on the fluorescent image of the chip the most intensive spots are the places where the DNA sequence joined entire oligonucleotides. Knowing coordinates of these points one can determine the oligonucleotides which are the fragments of the examined DNA sequence (they are complementary to the oligonucleotides from the most intensive points). The set of these oligonucleotides is called *spectrum*. The *computational phase* of the sequencing process is to reconstruct the examined sequence from the oligonucleotides in the spectrum, which is to determine the order of the elements in the spectrum.

If the hybridization experiment were executed without any errors, the spectrum would be ideal and it would be composed of all the oligonucleotides of the same length l, that can be recognized in the target sequence of length n. The length n of the sequence can be determined by an additional experiment – polyacrylamide gel electrophoresis. The ideal spectrum consists of $n - l + 1$ oligonucleotides. The goal is to combine all the oligonucleotides from the spectrum in such a way that the neighboring oligonucleotides are shifted by one nucleotide (see Example 2.1). This is done in the computational phase.

Example 2.1: *Suppose the original sequence to be found is CTTCGACC. The length of the sequence is* $n = 8$. *In the hybridization experiment all the tetranucleotides (oligonucleotides of length 4) are used. The library of oligonucleotides is composed of* $4^4 = 256$ *oligonucleotides:* $\{AAAA, AAAC, AAAG, AAAT, AACA, AACC, \ldots, TTTC, TTTG, TTTT\}$. *After the hybridization experiment one can get the spectrum, that is the set of oligonucleotides that are parts of the target sequence. In the ideal experiment we get (after reading the complementary bases) spectrum* $=\{CGAC, CTTC, GACC, TCGA, TTCG\}$. *See Figure 2.5.* □

In the hybridization experiment some errors occur. The errors can be of two types:

Fig. 2.5: The target sequence is reconstructed from the oligonucleotides from the spectrum. The neighboring oligonucleotides are overlapping on $l-1$ nucleotides and are shifted by one position.



a *negative error* appears when there is the lack of an oligonucleotide in the spectrum despite the oligonucleotide can be distinguished in the target sequence, and a *positive error* which is an extra oligonucleotide in the spectrum that is not contained in the target sequence. Mainsprings while negative errors occur in the spectrum are:

- Due to unfavorable conditions, e.g. temperature or salt concentration, the hybridization between the oligonucleotide on the chip and the examined DNA fragment does not happen. There is no light signal detected during the experiment. The oligonucleotide is not recognized as a part of a sequence. (see Table 2.1, row (1)).
- Some oligonucleotides are contained in the sequence more than once. Since spectrum is not a multiset, the oligonucleotide appears in the spectrum only once (see Table 2.1, row (2)). The errors of such type are called *repetitions*.

There are two sources of positive errors in the spectrum:

- An oligonucleotide, which is not quite complementary to the DNA sequence, hybridizes partially with it. The light signal is emitted on the chip and the improper oligonucleotide is added into the spectrum (see Table 2.1, row (3)).
- Sometimes the fluorescent image of the chip is noisy and the oligonucleotide can be included into the spectrum by mistake. (see Table 2.1, row (4)).

Tab. 2.1: Examples of an erroneous spectrum. In the first row (1) one oligonucleotide in the spectrum is missing (underlined in the sequence). (2) In the sequence the same oligonucleotide (underlined) is contained twice. (3) One positive error in the spectrum. By mistake the oligonucleotide complementary to CTTA hybridizes with a similar fragment in the sequence, CTT or CTTC. (4) The underlined oligonucleotide, due to a noisy image of the chip, has entered the spectrum, although it is not similar to any part of the sequence.

	sequence	spectrum
(1)	CTTCACTTG	ACTT, CACT, CTTC, CTTG, TTCA
(2)	CTTCACTTC	ACTT, CACT, CTTC, TCAC, TTCA
(3)	CTTCACTT	ACTT, CACT, CTTA, CTTC, TCAC, TTCA
(4)	CTTCACTT	ACTT, CACT, CTTC, GGCC, TCAC, TTCA

Therefore, as a result of the hybridization experiment, one gets a spectrum, in which not all the elements are parts of the original sequence and not every element that is contained in the sequence appears in the spectrum. Usually, no additional information is provided about the spectrum elements (as, for example, the probabilities of existence of the oligonucleotides in the original sequence, or about precedence relationships between elements in the spectrum). However, sometimes it is possible to obtain knowledge about the first oligonucleotide in the original sequence. Examined DNA fragments are often cut out from a genome with restriction enzymes, and then are amplified with the polymerase chain reaction (PCR). Restriction enzymes cut a sequence in the places of known defined fragments of a few nucleotides, while in PCR some primers are used, which are known beginnings of the amplified fragments.

Selecting suitable parameters for the hybridization experiment, like temperature or salt concentration, we can eliminate with high probability some types of errors at the cost of increasing the other type. On the other hand, some types of errors are systematic, meaning that they are likely to repeat each time an experiment is run. For example, palindromic sequences tend to form secondary structures which interfere with hybridization. Therefore, we cannot be sure that all errors of one type are absent in the spectrum. The number of errors coming from repetitions can be reduced by

extending the length of the oligonucleotides l. Unfortunately l cannot be increased to infinity, because the size of the oligonucleotide library grows exponentially with l. Thus the algorithms should deal with all types of errors.

The assumed temperature is constant during the hybridization experiment. In the standard DNA sequencing the oligonucleotides from the library are of equal length, while they differ in the content of G/C and A/T. It is known that duplexes of C/G rich oligonucleotides are more stable than A/T rich ones. This property may result in numerous errors of the positive type in the standard hybridization approach. At the same time the modification of hybridization conditions directed towards diminishing the number of imperfect duplexes of C/G rich oligonucleotides may result in increasing the number of missing perfect duplexes of A/T rich oligonucleotides. This means that the modification that decreases the number of positive experimental errors might increase the number of negative ones. The duplex formation depends on the base composition but also on its length. In some models a simple equation was used to calculate melting temperatures of oligonucleotide duplexes assuming 4 degrees for every G/C pair and 2 degrees for every A/T pair [WJH$^+$81]. (More accurate procedure of determining the temperature of the oligonucleotide duplexes is based on the nearest neighbor model [BFBM86].) Thus, at least formally, it is possible to compensate lower stability of A/T rich duplexes by increasing their length. It is known that the above description is not very accurate although it reflects a general relative stability of different duplexes quite well. Therefore, to form more stable duplexes with hybridized sequences, a new approach to SBH has been introduced in [BFKM99] and [BFKM00].

The key idea of this new approach is to obtain a set of oligonucleotides that differ in base composition and length, and are characterized by a predefined relation between these two parameters. In a specific case, if in a library the increment of C or G is twice of A or T, and the sum of increments for all oligonucleotides is constant, then such a library is called *isothermic*. In what follows, the sum of increments of nucleotides forming an oligonucleotide will be called the oligonucleotide *temperature*.

Isothermic oligonucleotide library follows the experimentally established relationship between the base composition and the duplex stability. Oligonucleotides contained in such a library should form duplexes with their complements in a more narrow range of experimental conditions (temperature, salt concentration etc.) than

that characteristic for an oligonucleotide library with oligonucleotides of the same length. Therefore, the hybridization experiments performed with isothermic libraries should result in a smaller number of experimental errors. The use of such libraries should substantially limit the number of these errors to be considered in the computational phase of the SBH approach.

A formal definition of the isothermic oligonucleotide library was presented in [BFKM04]. In this definition w_i denotes the increment of a nucleotide of type i ($i \in \{\text{A,C,G,T}\}$) added to the melting temperature of the oligonucleotide containing it, and x_i denotes the number of appearances of a nucleotide of type i in the oligonucleotide.

Definition

An *isothermic library* L of temperature t_L is a library of all oligonucleotides satisfying the relations: $w_{\text{A}}x_{\text{A}}+w_{\text{C}}x_{\text{C}}+w_{\text{G}}x_{\text{G}}+w_{\text{T}}x_{\text{T}} = t_L, w_{\text{A}} = w_{\text{T}}, w_{\text{C}} = w_{\text{G}}$, and $2w_{\text{A}} = w_{\text{C}}$.

Without loss of generality we may assume that $w_{\text{A}} = w_{\text{T}} = 2$ and $w_{\text{C}} = w_{\text{G}} = 4$.

Let us now discuss properties of isothermic libraries. One can easily realize that one such a library may be not enough. For example, it is not possible to cover DNA chains composed of only C and G nucleotides by oligonucleotides from a library of a temperature not divisible by 4. Moreover, a library of a temperature divisible by 4 is not sufficient to cover sequences where one nucleotide of A or T type is surrounded by only G or C nucleotides. But, on the other hand, two isothermic libraries with temperatures differing by one increment of A or T, i.e. by 2 deg, are sufficient to perform a proper hybridization experiment with any target sequence and, moreover, shifts between oligonucleotides from the libraries covering the sequence will be always less or equal to 1 (the proof can be found in [BFKM04]).

The cardinality of the isothermic library of temperature t is between the cardinalities of the standard libraries involving, respectively, only the shortest ($t/4$ or $\lceil t/4 \rceil$) or only the longest ($t/2$) oligonucleotides of equal length.

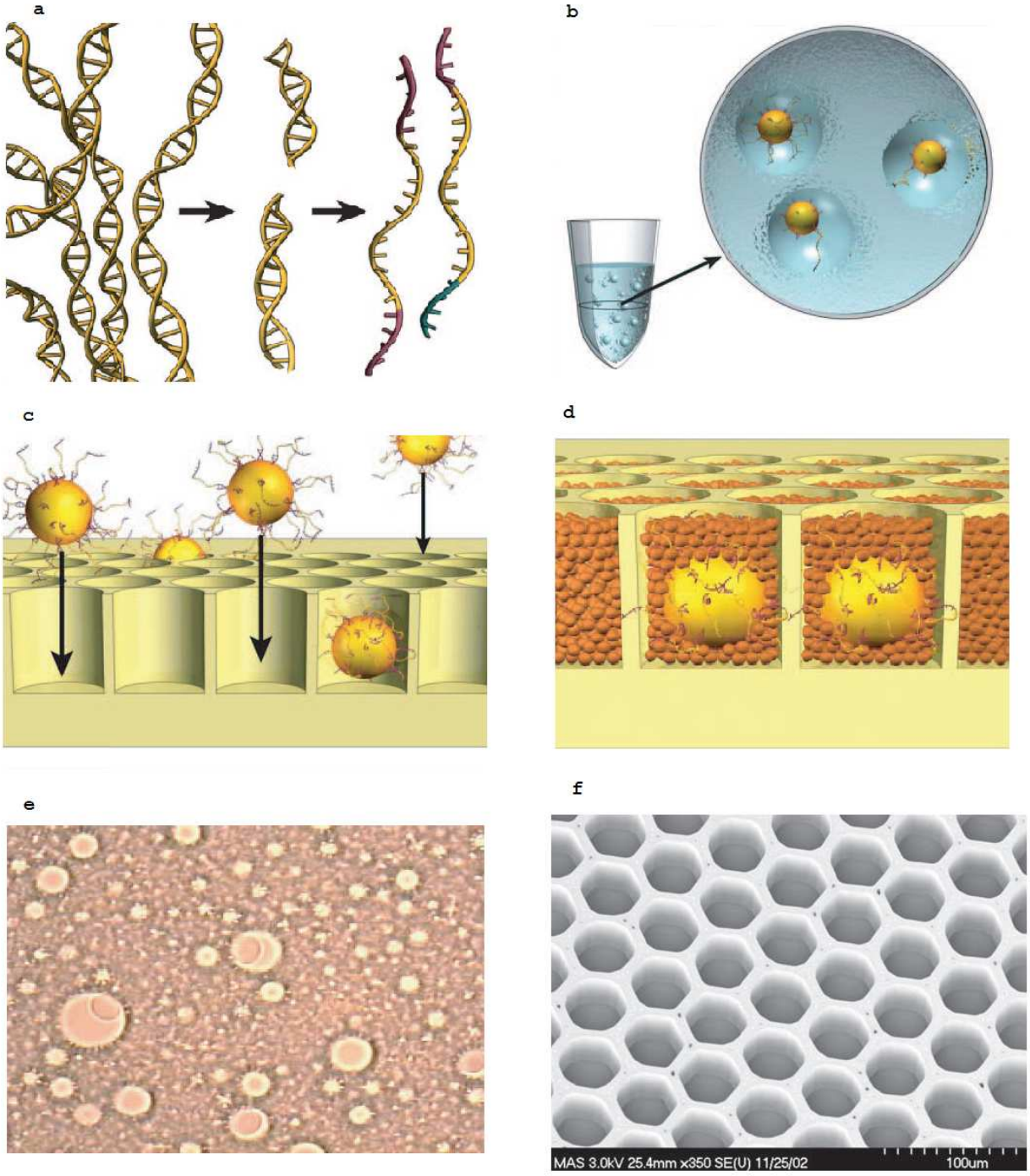
Pyrosequencing is a novel method of DNA sequencing developed by Mostafa Ronaghi ([RKP+96]). This method allows for a sequencing of a single strand of DNA by synthesizing a complementary strand along it. Each time after a nucleotide A, C, G, or T is incorporated into a newly created chain, many cascade enzyme reactions

are triggered what at last ends in light emission. The number of nucleotides of one type, which joins the complementary sequence in one step, depends on the number of the consecutive nucleotides of the same type in the sequence. The light signal is proportional to the number of nucleotides incorporated and is detected on the *charged coupled device (CCD) camera*. After each cycle the rest of not joined nucleotides are washed up and the process is continued with another nucleotide. It is possible to analyze a large number of samples in the same time. The pyrosequencing process is fully automated and with relatively low cost comparing to other sequencing methods. However, this method has serious limitations with short reads, which makes the process of genome assembling much more complicated.

Recently 454 Life Sciences company has applied the pyrosequencing approach for assembling short genomes [MEA$^+$05]. In this approach the step of sequencing is done automatically. The Genome Sequencer 20 System (GS20) is able to resolve hundreds of thousands of nucleotides per one run. The process of preparation of the sequence is presented in Figure 2.6.

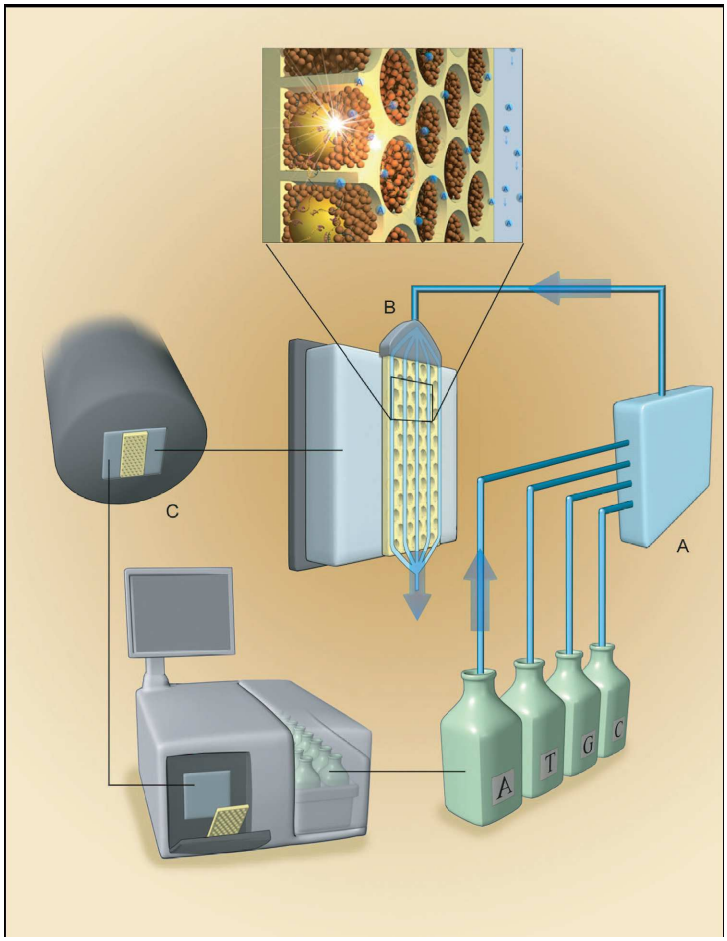
At the beginning a DNA sequence is isolated, randomly fragmented and the adapters (short unique sequences) are ligated to both ends of the fragments. Next the fragments are separated into single strands (see Figure 2.6 (a)). Later on the DNA fragments are captured by beads in the PCR-mixture of water-in-oil emulsion in such a way that only one fragment can be attached to a single bead. In every droplet of the emulsion there is only one bead (see Figure 2.6 (b) and (e)). The DNA fragments are amplified in each droplet, and at the end of the reaction each bead is carrying ten millions copies of the DNA sequence, that was attached to it. The sequences on the beads are prepared for the sequencing process: they are denaturated (double strands, produced in the PCR reaction, are separated into single ones) and the primer is added into the beginning of the sequence. Next the beads are placed in specially prepared wells on a fiber-optic slide, called Pico TiterPlate device (see Figure 2.6 (f)). This device can contain up to 1.6 million wells, each of them of width 44 μm, and have enough room only for one bead with sequences (see Figure 2.6 (c)). Into the wells smaller beads of enzymes (sulfurylase and lucyferase) are added which are necessary for further chemical reactions (see Figure 2.6.d). Such prepared sample is next used in the GS20.

Fig. 2.6: Preparation of DNA fragments for the sequencing experiment: the fragments are separated into single strands (a); DNA fragments are captured by a bead in water-in-oil emulsion (b) and in one droplet of the emulsion there is only one bead (e); beads are placed in Pico TiterPlate device (f) in such way that in a single well there is only one bead (c) (Adapted by permission from Macmillan Publishers Ltd: Nature, [MEA+05], 2005).



The instrument which uses the 454 technology is composed of the fluidic part (sequencing reagents – nucleotides), a flow chamber in which the nucleotides are introduced into the DNA beads placed in the wells on the Pico TiterPlate device, the optical part (CCD camera) where the light signal of the sequencing reaction is detected, and a computer which analyzes the data and presents the output data to the users (see Figure 2.7).

Fig. 2.7: Sequencing device. The Genome Sequencer 20 is composed of the following parts: fluidic assembly (A), a flow chamber which includes the fiber-optic slide with wells and beads (B), a CCD camera-based imaging assembly (C), and a computer that provides the necessary user interface and instrument control. Adapted by permission from Macmillan Publishers Ltd: Nature, [MEA+05], 2005.



The process of resolving the DNA fragments proceeds as follows: in the flow cham-

ber the nucleotides are cyclically delivered, each time only one type of nucleotides (A, C, G or T) is present. The nucleotides are incorporated to these DNA fragments for which the complementary nucleotide is next in the reconstruction process. It is possible that more than one nucleotide of one type in one step is incorporated, if the DNA fragment contains a sequence of the same nucleotides. Nucleotide incorporation generates pyrophosphate molecule which together with enzymes: sulfurylase and lucyferase induces the light emission, which is later on detected on the CCD camera. The signal intensity of each nucleotide flow, for each well, indicates the number of nucleotides attached to a single sequence. The image on the CCD camera is next transfered to the computer, which analyzes the data. At the beginning, the images are translated into *flowgrams*; for each well a different flowgram is made. Next, from the flowgrams, the DNA sequences are obtained together with *quality scores*, which evaluate the confidence of each nucleotide position. After each cycle of nucleotide flow, the rest of unbinded nucleotides are washed away. The process of incorporating the nucleotides is repeated for a several cycles of A, C, G, and T flow. After the reaction millions of sequences are obtained, each of length around 100 nucleotides.

Large scale sequencing, also called *sequence assembling* , is the next step – after the sequencing – of recognizing genetic information. The sequences over the alphabet {A,C,G,T} (the letters stand for four nucleotides) of length up to 700 nucleotides, obtained in the sequencing step, are merged together into longer contigs or even a whole genome (for example a genome of a bacteria).

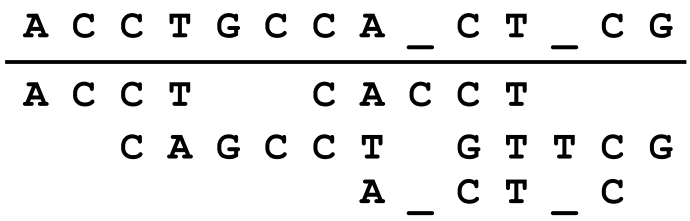
It was already mentioned in this section, that the process of preparing a DNA chain for the sequencing, causes that the short fragments are coming from both strands of a DNA molecule. Also due to a shotgun method, which randomly cut DNA sequences, the fragments randomly overlap the DNA sequence. Some parts of the sequence may be covered by many sequences, while some parts might not be covered at all (i.e. due to problems in replication). Usually, the coverage is in the order of 8-15 fragments. The input sequences generally have different lengths and there may exist an inclusion relation between them. The solution is a long sequence (or fragments of the examined sequence, called contigs, if the sequence is not covered in some parts) which is composed of all the input sequences (or their reverse complementary counterparts) as substrings. The criterion of an evaluation of the solution can be for example minimizing its length or maximizing its likelihood.

In the assembling step the input sequences are outcomes of the DNA sequencing process. Unfortunately, the input sequences are charged with errors coming from the biochemical experiment. The errors can be insertions (additional nucleotides), deletions (missing nucleotides) and substitutions (changed nucleotides). Thus, inexact matches of sequences have to be allowed. Of course, to avoid accident overlaps of sequences an error bound of mismatches have to be defined. Another type of problems in the assembling is that the sequences may come from both strands of a DNA helix fragment. In this case some of input sequences have the opposite orientation than the others and they should be matched obeying the rule of complementarity. Some methods deal with this problem by addition of reverse complementary sequences to the input ones. In the reconstruction phase only one of the two sequences can be used: the original or the reverse complementary one. The last phase of the assembling, when the order of input fragments is already known, consists of creation of a *consensus* sequence from the multiple alignment of the input sequences. Each character in a column places a vote for each letter in its set, where a null character is equivalent to the empty set. For each letter, the votes are gathered. The consensus character comes from the set of letters that receive the most number of votes. Example 2.2 shows an illustration of the sequence assembling.

Example 2.2: *Let the set of input sequences of the assembling process be* $\{ACCT,$ $ACTC,$ $AGGTG,$ $CAGCCT,$ $GTTCG\}$. *Sequences may come from two strands of the DNA helix, so we must consider as the possible input also all the reverse complementary counterparts of the sequences. In the reconstruction phase we must use exactly one of the sequences from the pair. Now the input set can be viewed as composed of the fragments:* $\{\{ACCT,$ $AGGT\},$ $\{ACTC,$ $GAGT\},$ $\{AGGTG,$ $CACCT\},$ $\{CAGCCT,$ $AGGCTG\},$ $\{GTTCG,$ $CGAAC\}\}$. *We assume that in these sequences errors (like insertions, deletions, and substitutions) may appear, thus, we allow for inexact matching. A possible result is shown in Figure 2.8. The resulting sequence (*the consensus sequence*) is created according to the majority rule: on every position the sign which appears the most frequently is chosen.* □

Sometimes in recognizing the genetic information another step is required (*mapping*) which places the contigs into proper place on a chromosome.

Fig. 2.8: A possible assembly of sequences in the Example 2.2



2.3 Theoretical formulations of DNA sequencing and assembling problems

Following introduced biochemical concepts and less formal descriptions, we may now formulate basic DNA sequencing and assembling problems in a more exact, combinatorially oriented manner. We will formulate all the sequencing problems without errors in the spectrum in the search version and the the other problems in their optimization versions, since the aim of the thesis is a construction of efficient algorithms for the practical problems, complexity of which being in most cases already established. We start with the problem of DNA sequencing by hybridization with standard libraries [BK03]. We distinguish the case of error-free spectra (ideal case) and of those containing errors.

Definition 2.1: *Standard SBH problem without errors in the spectrum – search version.*

Parameters: Set S of oligonucleotides of equal length l, length n of an original sequence, where $|S| = n - l + 1$ and S is the ideal spectrum for the sequence.

Answer: A sequence of length n composed of all elements of S.

Definition 2.2: *Standard SBH problem with positive and negative errors – optimization version.*

Parameters: Set S of oligonucleotides of equal length l, and length n of an original sequence.

Answer: A sequence of the length $\leqslant n$ containing the maximum number of elements of S.

Next, two definitions are presented of isothermic DNA sequencing by hybridization problem in the case without errors in the spectrum and in the case with errors [BFKM04].

Definition 2.3: *Isothermic SBH problem without errors in the spectrum – search version.*

Parameters: Set S of oligonucleotides, each of them of temperature t or $t+2$, length n of an original sequence, where S is an ideal spectrum for this sequence.

Answer: A sequence of length n such that an ideal spectrum for this sequence is equal to S.

Definition 2.4: *Isothermic SBH problem with positive and negative errors – optimization version.*

Parameters: Set S of oligonucleotides, each of them of temperature t or $t+2$, length n of an original sequence.

Answer: A sequence of the length not greater than n with a minimum value of $\beta + |S| - 2\alpha$.

α and β denote, respectively, a number of oligonucleotides from the spectrum being the part of the constructed sequence, and a number of oligonucleotides being members of the two used isothermic libraries of temperatures t and $t+2$ which can be distinguished in this sequence (each oligonucleotide adds to β number of its occurrences in the sequence). The number of positive and negative errors in the constructed sequence can be defined as, respectively, $|S| - \alpha$ and $\beta - \alpha$.

And last but not least there are two definitions of DNA sequence assembling problem; the first one in the case without errors in the input sequences. This case is not considered in this work in more details, since it is only a theoretical model. In general the input sequences contain errors: insertions, deletions and mismatches. The fragments may also come from different strands of a DNA helix fragment. The problem becomes more difficult because imperfect matches have to be allowed in aligning two fragments. The problem where the errors appear in the input fragments is defined in the last definition.

Definition 2.5: *DNA sequence assembling problem without errors – optimization version.*

Parameters: Multiset S of fragments over the alphabet {A, C, G, T} coming from the same strand of a DNA helix.

Answer: A sequence of the minimum length, containing all the fragments from the multiset as the subsequences.

Definition 2.6: *DNA sequence assembling problem with errors – optimization version.*

Parameters: Multiset S of fragments over the alphabet {A, C, G, T} which may come from both strands of a DNA helix. All the reverse and complementary fragments to the original ones in the multiset are added into S .

Answer: A sequence of a maximum likelihood value, containing from every pair of fragments in the multiset (the original fragment or its reverse complementary one) only one fragment as the subsequence with mismatches allowed.

The above problems will be studied in the following chapters. Firstly, their complexity status will be mentioned together with the presentation of existing methods. Next, new algorithms will be proposed and compared with the existing ones.

3. PREVIOUS APPROACHES TO SEQUENCING

3.1 Small scale sequencing

In this chapter, several biochemical approaches and computational methods solving the second phase of the sequencing problem by hybridization, are presented. In Section 3.1.1 the standard approach is described in which equal-length oligonucleotides are used in the hybridization experiment. Next section discusses the case of oligonucleotides containing universal bases, i.e. the bases which bind to any nucleotide. This approach allows to increase the length of the examined sequence, but it is difficult to apply in a laboratory. Another approach presented in Section 3.1.3 minimizes the number of errors coming from the hybridization experiment by using oligonucleotides of equal melting temperature (isothermic libraries).

3.1.1 Standard sequencing by hybridization

The DNA sequencing by hybridization was first proposed in 1988 [Sou88] and since then many proposals of the solution to the computational part of this problem appeared in the literature. Here, the most important algorithms are cited (formal description of the basic problem was given in Definitions 2.1 and 2.2).

The first algorithm that reconstructs a sequence from the elements of the spectrum was presented in [BS88]. It assumes that all the oligonucleotides that can be distinguished in the examined sequence are in the spectrum, and some elements can appear more than once if an oligonucleotide is contained in the sequence more than once. Thus, the cardinality of the spectrum is always equal $n - l + 1$. The algorithm builds a tree in which nodes are the elements from the spectrum and the nodes can be connected by an arc if last $l - 1$ nucleotides from an oligonucleotide of the first

node overlap first $l-1$ nucleotides from the oligonucleotide of the other node. If an oligonucleotide is already included in the current path, then it cannot be visited once again. As the root of the tree the probe which begins the examined sequence is taken. If it is not known, then each of the oligonucleotides from the spectrum is taken as the root consecutively. A path which starts from the root and ends in a leaf, and contains all the elements from the spectrum is a solution. Sometimes it is possible to obtain more than one solution. Example 3.1 presents the trees which result in finding solutions.

In [LFK+88] Lysov et al. noticed that the SBH problem without errors (Definition 2.1) refers to a well known problem of searching for a Hamiltonian path in a particular graph. A directed graph H is constructed in the following way: nodes are the elements from the spectrum and an arc connects two nodes if $l-1$ last nucleotides from the first node overlap first $l-1$ nucleotides from the second one. In such a graph a Hamiltonian path is looked for (Example 3.1).

Example 3.1: *Suppose we have to reconstruct a sequence GTGTAGTC of length n=8. After simulating the ideal hybridization experiment with the library of oligonucleotides of length $l = 3$ we get the ideal spectrum, which is composed of $|S| = 6$ elements, $S = \{AGT, GTA, GTC, GTG, TAG, TGT\}$. The method by Bains and Smith [BS88] builds a tree, starting each time from a different node as the root. Only two trees which start with 'GTG' or 'GTA' finish with a path which crosses all the elements of the spectrum (see Figure 3.1). The algorithm results in two solutions: GTGTAGTC and GTAGTGTC.*
Considering the same sequence and the spectrum, the algorithm by Lysov et al. [LFK+88] constructs a directed graph (see Figure 3.2). In such a graph a Hamiltonian path is searched for. There exist two Hamiltonian paths and they result in two solutions: GTGTAGTC and GTAGTGTC.
A decision which sequence to choose can be done after an additional biochemical experiment which dispels ambiguity of the solution. □

The above methods need an ideal spectrum and their complexity is exponential in time. A polynomial-time algorithm for DNA sequencing problem by hybridization with an ideal spectrum was presented by Pevzner in [Pev89], what proved that the *error-free variant* of the problem (Definition 2.1) belongs to the *class of easily solvable*

Fig. 3.1: A tree constructed with the method by Bains and Smith [BS88]

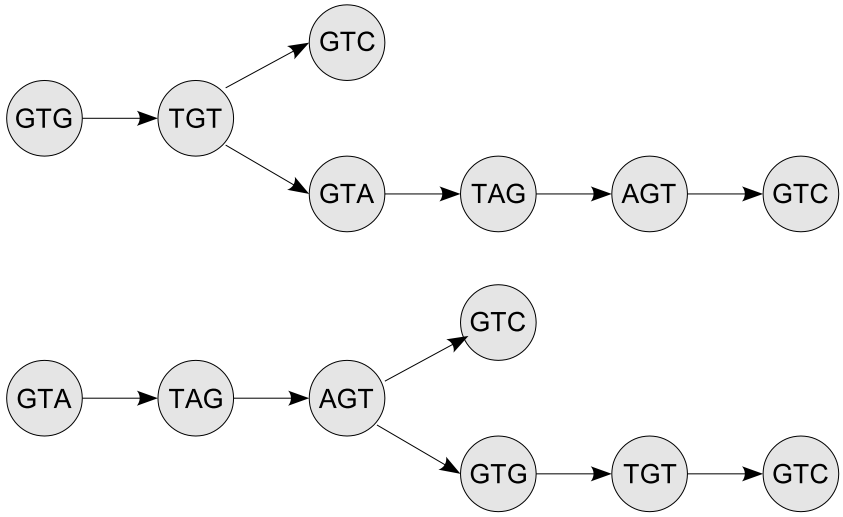
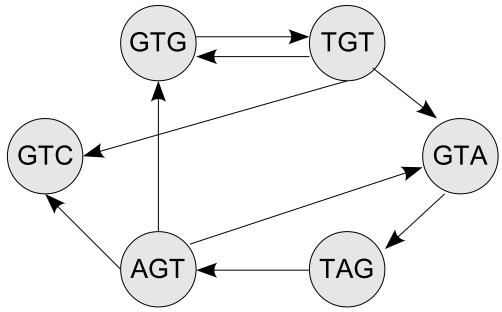


Fig. 3.2: A directed graph constructed with the method by Lysov et al. [LFK+88]



problems. The algorithm searches for an Eulerian path in a directed graph, but this time the elements of the spectrum are the labels on arcs, and each arc connects the nodes which are labeled with the $l-1$ prefix and $l-1$ suffix of the arc label. (Example 3.2)

The transformation of the graph where a Hamiltonian path is looked for to the graph, in which an Eulerian path is searched, changes the complexity of the algorithm. The class of these labeled graphs in which such a transformation is possible was widely discussed in [BHKW99]. The graphs which are built from the elements of the spectrum with the method by Lysov et al. belong to this class and are called *DNA graphs*. The graph constructed on the basis of the Lysov's method is a directed line-graph of the Pevzner's graph. For such pair of digraphs searching for a Hamiltonian path and an Eulerian path, respectively, is equivalent.

Another algorithm presented in [Pev89] allows for negative errors in the spectrum (Definition 2.1 with an additional information that only negative errors may appear). Not in every case it finds a solution, so it can be treated only as a heuristic. In this method the number of missing elements in the spectrum is first determined, which is equal to $n-l+1-|S|$, and next these oligonucleotides are to be recovered. For this purpose the problem is transformed to the one of searching for a flow in a network based on the bipartite graph $K_{m,m}$. The bipartite graph is constructed from the vertices of the Pevzner's graph which have different in- and out-degrees. A vertex for which this difference is greater than 1 is multiplied in the bipartite graph. Arcs in the bipartite graph are drawn from the vertices which have greater indegree to the ones having greater outdegree in the Pevzner's graph. The cost for an arc is equal to the minimal shift of the labels of its vertices in their overlap. Therefore, the greatest value of the overlap between labels is equal $(l-2)$ and the cost is equal 1, while the greatest cost is equal $(l-1)$ when labels don't overlap at all. Each arc in the bipartite graph corresponds to a path in the Pevzner's graph of length equal to the cost of this arc. To the bipartite graph two vertices are added: the source s, which is connected with the vertices of a greater indegree and the sink t which is connected with the other vertices. All arcs in this network have the capacity equal 1. In such a network we look for a flow of value $m-1$ with a minimum cost. If the cost is equal to the number of missing elements from the spectrum, then to the Pevzner's graph the missing arcs (or paths) can be added. Now we are looking for an Eulerian path in the newly developed graph (see Example 3.2).

Example 3.2: *Suppose once more that we have the same sequence and the spectrum as in Example 3.1. The graph constructed with Pevzner's method [Pev89] has the labels of the oligonucleotides from the spectrum on the arcs (see Figure 3.3). There are two Eulerian paths and reading the labels from the arcs we obtain two possible solutions: GTGTAGTC, GTAGTGTC.*

Let us now consider the case where during the hybridization experiment an error occurred or the fluorescent chip image is unclear and one of the oligonucleotides is missing in the spectrum: TGT. We have to count how many nucleotides are missing: $n-l+1-|S|=1$. *Next the network is constructed from the vertices with different in- and out-degrees based on graph from Fig. 3.3, from which an arc 'TGT' was deleted.*

Notice that in the network (Figure 3.4) there are two vertices labeled 'GT', because

Fig. 3.3: A digraph constructed with the method by Pevzner [Pev89]

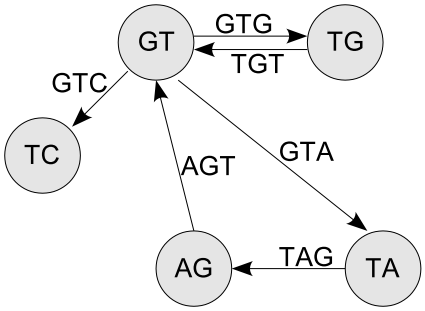
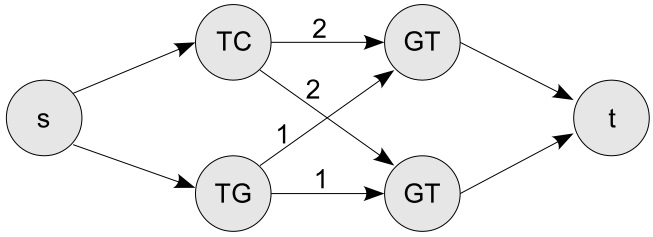


Fig. 3.4: A network for finding missing oligonucleotides in the spectrum [Pev89]



the difference between the in-degree and the out-degree of this vertex is equal two. The values on arcs in between the vertices are equal to shifts of vertex labels in their maximum possible overlaps. Now we are searching for a flow of value $(m-1=)$ *1 with its cost equal to* $n-l+1-|S|=1$. *There exist two flows with the cost equal to 1, both between the vertices: (TG,GT) and they are equal to our missing oligonucleotide 'TGT'. After adding this arc to the base graph we can look for an Eulerian path and find the sequence. Notice that if the missing oligonucleotide is 'GTA' then it will not be possible to find missing arc in the network, because the vertex with label 'GT' will not be considered in the network, due to equal in- and out-degrees. Thus, reconstructing the original sequence won't be possible.* □

The first exact algorithm that deals with both negative and positive errors in the spectrum and has no additional information about the elements from the spectrum was introduced in [BFK+99]. The sequencing problem was formulated as a variant of the traveling salesman problem, called Selective Traveling Salesman Problem. Similarly as in [LFK+88], the vertices in the graph are labeled with the oligonucleotides. The graph is complete symmetric and to every arc a cost is assigned equal to the maximal shift between two labels of the adjacent vertices. With every vertex the

profit equal to 1 is connected. In such graph a simple path with a maximum profit is looked for and with a cost not greater than $n - l$. Example 3.3 illustrates this method.

Example 3.3: *Let the sequence to be reconstructed be the same as in Example 3.1: GTGTAGTC. Into the spectrum some errors have been introduced: negative one – AGT, and positive one – CAT, and the spectrum is now composed of 6 elements:* $S = \{CAT, GTA, GTC, GTG, TAG, TGT\}$. *First, costs of arcs between each pair of elements are determined (see Table 3.1).*

Tab. 3.1: Costs of arcs between vertices in the graph from Example 3.3

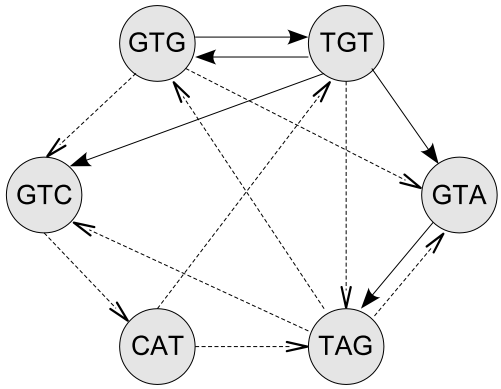
	CAT	GTA	GTC	GTG	TAG	TGT
CAT	-	3	3	3	2	2
GTA	3	-	3	3	1	3
GTC	2	3	-	3	3	3
GTG	3	2	2	-	3	1
TAG	3	2	2	2	-	3
TGT	3	1	1	1	2	-

Next, a complete graph is constructed, see Figure 3.5, where the profit for each vertex is equal to 1. The method by Blazewicz et al. [BFK+99] searches for a simple path of a maximum profit and a total cost not greater than $n - l = 5$, *and results in two paths which consist of 5 elements and are of cost 5. Reading the labels we get the solutions: GTGTAGTC and GTAGTGTC.* □

The algorithm works in exponential time. The complexity of the SBH problem with errors (Definition 2.2) was analyzed in [BK03] and the authors proved that even in the case of errors of one type (only positive or only negative ones), the problem is *strongly NP-hard*. Therefore, an exact algorithm, like branch and bound proposed in [BFK+99], may not finish in reasonable time for large instances. Thus, almost all of the algorithms dedicated to this problem are heuristics, see for example methods proposed in [BKK02, ZWZ03, BGK04, End04].

In [Kru98] a multistage variant of the SBH method was proposed. A series of experiments are executed with oligonucleotide libraries containing molecules of in-

Fig. 3.5: The graph from Example 3.3 constructed with the method by Blazewicz et al. [BFK+99]. The arcs with the cost equal to 1 are denoted with the solid arrow, with the cost 2 are denoted with the dashed arrow, and the ones with the cost equal to 3 are not drawn (but present in the graph) in order to keep the picture clear.



creasing lengths. The first experiment is performed with the classical DNA chip containing a full library of oligonucleotides of length l. In the next experiments the goal is to detect all the oligonucleotides of length $2l$ which are contained in the examined sequence. Thus, the library of oligonucleotides on the chip consists of all concatenated pairs of elements of length l detected in the previous experiment. The author noticed that every substring of length $2l - k$ is composed of two oligonucleotides which overlap on k positions. Thus, instead of using the library composed of all concatenated pairs of oligonucleotides (of length $2l$), the library that contains the elements of length $2l - k$ is used (these new elements are the concatenations of two oligoncuelotides, overlapped on k positions). This reduces the size of the library for the next experiments by restricting the possible merging of two oligonucleotides only to the probable pairs. Every repetitive substring in the target sequence will be detected in the experiments with the oligonucleotides of increased length. However, this method is vulnerable to errors coming from the experiments. Every negative error is propagated through the next iterations. And every positive error may increase the cost of the method by increasing the size of the library for further experiments.

In [PS01] an *interactive method for sequencing by hybridization*, on the basis of a spectrum in which both positive and negative errors appear, was proposed. The traditional hybridization experiment is enriched there with a series of comparisons of

the original sequence to the elements of increasing lengths in order to resolve all the ambiguities. The method constructs a graph, the same as presented in [LFK+88], where oligonucleotides are the labels of the vertices and the arcs connect two vertices if the labels overlap with shift equal to 1. Series of additional queries (being additional biochemical experiments) about oligonucleotides of increasing lengths, starting from l, help to get rid of useless vertices, which are positive errors, and to add additional vertices which are the missing oligonucleotides in the spectrum. When all the branching points in the graph are resolved, the algorithm stops querying and returns as a solution the only Hamiltonian path in this graph.

A simple genetic algorithm for SBH with any errors was proposed in [BKK02]. The algorithm starts with an initial population of s individuals. Each individual is a permutation of all elements from the spectrum. In each individual the best substring of oligonucleotides is identified, i.e. the one that is composed of the greatest number of elements from the spectrum and is not longer than n. Next, to each individual a normalized fitness value is assigned, which is the ratio of the number of elements in the best substring to the maximum number of elements from the spectrum in any valid sequence, being $n - l + 1$. The individuals are then selected as parents with stochastic remainder method without replacement according to their fitness values. Next population is formed from these individuals, randomly paired, to which a greedy crossover is applied. The crossover starts the construction of an individual with the first oligonucleotide chosen randomly. With 20% of probability the oligonucleotide that best overlaps the last oligonucleotide in the individual built so far, is added. Moreover, with 80% of probability the best of the succesors of the last oligonucleotide in parents of the individual is selected, providing it has not been already used in the construction. Otherwise, a random oligonucleotide from the remaining ones is chosen. The crossover stops when all elements are in the individual. These moves are done until all the individuals in the population are constructed. Every new population is submitted to the above series of transformations and the steps are repeated until a given number of iterations without improvement of the best fitness value is met. The best solution found is given as a result.

In [BY04] another genetic algorithm is presented. The idea of the method is as follows. During a preprocessing step the oligonucleotides are joined into longer fragments in which two adjacent elements overlap on $l-1$ positions. If there is more than

one possible extension of the oligonucleotide, then such oligonucleotides cannot be joined together. The algorithm next operates on the longer fragments. The solution is represented as a permutation of the fragments. The number of the individuals in the population remains constant throughout the algorithm. The fitness of each individual is calculated on the basis of the number of overlapped positions between two adjacent fragments and the length of the sequence. Parents are selected using the proportional selection method, and the individuals with the higher fitness are more likely selected. Parents are next reproduced with crossover and mutation operators. In the crossover 3 cutpoints are determined, and each offspring chooses different parts from parents. This process may create offspring that contains duplicated fragments. Thus, it is necessary to employ a repair mechanism which throws out duplicated fragments, and adds omitted ones. Additionally, the repair mechanism improves an individual with a local search. On the newly created offspring the mutation is performed and with 10% probability randomly selected oligonucleotides are swapped. An offspring replaces the worst parent in the population. The algorithm stops when there is no improvement of the solution. As a result the best solution found, is given.

In [ZWZ03] the algorithm assumes the knowledge about consecutive missing oligonucleotides in the sequence. The authors first transform the problem with both types of errors to a problem in which only positive errors appear. They assume that the error level, which is the number of consecutively missing oligonucleotides, is equal to δ and $1 \leqslant \delta < l$. The assumption comes from the observation that for $\delta \geqslant l$ it is impossible to reconstruct the original sequence from the spectrum elements. For each pair of elements (u, v) from the spectrum S for which the shift between u and v is t, such that $1 < t \leqslant \delta$, a path of length t is determined. All the elements from the path not being in the spectrum are *artificial elements* and are added to the extended spectrum S'. If δ is too large it will result in adding too many elements to the extended spectrum. The authors assume that $\delta \leqslant 3$. After this preprocessing step in the extended spectrum all oligonucleotides which can be distinguished in the examined sequence, are present. Next, the algorithm for positive errors only is applied to the extended spectrum. It starts from the adjacency matrix A of size $|S'| \times |S'|$ with the entries '1' if the elements overlap on $l-1$ positions and '0' otherwise. Next matrices A^k are calculated in which the values $a^k_{i,j}$ are the numbers of all k-paths (paths of length k) from probe s_i to probe s_j. The goal of the algorithm is to maximize k, i.e.

to maximize the number of elements in the solution. The examined sequence cannot be longer than n, thus, $k \leqslant n - l$. Notice that some elements can be used more than once in one path, what solves the problem of repeated fragments in the DNA sequence. As a result the method gives all possible paths of maximum value of k.

A tabu and scatter search method was presented in [BGK04]. The main idea of the algorithm is to optimize the maximum number of elements in the solution with the local search heuristic – tabu search – and next in order to improve the results and to diversify searching in the solution space, the adaptive memory strategy was applied in the restart process for forming a new initial solution. The tabu search algorithm works as follows: the initial solution is created with a greedy heuristic, and is composed of the oligonucleotides from the spectrum which create the sequence of nucleotides not longer than n. The rest of the oligonucleotides which are not in the solution are in a set, called trash. Next, a defined number of cycles is performed and each one consists of a few condensing and extending moves. The goal of the condensing moves is to *condense* the oligonucleotides that composes the solution tending to their maximum overlap. The goal of the extending moves is to insert into the solution these oligonucleotides which have been rarely used in solutions constructed till now, or delete frequently used ones. The possible condensing moves are: inserting a probe from the trash into the solution, shifting a probe in the solution or deleting a probe from the solution to the trash set, providing the solution is feasible. Inserted and shifted oligonucleotides are remembered on the tabu list, and are prohibited for a couple of moves. A collection of the best solutions found during executing the cycles is stored and later on, during the restart process, it is used to build a new starting solution. After a few restarts of the local search procedure the algorithm gives as a result the best solution ever found.

3.1.2 Universal bases

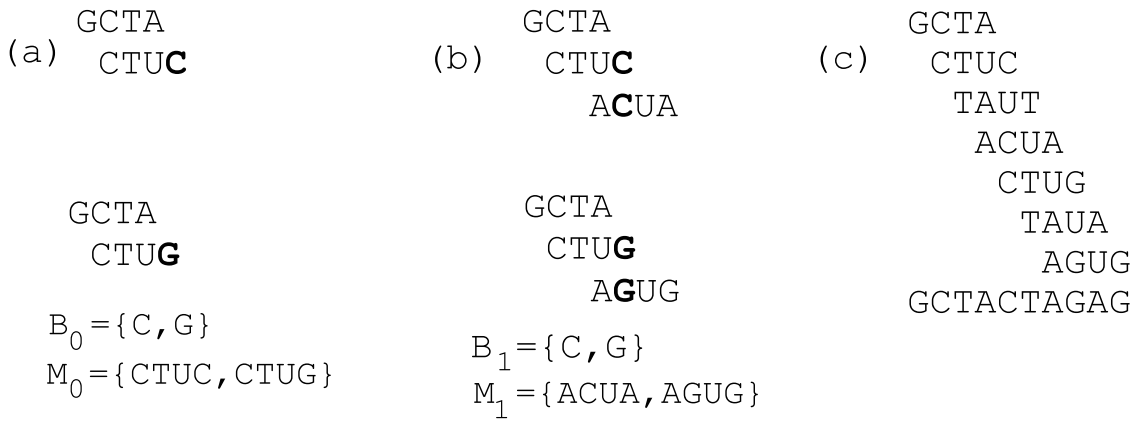
Sometimes it is impossible to reconstruct a target sequence from the library of equal length oligonucleotides due to errors in a spectrum and ambiguous reconstruction. To omit some difficulties a new chip design was presented by Preparata et al. [PFU99, PU01]. The chip consists of longer elements than in the standard chip design and in the same time the size of the library of all the elements is the same. The elements design is based on universal bases which are chemical entities and when present in

the element will engage in the base-pairing relationship with any of the standard nucleotides (A, C, G, or T). A universal element can be for example: 5-nitroindole [LB94] or 3-nitropyrrole [BANZ95].

Proposed special pattern of elements is called (s, r) *gapped probe* and is denoted by $GP(s, r)$. For fixed parameters s and r the element is of form $X^s(U^{s-1}X)^r$, where X is one of the standard bases (A, C, G, or T) and U is the universal base. For example, the pattern GP(3,2) is of the form: XXXUUXUUX. A simple algorithm presented in [PFU99] is a heuristic, because not in every case it finds a target sequence. Despite of it, it is vulnerable to errors. The procedure starts with a known prefix of length $s(r + 1)$. With every iteration it tries to increase by 1 the length of a hypothetical sequence. To do it, the method searches for elements which $(s(r + 1) - 1)$ prefix matches the suffix of the hypothetical sequence. The set of these elements is called M_0. If there is only one extension of the hypothetical sequence, then the symbol is appended to the sequence. In other case all possible extensions are put into set B_0. It must be decided which of these symbols are correct. Afterwards, set M_1 is constructed which consists of all the elements shifted by s to the elements in M_0, which at the same time match the hypothetical sequence. In set B_1 there are symbols which are possible extensions of the hypothetical sequence caused by elements from M_1. If $B_0 \cap B_1$ is a single symbol, then the hypothetical sequence is extended with it. Otherwise, in the similar way sets M_i and B_i are determined for all $i \leqslant r$. If there exists such i that intersection of sets B_0 to B_i contains only one symbol, then there exists an unambiguous extension. Otherwise the algorithm stops and returns the current sequence. The process of the reconstruction of a sequence with this algorithm is presented in Example 3.4.

Example 3.4: *Suppose we have a sequence GCTACTAGAG. The set of gapped (2,1) - probes detected during the hybridization without errors is* $S = \{ACUA, AGUG, CTUC, CTUG, GCUA, TAUA, TAUT\}$. *The prefix is assumed to be known, and it is GCTA. In the first step we look for all the elements that match the hypothetical sequence, which is the known prefix. Thus, in the set* M_0 *we get* $\{CTUC, CTUG\}$ *and possible extensions of the sequence are* B_0=$\{C, G\}$ *(see Figure 3.6 (a)). Next, we search for elements shifted by* $s = 2$ *nucleotides to the elements in* M_0 *and we get the set* $M_1 = \{ACUA, AGUG\}$ *and possible extensions to the sequence:* B_1=$\{C, G\}$ *(see Figure 3.6 (b)). There are two possible extensions* $B_0 \cap B_1 = \{C, G\}$ *and*

Fig. 3.6: An example of a reconstruction of a sequence from the spectrum composed of gapped elements.



the algorithm stops here ($i = r$). More sophisticated algorithm would develop all the possible extensions and find the sequence as presented in Figure 3.6 (c). □

Later on the reconstruction of a sequence using tandem spectra was proposed, which combines direct and reverse spectra [HPY03]. The direct spectra are of the form as used in [PFU99]: $X^s(U^{s-1}X)^r$, while the reverse spectra are opposite to them: $(XU^{s-1})^rX^s$. The usage of both spectra should result in better performance because when some elements in the spectrum are erroneous, the extensions of the sequence may be supported by elements from the reverse spectrum.

Another algorithm was presented in [HHHS03]. The authors proposed an alternative chip design which instead of a deterministic periodic structure uses randomized elements in which locations of some specified bases were chosen randomly. The elements are prepared in the following way. The length of each element is $l = c * k + 1$, where c is a sufficiently large integer and k is the number of known nucleotides (A, C, G, T). Next the subsets A_i are formed, where $i = 1...\beta k$ and β is a constant that depends on error rate. For each subset a set of random k positions are chosen out of $\{1,2,...,ck\}$ and form all possible 4^{k+1} probes, with known nucleotides on the chosen positions. The total number of elements used in the experiment is equal $\beta k 4^{k+1}$, and it is a union of A_i sets. The sequence that can be unambiguously reconstructed with high probability may be of length $\alpha\frac{4^k}{\beta k}$ where α is a constant that depends on

the error rate. The algorithm presented in [HHHS03] is noise resistant, what means that even despite of positive and negative errors in the spectrum the sequence can be usually unambiguously reconstructed.

Although universal bases have been successfully generated in a laboratory [LB94, BANZ95], it is still unclear how they hybridize in longer oligonucleotides and in the case there are many universal bases. The presently available products do not seem to be sufficiently well behaving for de novo sequencing.

In [PO04] the authors analyze the problem of DNA sequencing using universal bases once again, but this time instead of universal bases, which are ideal but not practically reachable, they use their substitute, degenerate bases, which are uniform mixtures of the four natural bases. The number of degenerate bases in the element is limited since each degenerate base causes a reduction of the number of hybridizing DNA strands to the element on the microarray chip by the factor of 4 and it results in deterioration of the hybridization signal. In the model of the standard hybridization it is assumed that all Watson-Crick complementary pairs have identical hybridization strength. It is well known (it was already mentioned in Section 2.2) that the hybridization strength depends not only on the G/C or A/T content in oligonucleotide duplexes ([WJH$^+$81]), but also on the position of a particular base in the oligonucleotide duplex (nearest neighbor model [BFBM86]). Thus, the authors deduce that the position of a degenerate base has an effect on the hybridization signal. Following this, the binding-energy detection threshold is a set of the lowest values compatible with the selection for the bases occurring in the degenerate positions. Unfortunately, the binding energy values associated with a degenerate base position in the probing patterns are spread, what makes degenerate bases non-universal. The degenerate bases are replaced with semi-degenerate bases, which are mixtures involving two subsets separately: A/T (weak bases) and G/C (strong bases). At that time the binding energy values are in a more narrow range.

3.1.3 Isothermic sequencing by hybridization

In the computational part of the isothermic sequencing problem by hybridization, as the input data we get spectrum S composed of elements of two isothermic libraries differing by two degrees, and the length of the examined sequence n (cf. Sections 2.2 and 2.3). The objective in case of errors in the spectrum is to find a sequence that

is composed of the oligonucleotides from the spectrum with the minimum number of positive and negative errors. This solution is mostly equivalent to a solution obtained when maximizing the number of oligonucleotides in the solution. Note, that for equal-length libraries in the standard SBH both criteria are equivalent. For the isothermic sequencing in some cases maximizing the number of oligonucleotides in the solution would prefer different sequence than the one looked for.

The complexity of the problem was widely examined in [BK05]. It was proven that in case of positive or negative errors only, or errors of both types in the spectrum, the problem is strongly NP-hard. However, if the spectrum is ideal the isothermic SBH problem is solvable in polynomial time. The corresponding algorithm for the sequencing without errors is as follows [BK05]:

A directed graph G is constructed on the basis of the spectrum, and after some transformations the algorithm searches for a path corresponding to a DNA sequence. The first and the last oligonucleotide are assumed to be known. If there is lack of knowledge about the first and the last oligonucleotide, all the possible pairs of the oligonucleotides are taken respectively. In graph G oligonucleotides are on the vertices and arcs connect two vertices if their oligonucleotides are of equal length and they overlap with the shift by one letter, providing it does not produce a negative error. If one oligonucleotide o_i is contained in the other one o_j and is shifted to the left, then all the arcs entering o_j and leaving o_i are deleted and the arc from o_i to o_j is added. On the other hand, if o_i is contained in o_j, but shifted to the right, then the arcs entering o_i and leaving o_j are deleted and the arc from o_j to o_i is added. Every arc entering the vertex corresponding to the first oligonucleotide or leaving the vertex which corresponds to the last oligonucleotide are deleted. Next, some deletions of arcs are made and some temporary arcs are added to G in order to make graph G a line graph. Graph G is then transformed to its original graph H, and now the oligonucleotides correspond to the arcs in graph H. The algorithm searches for an Eulerian path in the directed graph H, provided that the transitions corresponding to temporary arcs in G are forbidden.

In [BF05] another method to isothermic SBH problem was proposed, which combines multistage approach with isothermic libraries. (Multistage variant of SBH method was first proposed in [Kru98] and it was applied to standard libraries.) This approach decreases the number of experimental errors, especially the negative ones.

In the first experiment two isothermic libraries are used, of temperatures respectively t and $t+2$. The library for the next experiment is composed of all pairs of concatenated oligonucleotides coming from the previous experiment, for which the overlap fragment is of temperature k or $k+2$. As a result from the hybridization experiment a spectrum is obtained, and it is composed of oligonucleotides of temperatures $2t-k$ and $2t-k+2$. If it is necessary, the temperature for the experiment is further increased. Otherwise, the problem is solved with an algorithm for the isothermic SBH, for example a tabu search algorithm presented in [BFS+04].

3.2 Assembling – large scale sequencing

Assembling, also called *large scale sequencing*, is the next step of recognizing genetic information. In the classical approach, sequences of length up to 700 nucleotides obtained in the sequencing step are merged together into longer contigs or even a whole genome (for example a bacterial of viral genome) – cf. Definitions 2.5 and 2.6. Sometimes another step is required (mapping) which places the contigs into the proper places on a chromosome.

In the novel approach (454 sequencing – cf. Section 2.2), the sequencing step is done by a machine and we can receive sequences of length around 100 nucleotides. With these sequences an additional information is associated about the quality of every nucleotide in the sequence. Every position in the result contig is covered here by more sequences than in the classical approach. However, this approach is less robust with respect to repetitions in the genome. If the repeated part is longer than 100 nucleotides, then the fragments will not cover the whole ambiguities and the sequence cannot be fully reconstructed.

The algorithms solving the assembling problem are usually composed of three parts. First part consists in aligning input fragments. Due to errors coming from the experiment the alignment algorithm must allow for inexact matchings (some deletions, insertions or substitutions are possible). In the second part, the fragments are joined together into a longer sequence, of possibly the shortest length. And finally, a consensus sequence is determined, by voting of the fragments on every place in the sequence (see Example 2.2 in Section 2.2). Although there exist many algorithms solving the assembling problem, some of them are parts of commercial packages and

the details about the algorithms are not published. Here, a few of the most known algorithms are presented. In general they are designed for the classical large scale sequencing, but the last method is dedicated to the assembling of shorter fragments.

The algorithm from [KM95] assembles sequences with errors (insertions, deletions and substitutions of nucleotides) coming from both strands of a DNA fragment. At the beginning the reverse complementary sequences to every fragments in the input set are added. Next, all the pairs of the sequences are compared to determine their approximate overlaps of some statistical significance. The complete graph is created in which vertices are labeled with these sequences and arcs correspond to their overlaps. Some of the arcs are culled according to the minimum match significance threshold, thus the graph is more sparse. The orientation of the fragments is determined by a heuristic algorithm finding a maximum weight spanning tree in the graph (only the half of the vertices are considered corresponding either to the original fragments or to the reverse complementary ones). The solution to the problem is a Hamiltonian path with the maximum total significance score.

The method proposed in [IW95] applies the Pevzner's algorithm for the standard sequencing by hybridization [Pev89]. Every input sequence is represented as a collection of $n - l + 1$ oligonucleotides of length l composing the sequence, where n is the length of the sequence. Next, the Pevzner's graph is constructed from these collections and the Eulerian path is looked for. A big value of l is preferred in order to retain the information about the input fragments. This method works properly only for error-less instances (Definition 2.5), however, the authors show how to adapt their algorithm for the case of errors coming from the sequencing phase and from fragment orientation (Definition 2.6). To the input set the reverse and complementary fragments are added and at the end two complementary sequences are produced. In case of repetitions or errors in the input sequences, the algorithm allows to use some arcs more than once (repetitions) or to not visit them at all (errors in the sequences). Therefore, a variation of an Eulerian tour is applied in a similar way as in the Selective Traveling Salesman problem.

In the method presented in [JL96], the authors model the sequence assembling problem as a variant of the shortest common superstring problem. Given the set of fragments, the goal is to find the shortest sequence having all the fragments as substrings. In this problem the fragments are coming only from one strand, and

may contain at most k errors each. The algorithm merges two strings into a longer contig in every iteration. The pair of elements is chosen for which the ratio between the length of the contig and the sum of lengths of the fragments in this contig is the smallest. The method allows at most k errors in aligning the strings. In the first phase of the algorithm the fragments are merged into small groups of a few fragments tightly overlapping each other. The output of the last phase is one string composed of all the fragments.

In [PTW01] the authors presented another method based on the Pevzner's approach of finding an Eulerian path in a graph [Pev89]. The method can deal with sequencing errors in the input fragments as well as with long repeats. The first step of the algorithm consists in elimination of errors by some modification of the input sequences. The input fragments are decomposed to the oligonucleotides of length l (the set of them all is called spectrum), where l is much smaller than the length of the fragments. The multiplicity of appearance of each oligonucleotide in the spectrum is calculated and if the number for an oligonucleotide is higher than M (which is a threshold) then it is marked as *solid*, otherwise it is marked *weak*. If there is a mutation in the fragment then a series of oligonucleotides is weak. An error correction procedure transforms weak oligonucleotides into solid ones and reduces at the same time the number of possible oligonucleotides in the spectrum. Next, a graph is created out of oligonucleotides from the spectrum on the base of Pevzner's approach. For each fragment the path representing it in the graph is remembered. At the end, the Eulerian path is searched for having all the remembered paths as the subpaths.

The method presented in [BKJ$^+$04] can deal both with errors coming from the sequencing step and with fragments coming from both strands of a DNA helix. At the beginning, to the set of fragments the reverse complementary sequences are added. The heuristic searches for pairs of fragments which have common parts and may overlap each other. And next, for every such pair, their shift and the corresponding error rate are calculated. The graph is constructed, where vertices are labeled with the fragments and two fragments are connected if the shift was determined and the error rate does not exceed a feasible value (bound). To make the graph more sparse some of the arcs are deleted on the basis of the following rule. The arc (i, j) with the corresponding shift d_{ij} is deleted if there exists vertex k for which arcs (i, k) and (k, j) with the shifts, respectively d_{ik} and d_{kj}, summing up to d_{ij}, may be found. The

deleted arc (i, j) increases weights of arcs (i, k) and (k, j). At the end, the path is searched for, which has maximum weights and includes half of the vertices (for every pair: a fragment and its complementary counterpart, only one can be included in a path).

In [CPT04] the method for sequencing with short reads is presented. The input fragments are of length 100-200 bp and may contain sequencing errors. The algorithm is based on the algorithm presented in [PTW01]. In the first phase of the algorithm the sequencing errors are corrected. The input fragments are divided into oligonucleotides of length l (15-20), which are marked as solid if they appear more than M times or as weak otherwise. To resolve the sequencing errors the dynamic programming algorithm was proposed. With increase of the length of the input fragments, the expected number of errors also increases, and the size of the search space becomes prohibitive. After the error correction phase, the graph is constructed from the oligonucleotides and the Eulerian path is searched for.

3.3 Summary of the existing approaches

From the analysis of this chapter it follows that there is no universal approach to sequencing and assembling stages of the genome recognition problem. Perhaps gel electrophoresis based approach is the most widely used for the moment in biochemical laboratories. In the future, most probably the pyrosequencing method (454 sequencing) will take the lead among sequencing approaches. However, for the moment it has some drawbacks, e.g. it is expensive and reads only short fragments, what makes the process of genome recognition much more complicated. Thus, it seems that the DNA sequencing by hybridization (SBH) approach is a good alternative for other sequencing methods in both its versions: with standard or isothermic oligonucleotide libraries. What is more, good algorithms for the computational phase of SBH may be a perfect starting point for design of the assembling algorithms, especially for the short reads coming from the novel pyrosequencing approach.

Thus, one may formulate the following goals for the research described in the thesis:

Design novel competitive algorithms for the DNA sequencing by hybridization, prove their advantage over existing procedures, and use the elaborated ideas to con-

struct a novel algorithm for the assembling stage.

The realization of the above goals is described in the next chapters of the thesis.

4. NEW ALGORITHMS FOR DNA SEQUENCING BY HYBRIDIZATION WITH ISOTHERMIC LIBRARIES

As it was mentioned earlier the goal of this work is to construct new, efficient and robust methods for DNA sequencing. In particular, the goal is to develop new algorithms for sequencing by hybridization with newly proposed isothermic libraries. Let us recall (Definition 2.4) that in its mathematical formulation the problem consists in finding a sequence of the length not greater than n with a minimum value of $\beta + |S| - 2\alpha$, given set S of oligonucleotides (spectrum), each of them of temperature t or $t + 2$, and length n of an original sequence. In this formulation α and β denote, respectively, the number of oligonucleotides from the spectrum being parts of the constructed sequence, and the number of oligonucleotides being members of the two used isothermic libraries of temperatures t and $t + 2$ which can be distinguished in this sequence (each oligonucleotide adds to β a number of its occurrences in the sequence). The number of positive and negative errors in the constructed sequence can be defined as, respectively, $|S| - \alpha$ and $\beta - \alpha$.

Determining the number of errors for every intermediate solution is very time consuming, and since all the proposed hereafter algorithms are heuristics, and can accept a little inaccuracy, this criterion was substituted with the same criterion as for the standard sequencing – maximizing the number of oligonucleotides from the spectrum in constructing the solution.

In this chapter two new algorithms devoted to isothermic sequencing problem by hybridization are described: tabu search (Section 4.1) and hybrid genetic algorithm (Section 4.2). Next the proposition comes of applying some features of the genetic algorithm for solving Traveling Salesman Problem (Section 4.3). In Section 4.4 the preliminary tests for tuning parameters of the algorithms are presented and in Section

4.5 preparation of spectra is described. In Section 4.6 both algorithms are compared together in extensive computational tests.

4.1 Tabu search algorithm

Tabu search for isothermic sequencing by hybridization is based on the algorithm which was presented in [Swi02, BFS$^+$04] and has the following structure. The spectrum is divided into two parts: an ordered collection of elements which is a *current solution* and an unordered set of remaining oligonucleotides, called *trash set*. At each stage of the computations the oligonucleotides from the solution produce a DNA sequence with length not greater than n. Only the moves that lead to *feasible* solutions, i.e. with the sequence not longer than n, are considered in this method. The algorithm starts with an initial solution constructed by a greedy heuristic. As the first oligonucleotide in the initial solution the first oligonucleotide from the spectrum is chosen and it is marked as used. Next, every combination of pairs of unused oligonucleotides from the spectrum is checked, and the first oligonucleotide from the pair with the best sum of overlaps is added at the end of the solution. This oligonucleotide is marked as used. The process of adding the oligonucleotides is repeated until the length of the sequence composed of the oligonucleotides is not longer than n. The process of constructing the initial solution is repeated for all oligonucleotides from the spectrum as the first oligonucleotide in the solution. After that the best solution, i.e. the one having the greatest number of oligonucleotides from the spectrum, is taken as the initial solution for the tabu search algorithm.

In the tabu search method three types of moves are initially defined: an *insertion* of an oligonucleotide from the trash set into the solution, a *shift* of an oligonucleotide from one position to another in the solution, and a *deletion* of an oligonucleotide from the solution to the trash set. The moves performed on single oligonucleotides are usually not sufficient to produce a solution of a good quality, thus operations on clusters have been also proposed. A *cluster* is a group of neighboring oligonucleotides in the solution linked together in such a way that the overlap p of every pair of neighbors is $p \geqslant l(o_r) - 1$, where $l(o_r)$ is the length of the right oligonucleotide in the pair. A list of clusters is updated after every move since moves like insertions, shifts or deletions can change clusters. Now, the set of moves is defined as: the insertion

of an oligonucleotide, the shift of an oligonucleotide or a cluster, the deletion of an oligonucleotide or a cluster. Note, that clusters exist only in the solution; after deleting a cluster all the oligonucleotides from the cluster are put to the trash set as separate elements.

Some constraints have been added to provide that oligonucleotides which fit together very well will not be easily separated:

- a cluster may be shifted only if it does not break any cluster;
- an oligonucleotide may be shifted if it is not a part of a cluster and it does not break any cluster;
- an oligonucleotide may be deleted if it is outside a cluster or on one of cluster ends.

The algorithm tends to maximize the number of elements from the spectrum in the solution. This criterion, called the *global criterion function*, leads to a reconstruction of a desired sequence, however, it cannot objectively evaluate all the moves, since deletions and shifts would be then chosen only if insertions are not possible. The function that can evaluate all the moves is the *condensation* function, which calculates the ratio between the number of oligonucleotides in the solution to the length of the solution measured in nucleotides. The usage of the condensation function leads to a transformation of a weak initial solution into a series of well matched clusters. However, using only this function would cause at the end the creation of one cluster only, while the length of the corresponding sequence would be much less than n. Thus, it is necessary to combine both of the criteria in a series of cycles, which at the end will give a solution composed of many elements from the spectrum well matched in clusters. The basic moves are the *condensing* ones. If a maximal value of the condensation function is achieved by more than one move, the algorithm selects the move resulting in a greater number of oligonucleotides in the solution. Hence, the insertion is the most preferred move with shifts, with deletion of an oligonucleotide and deletion of a cluster being next.

After a given number (*condens*) of condensing moves without an improvement of the global criterion function, the method executes a selected number (*extend*) of *extending moves*. Extending moves are feasible moves based on frequency memory

instead of the condensation function. This memory is a structure that remembers the number of times the oligonucleotides have been in solutions (for example, if an oligonucleotide has not been in any solution so far its frequency value is equal to 0, while if an oligonucleotide has been all the times in the solution, the value is equal to the number of performed moves of the method). Extending moves can be of two types: the moves that insert the oligonucleotides being in solutions the most rarely and the moves that delete the most frequent oligonucleotides. The inserting moves are preferred to the deleting moves – deletions are performed only in the case where no insertion is possible. After the execution of the selected number of the extending moves, the method returns to the normal scheme with the condensation as the criterion function. The values *condens* and *extend* were tested in preliminary tests in order to choose the best combination of the parameter values (see Section 4.4).

Inserted or shifted oligonucleotides are remembered on a *tabu list* for a given number of iterations (l_{tabu} – see Section 4.4 for details). The list is checked if an attempt to shift or to delete an oligonucleotide is made, and such moves are prevented if the oligonucleotide is on the list. This procedure makes impossible to reverse immediately the extending moves. The clusters are not remembered on the tabu list since they likely change after each move. Even if the method gets into a cycle of the same moves, it will stay there for a short time, until the next cycle of extending moves. An element from the tabu list may be shifted or deleted together with the cluster containing it. The element from the tabu list may also be deleted if there is no other feasible move. In such case the oligonucleotide which was the longest time on the tabu list is chosen.

To increase the quality of the solution, the algorithm does not stop after a given number of cycles (C) consisting of condensing and extending moves but restarts the search process. A new starting solution is generated on the basis of the frequency values, by the choice of oligonucleotides having lowest frequencies. After that, most of the variables are set to initial values (tabu list, frequency memory). The algorithm stops after a selected number of restarts (R) and gives as the result the best solution found, i.e. the one containing the greatest number of oligonucleotides.

4.2 Hybrid genetic algorithm

Another new approach developed for the problem in question was a genetic algorithm. In general, its effectiveness depends on how the following features are defined: initial population, population size, genetic operators, fitness criteria, and stopping criteria. Below these issues are addressed as well as the implementation of the proposed genetic algorithm for the computational phase of the isothermic SBH problem (see also [BOSW06]).

Initial Population: The initial population consists of s randomly generated individuals, where s is the population size and is kept constant through the generations. s is one of the control parameters (see preliminary computational experiments in Section 4.4 for details).

Encoding of Solutions (Individuals): As each individual represents a solution in the genetic algorithm, it is important to define how a solution is encoded into an individual. Generally there are two types of encoding of a solution: a binary representation or a permutation representation. Considering the fact that a DNA sequence is formed from a number of oligonucleotides, a permutation of $1, 2, \ldots, |\mathcal{S}|$ is chosen to represent an individual. In this representation, each integer corresponds to an oligonucleotide from the spectrum. Every permutation is decoded into a nucleotide sequence which is usually longer than n and its best subsequence (in terms of the largest number of oligonucleotides) with a length not longer than n, is selected as the solution.

Fitness: The fitness of an individual is defined as the number of oligonucleotides from the spectrum used to form the best subsequence, which is not longer than n. Hence, the individual with the largest number of oligonucleotides in the subsequence has the highest fitness. Note, that maximizing the number of oligonucleotides in the solution is not trivial due to the restriction that we need a sequence which is not longer than n. This implies that the genetic algorithm should construct sequences so that the overlap between oligonucleotides is maximized. This information is further used in the crossover operator and it will be explained below.

Selection: Individuals from a current population are selected on the basis of their fitness values to form the mating pool for the reproduction step. The aim of the selection is to keep good individuals and eliminate the bad ones from one generation to another. This selection is performed by the use of the part sum selection procedure [PS00]. This procedure has some components of proportional selection schemes, like the roulette wheel selection, and of remainder schemes, like the deterministic selection (see also [Mic96] for details). Firstly, for each individual the probability of selecting this individual for the parent population is evaluated according to its fitness value. Then, in a deterministic way, the individuals are selected. This procedure consists of the following steps:

1. Let f_i be the fitness value of an individual $i, i = 1, 2, ..., s$, $\overline{f}$ be the mean value of the fitness for all individuals in the population,

$$\overline{f} = \frac{\sum_{i=1}^{s} f_i}{s}.$$

2. Calculate the expected count of each individual, denoted by array F_p

$$F_p[1] = f_1/\overline{f}$$

$$F_p[i] = f_i/\overline{f} + F_p[i-1], \quad i = 2, 3, \ldots, s.$$

3. Generate a random number r from the range (0,1].

4. Assign value 1 to variables i and *count*. The next step is repeated until *count* = s.

5. If $r < F_p[i]$ then put the individual i into array *select*, increment r and *count* by 1; else increment i by 1.

We note that some individuals may be selected more than once and some not at all with this procedure, and the following example illustrates that.

Example 4.1: *Consider the instance where we have four individuals in the population ($s = 4$). Their fitness and values in array F_p are presented in Table 4.1. Assume that $r = 0.75$. Assignment of the individuals into array select is presented in Table 4.2. We see that while individuals 1 and 2 are selected only once, individual 4 is selected twice, whereas individual 3 is never selected.* □

Tab. 4.1: Fitness of individuals and array F_p from Example 4.1.

i	f_i	$F_p[i]$
1	4.50	1.29
2	2.50	2.00
3	1.00	2.29
4	6.00	4.00
$\overline{f}$	3.50	

Tab. 4.2: Selected individuals are stored in array *select*.

i	*count*	$F_p[i]$	r	$r < F_p[i]$	*select*[*count*]	increment
1	1	1.29	0.75	yes	**1**	$r, count$
1	2	1.29	1.75	no	-	i
2	2	2.00	1.75	yes	**2**	$r, count$
2	3	2.00	2.75	no	-	i
3	3	2.29	2.75	no	-	i
4	3	4.00	2.75	yes	**4**	$r, count$
4	4	4.00	3.75	yes	**4**	$count = s$

Reproduction: New solutions (offspring) for a next generation are obtained by applying the crossover and mutation operators to the mating pool obtained in the previous step. The crossover rate c and the mutation rate m were determined during the preliminary computational experiments to set the best combination of parameters (see Section 4.4). In both the crossover and the mutation operators, the concept of

overlap degree in used, which is the number of nucleotides overlapping in two adjacent oligonucleotides. Further, the *total overlap degree* of a sequence is defined as the sum of overlap degrees of all neighboring oligonucleotides in the sequence.

Crossover: Crossover operators from the literature which are known to be working well for other problems were not considered here since they do not fit into this problem so well. In particular, Traveling Salesman Problem (TSP) is closely related to isothermic SBH problem. However, TSP crossover operators, like crossover operators developed for other combinatorial optimization problems for which the solution can be represented as a permutation, are designed without incorporating problem specific information. For example, in the graph for the TSP problem, the cities are on the vertices and with an arc two parameters may be connected: a *distance* between two cities, and a *cost* – road toll – of traveling from one city to another. Even though some of TSP crossover operators do take the distance into account, they ignore the cost of the travel. In isothermic SBH problem, both the total overlap degree and the temperature of the newly created solution are important information, since in case of oligonucleotides rich in A/T nucleotides (longer ones) the overlap degree must be higher (and the temperature lower) than for G/C rich ones. A crossover operator that incorporates both of these information will enhance the new solution obtained after the crossover operation. After experimenting with several crossover operators to incorporate these problem specific information, the following one has been chosen as the best crossover operator for this problem. The construction of a new individual departs from the classical crossover operator, and it is based on the pattern of structured weighted combination [Glo94] (more general approach allowing to incorporate these ideas into solving TSP are presented in Section 4.3).

Customary operations are replaced by weighted linear combination of vectors that preserve discrete relationships and feasibility of conditions. The approach is called **structured weighted combination** [Glo94] and it is based on tree properties:

- *Property 1 – representation.* Each vector is a set of votes for particular decision. For example, it may be a decision of assigning a particular binary value '0' or '1' to a variable or if the solution is represented as a permutation – a decision of establishing the precedence relationship between a pair of elements.

- *Property 2 – trial solution.* A process is defined which translates the vector (for

example a permutation of elements) into a feasible solution.

- *Property 3 – update.* Defined rules are applied to the vectors, such that the Properties 1 and 2 continue to hold. The vectors vote for particular features, depending on weights assigned with them. For example, votes coming from the vector with higher value of weights are applied more often.

These properties imply that the decisions made upon the time are deterministic, and are based on the decisions made upon time. The weights of each vector are evaluated during computational process depending on the decision outcome. The vectors that have the highest values of the weights are enabled to have greater influence in the current decisions. This ensures that the total influence of each vector over time is proportional to the assigned weights and that new vectors will have greater weights.

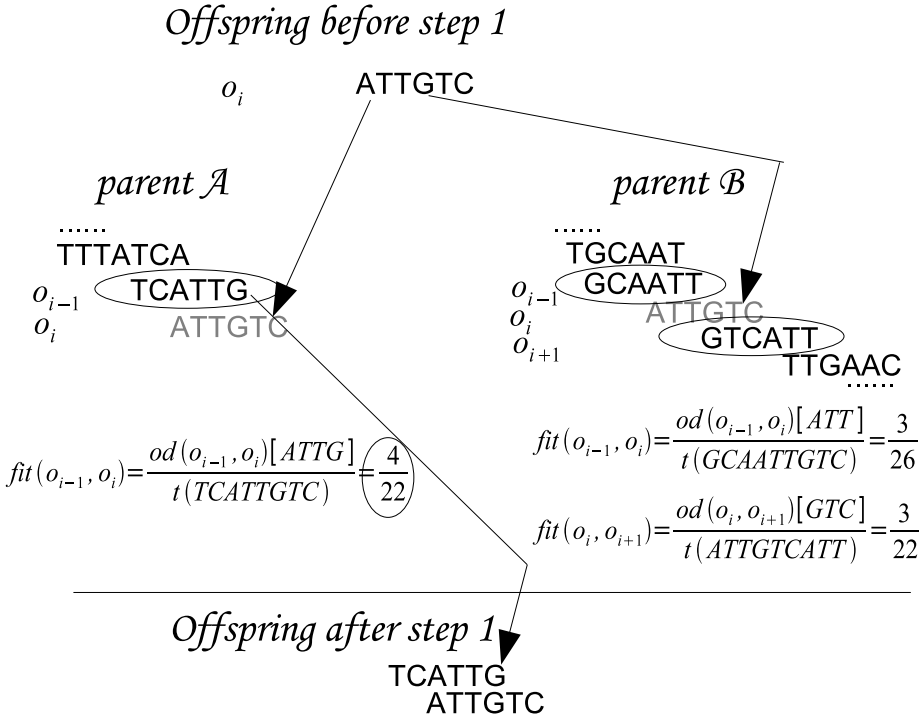
In the crossover operator, the process is not going randomly, as it usually goes in genetic algorithms but it is based on the concept of structured weighted combinations. The individuals are treated as *vectors*, in which precedence relationship between the oligonucleotides is established. The offspring choses from the vectors (parents) the *best votes*, which are the orders of the elements in the individuals. Both the overlap degree and the temperature of the obtained sequence are taken into account, and the child inherits the best characteristics from his parents. Such algorithmic construction process that uses "voting evaluations" based on the composition of the parents and the problem objective is also a basic feature of path relinking [GL93].

The crossover operator is applied to the parents population with a rate of c ($c{\cdot}m$ individuals in the population undergo crossover). It builds a new individual (offspring) from two parents selected randomly. For the offspring, the first oligonucleotide is randomly selected, call it o_i, and let o_f and o_l denote the first and the last oligonucleotide in the offspring, respectively. Initially, $o_f = o_l = o_i$. The *crossover operator selects* next oligonucleotides by *applying the following rules*:

1. Identify the oligonucleotides in both of the parents which are preceding and succeeding oligonucleotide o_i and which are not in the offspring yet. Select the one that fits better with o_i. If there is a tie, select one randomly. The function used here evaluates the fitness of temporary solutions so that they can be compared. The function (fit) is defined as the total overlap degree (tod) of a sequence (which is the sum of overlap degrees, od, of all adjacent

oligonucleotides) divided by the temperature (t) of the sequence constructed so far. If the selected oligonucleotide is the succeeding (preceding) one, then place it after (before) o_i in the offspring. Update o_f and o_l accordingly. Figure 4.1 illustrates this step.

Fig. 4.1: Creation of the offspring – first step. Predecessors and successors of oligonucleotide o_i are first identified in both of the parents. Note, that there is no successor of o_i in parent A. The one with the highest value of fit joins the offspring.

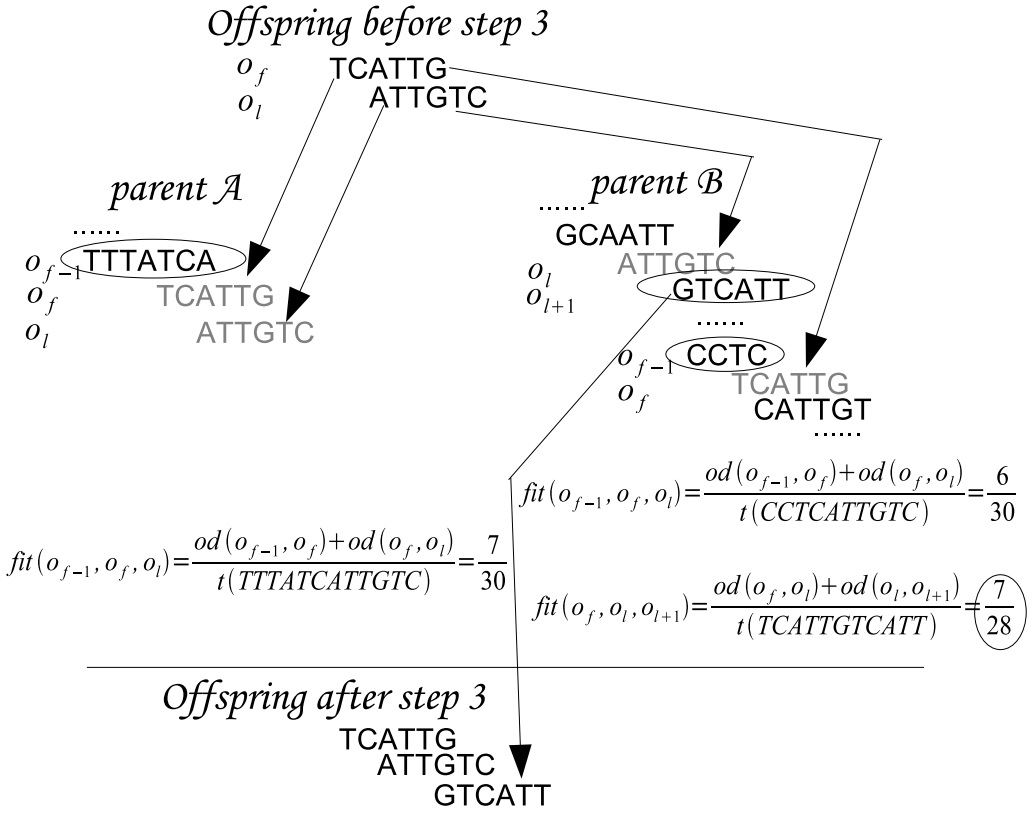


2. If all the preceding and succeeding oligonucleotides are already in the offspring, then select the oligonucleotide with the highest overlap degree with o_i among the oligonucleotides not used so far.

3. Now the crossover operator considers the newly formed offspring and takes the first and the last oligonucleotides of this offspring instead of o_i in the above description. The predecessors of o_f (the first oligonucleotide in the offspring) and the successors of o_l (the last oligonucleotide in the offspring) which are not in the offspring yet are identified in the parents. The one that fits better with

the offspring is selected. If there is a tie, one is selected randomly. This step is illustrated in Figure 4.2. If there is no unused oligonucleotides, apply Step 2 appropriately modified to o_f and o_l, respectively. Step 3 is repeated until all oligonucleotides are in the offspring.

Fig. 4.2: Creation of the offspring – third step. Predecessors of the first oligonucleotide in the offspring (o_f) and successors of the last oligonucleotide in the offspring (o_l) are first identified in both of the parents. o_l has no successor in parent A. The one with the highest value of fit joins the offspring.



One of the aims in developing the above crossover operator is to keep the best characteristics of the parents in terms of the neighboring oligonucleotides. If two oligonucleotides are adjacent to each other in both parents, which are having good fitness values, then we want to keep this structure in the offspring. The other aim of this operator is to keep the length of the new sequence created at minimum. Note,

that this crossover operator produces only one offspring which will be a feasible solution so that we do not need a repair mechanism to deal with infeasible solutions.

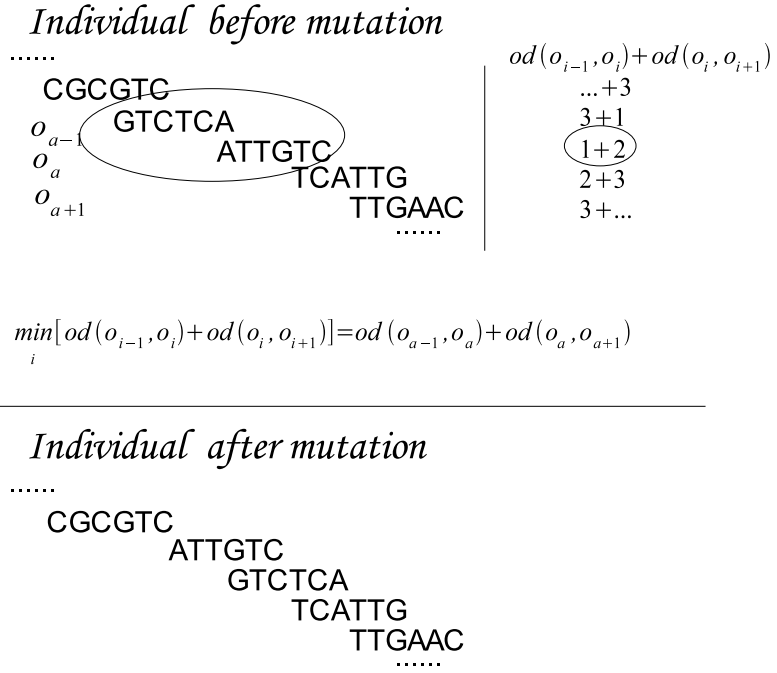
As mentioned earlier, this crossover operator was chosen due to the significant improvement of the results among several other crossover operators that were tested. One of the tested operators, for example, looked only in the parents for the best successor of the oligonucleotide that appeared in the offspring instead of searching in the parents for the best predecessor or successor of the newly formed offspring. Another operator created four offspring instead of producing one offspring from two parents. In addition to testing different crossover operators, various functions for evaluating the best move (fit) were also tested. In the chosen crossover operator (and in some others tested) the following functions were tried: the number of oligonucleotides in the offspring divided by the length of the sequence, the total overlap degree of the sequence (tod), the number of oligonucleotides in the offspring divided by the temperature of the sequence (t), total overlap degree of the sequence (tod) divided by the length of the sequence. The function that best harmonized with the crossover operator was the total overlap degree of the sequence (tod) divided by the temperature of the sequence (t).

Mutation: The following mutation operator is applied to both the parents and the offspring population with a rate of m (it means that in $m \cdot s \cdot |\mathcal{S}|$ individuals in the population the mutation occurs).

Select an individual randomly. Find the oligonucleotide with the smallest sum of overlap degrees with both of its neighbors (if more than one exist, select the first one found). Swap this oligonucleotide with the adjacent oligonucleotide that has the lower overlap degree with it. If both of the adjacent oligonucleotides have the same overlap degree, select one at random. If the selected oligonucleotide is the first (last) oligonucleotide in the permutation, then move the selected oligonucleotide to the end (beginning) of the permutation list. Figure 4.3 illustrates how the oligonucleotides are swapped.

Creation of the next generation: The next generation is composed of all individuals from the offspring population and depending on the value of the parameter c ($c \cdot s$ individuals, coming from the crossover operation, are in the offspring) the rest

Fig. 4.3: Firstly, for each oligonucleotide, overlap degrees with its adjacent oligonucleotides are evaluated. Next, an oligonucleotide with the smallest sum of overlap degrees with its neighbors, is selected (o_a). Then, the oligonucleotide is swapped with its neighbor that has a smaller overlap degree with it (o_{a-1}).



of the individuals with the best fitness value are taken from the parents population.

Stopping criterion: Stopping criterion is chosen as the number of generations without improvement (*iter*) of the criterion function – the number of oligonucleotides composing a solution, (see the preliminary computational experiments – Section 4.4, for details).

4.3 Structured crossover in TSP

The approach presented in Section 4.2 is of more general feature and, after some adaptation, can be also used for solving other combinatorial problems, especially the ones having a solution represented as a permutation of elements (i.e. in TSP – cities,

which a salesman has to visit). The best features of the hybrid genetic algorithm are selected and described how they can be involved in solving Traveling Salesman Problem.

The crucial elements of the algorithm are the crossover operator, which inherits some features of the structured combinations of individuals (vectors), weights, assigned to vectors, which votes for the individual in the selection procedure, and the way how the pairs of parents are selected from the pool of the selected individuals.

Let us now consider classical TSP [Bła88]. We have set $C = \{c_1, c_2, \ldots, c_n\}$ of the cities, and distance $d_{ij} \in N$ between every pair of cities $c_i, c_j \in C$. The goal is to find such an order $< c_{k[1]}, c_{k[2]}, \ldots, c_{k[n]} >$ that Equation 4.1 is minimized.

$$\sum_{j=1}^{n-1} d_{k[j],k[j+1]} + d_{k[n],k[1]} \tag{4.1}$$

The cities can be represented as vertices in a complete graph. With each arc a cost is assigned, which is a distance between two vertices (cities). In such graph a Hamiltonian cycle of minimum cost is searched for. However, in many process models, among them the biological ones, instead of looking for a Hamiltonian cycle, a Hamiltonian path is searched for. Here, in this example, a Hamiltonian path will be searched for.

An individual (a vector) is represented as a permutation of all the cities. The fitness (f_i) is calculated for every individual, and it is the cost of the travel through all the cities. The lower f_i, the better the individual is. The fitness is the weight of the vector and is treated as the basis for the selection procedure. If the weight of the vector is higher (the value of fitness f_i is lower), then the individual is selected more often as a parent for the next generation. The selection procedure can be for example, part sum selection procedure or other method known in the literature (roulette wheel selection, stochastic remainder with replacement).

Let us assume that we have 5 cities, and the distances between the cities are presented in Table 4.3. The algorithm starts with the initial population of individuals, randomly created. For example the individuals may be: $P_1 =< 1, 2, 3, 4, 5 >$, $P_2 =< 5, 4, 3, 2, 1 >$, and $P_3 =< 1, 4, 2, 5, 3 >$. Let us calculate the fitness f_i for each individual. The results are gathered in Table 4.4.

With the aid of the selection procedure, we select the individuals as parents for the next generation: P_3 and P_1, P_3 and P_2, and P_1 and P_2. Three pairs are selected,

Tab. 4.3: The cost of traveling between two cities.

	1	2	3	4	5
1	-	5	2	10	30
2	5	-	60	50	10
3	2	60	-	50	10
4	10	50	50	-	5
5	30	2	5	5	-

Tab. 4.4: The fitness values of the individuals.

individual	fitness
P_1	$f_1 = 5 + 60 + 50 + 5 = 120$
P_2	$f_2 = 5 + 50 + 60 + 5 = 120$
P_3	$f_3 = 10 + 50 + 10 + 5 = 75$

because from one pair of parents only one offspring is created. Pairing of the individuals should cause that the greatest diversity in the population is kept (the pairs of parents can not be repeated), and the possibility of the greatest improvement is ensured (individuals of the best fitness f_i are paired together). Now, the crossover operator works as follows:

1. Take a random city as the first element in the new offspring.

2. Check the successors of the last element and predecessors of the first element in the offspring in both parents, and choose the best one. The function fit that compares possible successors and predecessors of the newly created offspring checks for example for the smallest cost (d_{ij}) of the travel, if the city joins the trip. Notice, that in the first iteration of this step the first element is also the last one.

3. If there is neither successor nor predecessor which can be joined to the solution (the possible elements might have already been used in the solution), take the best one which has not yet been used. Return to step number 2 unless all the cities are in the offspring.

An example, how new offspring are created is presented in Table 4.5.

Tab. 4.5: Creation of the new offspring. The element, which was randomly chosen first is marked in the column 'individual' as bold. In offspring O_1 city 2 was chosen first. Next, the predecessors of '2' and successors of '2' in parents P_3 and P_2 were considered for the smallest distance. Thus, the possible extensions of offspring $< 2 >$ were $< 2, 3 >$, $< 1, 2 >$,$< 2, 5 >$, and $< 4, 2 >$. The smallest distance was for $< 1, 2 >$ and city 1 was added to the offspring. The rest of the creation process runs accordingly.

offspring	parents	individual	fitness
O_1	P_3, P_1	$< 1, \mathbf{2}, 5, 3, 4 >$	70
O_2	P_3, P_2	$< 2, 5, \mathbf{4}, 3, 1 >$	67
O_3	P_1, P_2	$< 1, 2, 3, 4, \mathbf{5} >$	120

Analyzing the results presented in Table 4.5, we can see that the fitness values (f_i) of two offspring were improved (O_1, O_2). Further application of the crossover operator may decrease the value of f_i. In order to increase the possibility of finding a global optimum (instead of a local one) one should use sufficiently large population of individuals to keep the diversity, and introduce to the genetic algorithm a mutation operator, which can work randomly or deterministically (i.e. changing the worst overlap in an individual).

It should be noted, that the formulation of the crossover operator differs from the structured weighted combination introduced in [Glo94]. In the original algorithm for TSP the authors propose what follows: the solution is represented as a vector, being a permutation of all the cities, and the votes decide about precedence relationships between the cities. To every vector a weight is assigned and the vote from a vector is stronger if it has higher weight (a function that evaluates possible extensions of an offspring is a combination of the weight and the cost of a vote). In the algorithm presented above, and also in the algorithm presented in Section 4.2, the weight – here it is the fitness function – decides how many times the individual will be selected as parent. The votes here are also the precedence relationships between the elements in the individual but they are once again evaluated with the function (fit) to check

which one of the votes coming from parents is the best. The decision of choosing a particular element depends only on the result of a comparison and not on the weights of individuals.

4.4 Tuning parameters

After developing the two algorithms (tabu search one and genetic algorithm one) for our main problem (SBH with isothermic libraries allowing errors in spectra) some preliminary tests were performed to set the best values for the parameters used in these algorithms. A full factorial experimental design could be preferred here since it allows to explore parameter interactions. However, it also requires too many runs due to all parameter combinations so only a few levels were tried for each parameter. An alternative to this method is the approach presented here in which certain parameter values are fixed and then the best value of one parameter at a time could be found. This method ignores the interactions between all parameters but allows for more levels to be tried for each parameter. This method is especially suited to problems for which we do not know which levels of the parameters should be considered in a full factorial experimental design, like the isothermic SBH problem.

For the tabu search algorithm the following values were taken for the preliminary test: $condens \in \langle 5, 11\rangle$; $extend \in \langle 4, 10\rangle$; $C \in \langle 300, 600\rangle$; $R \in \langle 4, 6\rangle$; $l_{tabu} \in \langle 25, 35\rangle$. The following steps were taken to find the best combinations of the parameters: 40 instances were taken with the spectra coming from sequences of length 400 nucleotides and containing 10% of negative errors and 10 % of positive errors. Mean values for all the results were checked to obtain the best parameters. *condens* value was tested first and all the other parameters were set to their border values. Thus, two values for each parameter were tested. $condens \in \langle 5, 11\rangle$ was tested with the increment of 1. The best values were {9, 10}. Next, $extend \in \langle 4, 10\rangle$ was taken with the increment of 1 and the best values were reached in range $\langle 6, 8\rangle$. Thus, for next tests to choose other parameters, values $\in$ {6, 7, 8} were used. Number of cycles $C \in \langle 300, 600\rangle$ with the increment of 100 was examined next and the best values of this parameter were in the set {400, 500, 600}. Value of R was set to 4, since after forth restart a better solution was never found. For the $l_{tabu} \in \langle 25, 35\rangle$ with the increment of 5 it seems that the best value was 30. In a smaller range, for these chosen values of the

parameters, more tests were peformed and finally the following values were chosen:

- The number of condensing moves, *condens*, was set to 10.
- The number of extending moves, *extend*, was set to 7.
- The number of cycles composed of condensing and extending moves, C, was set to 400. This value hardly rely on the combination of parameters *condens* and *extend*.
- The number of restarts, R, was set to 4.
- The length of the tabu list l_{tabu} was set to 30.

For the genetic algorithm the following values were taken: $c \in \langle 0.5, 1.0\rangle$; $m \in \langle 0.0001, 0.1\rangle$; $s \in \langle 50, 300\rangle$; $iter \in \langle 50, 300\rangle$. 40 spectra coming from sequences of 400 nucleotides with 20% of positive errors and 20% of negative errors were tested and mean values for all the results were checked to obtain the best parameters. The crossover parameter c was tested first and similarly to the experiment described above all other parameters were set to their border values. The crossover rate was taken as $c \in \langle 0.5, 1.0\rangle$ with the increment of 0.05; the best range of values was found to be $\langle 0.9, 1.0\rangle$. Hence, three values for c, namely $\{0.9, 0.95, 1.0\}$, were used for next tests to choose other parameters. Next, the mutation rate m was tested in three different ranges: $m \in \langle 0.0001, 0.001)$ with the increment of 0.0002, $m \in \langle 0.001, 0.01)$ with the increment of 0.002, and $m \in \langle 0.01, 0.1)$ with the increment of 0.02. The best values for m were observed to be $\{0.001, 0.01\}$. For the population size s, the interval $s \in \langle 50, 300\rangle$ with the increment of 50 was used and the best range was found to be $s \in \langle 150, 200\rangle$. The stopping criterion *iter* was tested in increments of 50, however, since no significant improvements were observed, the value 50 was chosen. Subsequently, for the ranges of the parameters obtained in the above tests, more tests were performed with small variations of chosen values and the following values for the parameters were finally selected:

1. The size of the population (s) was chosen as the half of the sequence length, with the exception of 200-nucleotide sequence for which the size of the population was the same as the length of the sequence (for example: the sequence with 500 nucleotides has 250 individuals in each population).

2. The crossover rate (c) was set to 0.9, which means that $0.9 \cdot s$ individuals undergo crossover.

3. The mutation rate (m) was set to 0.001 and it means that the mutation is applied to the individuals from the parents and the offspring populations with frequency $0.001 \cdot s \cdot |\mathcal{S}|$.

4. The number of the iterations ($iter$) without improvement of the criterion function (i.e. the number of oligonucleotides composing a solution) was set to 50.

The preliminary experiments suggest a very small mutation rate (m=0.001) and the mutation improves the result in some of the cases. In Table 4.6 the results of testing different instances with and without mutation operator are presented. Each value in the table is the mean value for 40 instances. The average improvement of the value of the similarity between the original sequence and the sequence obtained with the genetic algorithm (function that calculates the similarity value is described in more details in Section 4.6) for all tested instances with mutation operator when m=0.001 is equal to 1.18.

Tab. 4.6: Average similarity of the sequence obtained during computations to the original sequence for two values of m (m=0 means that mutation never occurs) and average improvement of the similarity value for m=0.001.

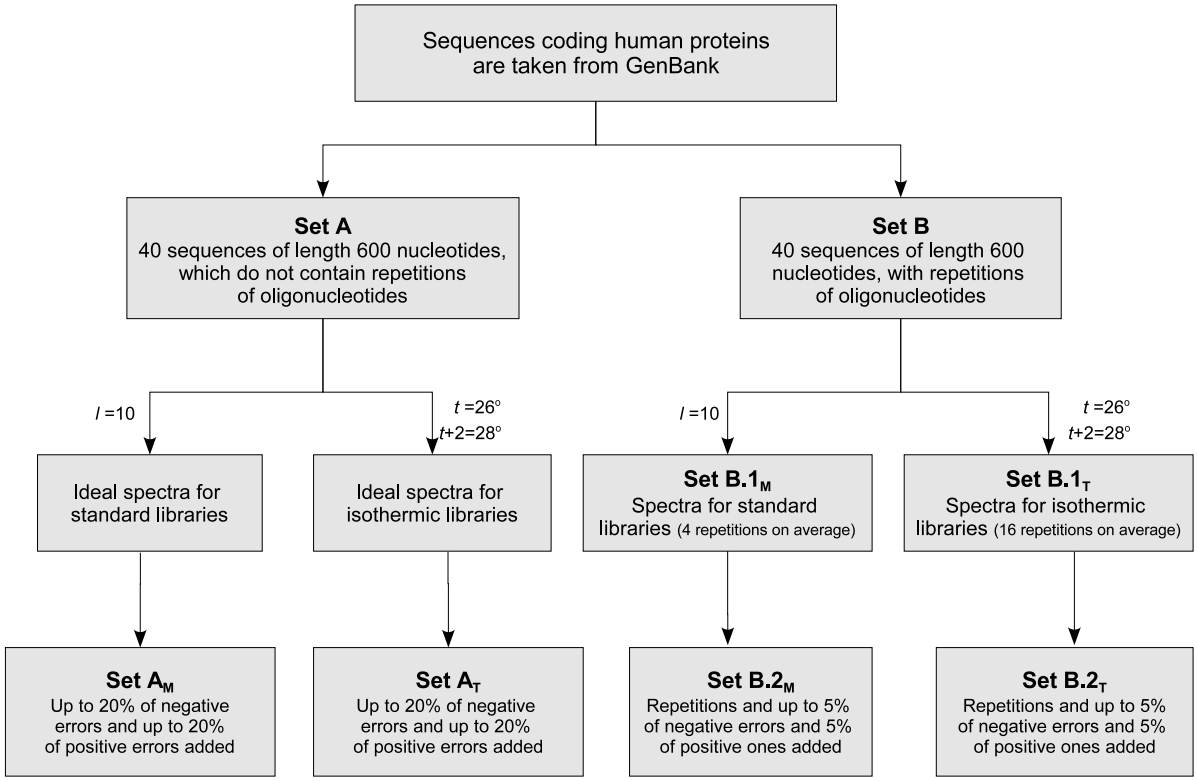
errors	n	m=0	m=0.001	improvement
±20%	200	99.26	99.34	0,08
	400	99.12	100.00	0,88
	500	98.98	99.79	0,81
	600	96.86	99.81	2,95
	average	98,555	99,735	**1,18**

Note, that the above parameter values cannot be taken separately as optimal values, these parameter values work well together, and if any of the parameters will be changed, the rest of the parameters should be tuned appropriately.

4.5 Preparation of spectra

All the sequences used in the experiments were obtained from the GenBank, National Institute of Health in USA. They are DNA sequences coding human proteins. Instances created on the base of these sequences were used also for testing the algorithm presented in [BGK04] in the computational experiments in Sections 4.6 and 5.2, and form a certain standard testbed when comparing SBH algorithms. The accession numbers in GenBank of the sequences are given in Appendix A (cf. also http://bio.cs.put.poznan.pl/, where these sequences and the way they were obtained are described in more details). The sequences were grouped in two sets. First set, let us call it *set A*, consists of the sequences which contain no repetitions of oligonucleotides, neither from standard libraries nor from isothermic ones. In *set B* the sequences contain repetitions of oligonucleotides from both libraries (see Figure 4.4).

Fig. 4.4: The scheme illustrating how the spectra are prepared.



Sequences for the computational experiments are prefixes of the above sequences

taken from GenBank. These prefixes are strings of nucleotides of lengths 200, 400, 500, and 600. Simulated hybridization experiments were made by extracting oligonucleotides of length 10 nucleotides – for standard libraries – and oligonucleotides whose melting temperatures t and $t+2$ were 26^oC and 28^oC – for isothermic libraries. These values of temperatures and of lengths were chosen in order to compare the results of the proposed algorithms for isothermic SBH with the algorithms for standard SBH [ZWZ03, BGK04], that used spectra with oligonucleotides of the same length $l = 10$. Two libraries of temperatures 26 and 28 degrees have a similar cardinality as one library of the length equal to 10.

Note, that there is a large difference between average repetition rates in the standard and isothermic spectra. In order to check these rates, 1000 randomly chosen sequences coding human proteins were obtained from GenBank and two spectra for each sequence (a spectrum with oligonucleotides of length 10, and the second one with oligonucleotides of temperature 26 and 28 degrees) were created, taking prefixes of those sequences of length for 600 nucleotides. For each spectrum the number of the oligonucleotide repetitions was calculated, and next the average repetition rates were determined. For the standard spectra it was equal about 4 and with the range $\langle 1, 17\rangle$, while for the isothermic spectra it was equal 16.5 with the range $\langle 4, 30\rangle$. The higher value of repetitions in isothermic libraries probably comes from the fact that the repeated parts in the sequences are rich in G and C, and in this case the oligonucleotides of constant temperature are shorter than the ones with constant length. (For example, the substring GCGGCCCGC appearing twice in a sequence causes three repetitions in the isothermic spectrum: GCGGCCC, CGGCCCG, GGCCCGC, while none in the standard spectrum with $l = 10$). The averages and range values are preserved in the sequences from Set B.

Both types of errors were then added to such generated spectra. Numbers of positive and negative errors were calculated according to the cardinalities of the spectra because a spectrum cardinality varies for each sequence. In sets A_M and A_T spectra contain 5% or 20% of every type of errors. The notation 400±20% means that a spectrum prepared from a sequence of length 400 nucleotides contains 20% of random positive errors (different from the oligonucleotides already present in the spectrum) and the same number (20%) of random negative errors. In other words, 20% of randomly chosen oligonucleotides of the original spectrum are missing (so

called negative errors) and in addition 20% of oligonucleotides in the spectrum are erroneous (so called positive errors). As a result, the cardinality of this new erroneous spectrum is the same as the old correct one. Spectra with ±20% of errors contain the same positive and negative errors as spectra with ±5%, plus 15% of extra ones. This ensures that obtaining a sequence from a spectrum with the higher error rate is not easier than from the spectrum with less errors. Spectra were sorted alphabetically, thus no information about the order of oligonucleotides in the sequences was kept.

From the instances in sets $B.1_M$ – with equal-length oligonucleotides and $B.1_T$ – with equal-temperature oligonucleotides, two sets $B.2_M$ and $B.2_T$ were generated by adding into the spectra 5% of random positive errors and up to 5% of random negative errors.

4.6 Computational experiment

In this section, the evaluation of the performance of the algorithms is presented. Computational experiments were performed on a PC with Pentium 4, 2.2 GHz processor and 512 MB of RAM. The algorithms were implemented in ANSI C language. The solutions obtained with the algorithms can be evaluated in two ways. Firstly, during computations, while no information about the order of oligonucleotides is provided, the number of oligonucleotides composing the solution determines the quality of the solution, because this is the main criterion function used in the heuristics. Secondly, once computations are over, the sequence generated by the algorithm is compared to the original one, which results in the similarity value. In the real world it is in general impossible, but in the experiments presented here we know the sequences being bases for the spectra. The similarity is quantified according to the Needleman-Wunsch algorithm [NW70] (cf. also the Smith-Waterman algorithm, [Wat95, SM97]). The algorithm evaluates the similarity of two sequences (the examined one and the solution) on the basis of three parameters: m (match of nucleotides), d (mismatch) and g (gap in a sequence). Let A_{pt} be the score of the alignment of two sequences, counted in points: for every match of nucleotides in the alignment $m = 1$ point, for every mismatch, $d = -1$, and for every gap in the alignment $g = -1$ point. n_1 and n_2 are the lengths of two compared sequences, and min and max values are calculated as follows:

if $n_1 > n_2$ then $min = n_2 d + (n_1 - n_2)g$

$max = n_1 m + (n_1 - n_2)g$

else $min = n_1 d + (n_2 - n_1)g$

$max = n_2 m$

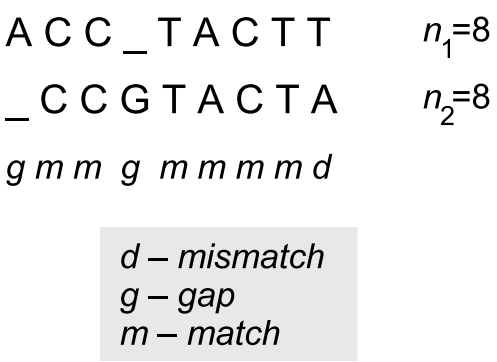
Then, the alignment score, expressed as percentage is computed according to the equation:

$$100 \cdot \frac{A_{pt} - min}{max - min}$$

Example 4.2 illustrates the way of computing the alignment score.

Example 4.2: *Let us consider two sequences ACCTACTT and CCGTACTA, with lengths* $n_1 = n_2 = 8$. *The best alignment of these sequences is presented in Figure 4.5. In the alignment there are two gaps, one mismatch and 6 matches. Calculating the score of the alignment, where for every match 1 point is given, and for every mismatch and gap 1 point is subtracted, the score is equal to 3 points. To obtain the alignment score expressed as percentage, the values min and max are calculated.* $min = n_1 \cdot d + (n_2 - n_1) \cdot g = 8 \cdot (-1) + (8-8) \cdot (-1) = -8$ *and* $max = n_2 \cdot m = 8 \cdot 1 = 8$. *Now, the alignment score is equal to* $100 \cdot \frac{A_{pt} - min}{max - min} = 100 \cdot \frac{3-(-8)}{8-(-8)} = 68,75[\%]$. □

Fig. 4.5: An example of the alignment of two sequences.



Basing on the above criteria, the performance of the proposed algorithms is compared to those of the earlier algorithms used for the standard DNA sequencing by hybridization problem. For the isothermic sequencing the only existing algorithm accepted spectra without errors, and the comparison was not possible. The best algorithms proposed for standard DNA sequencing problem with both positive and

negative errors were considered, namely the tabu search algorithm which uses a scatter search for diversification by Blazewicz et al. [BGK04], the algorithm by Zhang et al. [ZWZ03], and the enhanced genetic algorithm by Bui and Youssef [BY04]. In the following, each entry of the tables presented summarizes the results of the tests for 40 different sequences obtained from the GenBank (for the preparation of spectra see Section 4.5), except for the algorithm by Zhang et al., where a greater number of instances were tested for one entry, but it was not tested for all the cases considered here, and the algorithm by Bui and Youssef, where the tests were performed on different sequences under the same conditions of errors in the spectrum.

In Table 4.7, the results of the five algorithms are compared. The instances that were used contained no errors coming from repetitions (Sets A_M and A_T, and the first two algorithms used the instances, which where created under the same conditions – the error rate and the length of the sequences). There are a few measures for the evaluation of the obtained sequence, which can be divided into two main types:

- one type shows how much the solution is valuable for biologists and here we have: the similarity of the obtained sequence to the original one and the number of original sequences reconstructed by the algorithm, and
- the second one determines how does the algorithm work: the quality of the solution expressed by the number of oligonucleotides composing the solution and the number of optimal solutions derived by the algorithm (in the sense of the criterion function used).

Moreover, for two algorithms, tabu search and genetic algorithm, additional measure is given – the deviation which shows how the results of the similarity value for particular instance differ from the average value. Note, that in these tests the original sequences were known as they were DNA sequences taken from GenBank, but the comparisons of the sequences were calculated after the algorithms have finished the computations. Each entry of the rows with the heading "No. of original sequences", is composed of two numbers a/b and it means that among b instances tested, the algorithm found original sequences for a of them – with 100% of similarity. The cases for which the results were not given by the authors, are denoted by mark '-'. The measure usage of the oligonucleotides in the solution is compared only for the algorithms, for which the maximization of the usage of oligonucleotides is the criterion

Tab. 4.7: Comparison of the algorithms for the isothermic sequencing with the algorithms designed for standard sequencing. Tests performed on sets A_M and A_T, except for the first two algorithms, for which different instances were tested under the same conditions (error rate, length of the sequence).

length of the sequence	200		400		500		600	
error rate	±5%	±20%	±5%	±20%	±5%	±20%	±5%	±20%
Standard sequencing								
Method by Zhang et al. [ZWZ03]								
Similarity [%]	100.0	-	100.0	-	-	-	-	-
No. of original sequences	90/90	-	80/80	-	-	-	-	-
Enhanced genetic algorithm [BY04]								
Similarity [%]	-	97.6	-	92.9	-	92.0	-	-
No. of optimal solutions	-	26/40	-	13/40	-	13/40	-	-
Tabu and scatter search [BGK04]								
Usage of oligonucleotides [%]	99.9	99.9	99.9	99.7	99.8	99.6	99.8	99.4
Similarity [%]	99.9	98.4	98.9	95.7	95.8	92.4	95.8	88.5
No. of optimal solutions	39/40	39/40	37/40	28/40	33/40	27/40	32/40	20/40
No. of original sequences	39/40	38/40	37/40	28/40	32/40	27/40	32/40	19/40
Isothermic sequencing								
Tabu search								
Usage of oligonucleotides [%]	96.8	96.1	96.6	95.3	96.0	95.5	96.2	95.1
Similarity [%]	85.2	78.7	75.9	69.7	75.6	70.2	76.5	68.8
No. of optimal solutions	6/40	2/40	0/40	0/40	2/40	0/40	1/40	1/40
No. of original sequences	8/40	3/40	2/40	0/40	3/40	0/40	2/40	0/40
Deviation [%]	274.2	324.5	211.9	161.9	268.2	199.3	302.6	153.9
Genetic algorithm								
Usage of oligonucleotides [%]	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.9
Similarity [%]	99.5	99.2	99.5	99.0	99.6	98.9	98.5	96.1
No. of optimal solutions	40/40	40/40	40/40	40/40	40/40	40/40	40/40	40/40
No. of original sequences	39/40	38/40	39/40	39/40	39/40	37/40	39/40	37/40
Deviation σ [%]	1.7	0.4	5.9	11.9	3.0	8.6	23.7	35.0

function. For these methods the criterion function was the number of oligonucleotides used for composing the solution, thus it is very important to obtain a high value of the usage. The value 100% means that the number of oligonucleotides used for the construction of solutions is the same as the number of proper oligonucleotides in the spectra. For example, if in a spectrum there were 200 oligonucleotides and the error rate were equal 5% (it means that there are 10 negative errors and 10 positive errors), then the solution composed of 190 oligonucleotides would have 100% of the usage.

The first algorithm, described in [ZWZ03], is mostly designed for positive errors in addition to some repetitions that can occur in a sequence. However, it works well with a small rate of negative errors – up to 5%. Note, that the algorithm assumes that the maximum number of consecutive missing oligonucleotides is equal 3 (the shift between two neighboring oligonucleotides is not greater than 4). The other algorithms do not have such additional knowledge. As the output of the algorithm, one gets all the sequences with the maximum value of the number of oligonucleotides used in the solution, however, it is not said how the sequence is chosen to be the correct one. It is assumed here that one of the sequences is with 100% similarity value, thus the average value of similarity in Table 4.7 is 100%. This algorithm was not tested by the authors for instances with higher rate of negative errors nor for longer sequences.

The enhanced genetic algorithm [BY04] was tested only for the cases with 20% error rate. The method works quite well with the similarity of the obtained sequence to the original one on average not lower than 92%. The number of original sequences found for the 200-nucleotide sequences is equal to 26, and for the longer sequences only 13 out of 40.

The method described in [BGK04] seems to work well for hard instances, with high rate of both types of errors. The quality value – the usage of oligonucleotides – is very high for all the cases and mostly it is higher than 99.5%. However, even though the similarity is quite high (for sequences of 200-nucleotide length and the error rate in the spectrum 5%, the similarity is above 99%), the number of original sequences found decreases sharply with the increase of the length of the sequence; for sequences with 600 nucleotides in the case of 20% error rate, only 19 out of 40 original sequences were found with total average similarity value 88.5%. One should notice that the tested instances are derived from longer sequences and contain higher

error rate (±20%) in comparison with the algorithm presented in [ZWZ03], and since more oligonucleotides are missing in the spectrum, the sequencing problem is more difficult. The tabu and scatter search algoritm works better than the enhanced genetic algorithm comparing either the similarity value or the number of original sequences found.

For the last two algorithms in Table 4.7, an additional measure of the performance of the algorithms is also presented: the deviation of the similarity values from the average similarity value. It is defined as the square root of the variation according to the function:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{b}(x_i - \overline{x})^2}{b-1}}$$

where x_i are the consecutive values of the similarity, $\overline{x}$ is the mean value, and b is the number of tested instances ($b = 40$).

The tabu search algorithm for isothermic sequencing is presented in the following rows. The qualities of the solutions solved with the algorithm are contained in the range (95%-97%), which are worse than the qualities for the tabu and scatter search algorithm. This has an effect on the similarity values which are in range $< 68.8\% - 85.2\% >$ and it is not satisfying result. Also the original sequences were rarely found, in the case of 5% error rate in the spectrum and the length of the sequence 200 nucleotides only 8 out of 40 sequences were found, while for 20% error rate and the sequences longer than 200 nucleotides the original sequences were never reached. The deviation of the similarity values is rather high for every tested case, and it comes from the fact that the range of the similarity values is too wide. One can obtain the original sequences or the sequences which are not similar to the original one. This strongly depends on the instance.

The results of the genetic algorithm are presented in the last few rows. The initial solution is generated randomly what may cause the difference of the results for different runs of the algorithm (It was not the case for the tabu search algorithm, where the initial solution was generated deterministically with the greedy heuristic). Thus, the algorithm was tested 10 times for every instance, each time starting from a different population of individuals. The results of more detailed tests are presented in Tables C.1-C.8 in Appendix C and the analysis of the results is placed in the next paragraphs. Each entry in Table 4.7 under the similarity and the usage of

oligonucleotides is the average of 400 tests (40 instances multiplied by 10 runs). On the other hand, if within 10 times for every instance the algorithm found the original sequence or the optimal solution at least once, the number of original sequences or the number of optimal solutions was increased respectively. If the value is for example equal 36/40, it means that for four tested instances the algorithm could not find the optimal solution or the original sequence for any of the 10 runs. The algorithm works very well for all the instances, giving high similarity rate ranging from 96% in the worst case to almost 100%. For the hardest instances: sequences of length 600 and ±20% error rate, 96% of similarity and 37 original sequences (out of 40) are outstanding results. It should be noted here, that for these instances the criterion: the usage of oligonucleotides in the solution is very high (on average it is equal to almost 100%) and at least once every instance was solved optimally. The reason for the difference between values of the two criteria: the usage and the similarity, is the existence of more than one optimal solution derived from an instance. Without an additional (biochemical) information about the original sequence, it cannot be indicated which one is better because they are equivalent with respect to the data provided to the algorithm. One might notice that the deviation is much smaller than for the tabu search algorithm. For the hardest instances it rises up only to 35%. This case is more difficult and the results are not as obvious as for easier cases, shorter sequences, thus, many different sequences were obtained from one spectrum. This rather high value of deviation follows the reason that despite the algorithm found 37 original sequences (with the similarity equal to 100%), there were a few sequences with rather small similarity value, which was around 60%.

More detailed results of the computational experiment of the genetic algorithm for isothermic libraries are presented in Appendix C . The length of tested sequences varies from 200 to 600 nucleotides and the error rate in the spectrum is equal to ±5% and ±20%. The algorithm was tested with 40 instances, 10 times for each spectrum. The results in Tables C.1-C.8 are the average values of 10 runs for each instance.

In Tables C.1-C.4, where the error rate is ±5%, the most significant is that the usage is always equal to 100% and this follows optimal usage of oligonucleotides in every case. However, similarity of the obtained sequence to the original one is not always optimal. It is coming from the fact that from the same number of oligonucleotides from the spectrum, it is possible to obtain more than one sequence of nucleotides.

Very difficult instance to solve turned out to be the spectrum coming from the sequence number 4. In this case, the algorithm never found the original sequence despite the optimal usage in every case, what may be a result of the absence of a key oligonucleotide(s), which might differentiate two similar parts of the sequence. Also the sequence number 20 was difficult to reconstruct for every length of the sequence. Analyzing Table C.4, where the sequences are longer, one might see that usually repeating the computations may help in finding the original sequence. In this table we can also observe the increase of the deviation. It follows the fact that there exist more than one solution, and the differences between the obtained sequences and the original one are greater. But only an additional biochemical experiment can point out the correct sequence out of a few output ones, since the quality of the solution (usage of the oligonucleotides) is the same for different sequences.

Tables C.5-C.8 contain the results of the computational experiment with the spectra with $\pm 20\%$ error rate. Only for the sequences of length 200 nucleotides the usage of oligonucleotides is almost in every case optimal. However, for longer sequences the average value of usage is higher than 99%. The decrease of these few oligonucleotides in the solution often results in lower similarity of the obtained sequence to the original one. An exception is one instance in Table C.8 for the sequence number 34, where despite the number of oligonucleotides was lower than optimum, the similarity was still 100%. In these tables, similarly to Tables C.1-C.4, we can see again that sometimes from the optimal number of the used oligonucleotides we can get more than one possible sequence (see for example sequences number 4 and 20 in every table, or sequences 10,19, and 33 in Table C.8). Sometimes, even though the experiment was repeated, the algorithm didn't find the original sequence. For $\pm 5\%$ error rate only the sequence number 4 was this case, what can be caused by the occurence of a repeated fragment shorter than l in the sequence.

Table 4.8 presents a comparison of CPU times for the algorithms working with isothermic sequencing, tested under the same conditions. In Table 4.8 each entry is the mean value for all the results.

It is very significant that the genetic algorithm works for a similar period of time for the same length of the sequence but different error rate (Table 4.8). Although it works longer than the tabu search algorithm for the case with $\pm 5\%$, it is much faster while error rate increases, and the results are better in both cases (cf. Table 4.7). It

Tab. 4.8: CPU time of computations [s].

	Tabu search		Genetic algorithm	
n	±5%	±20%	±5%	±20%
200	3.5	9.7	6.5	8.5
400	18.0	74.7	23.9	30.8
500	30.0	125.5	46.5	53.6
600	45.9	199.9	80.9	91.6

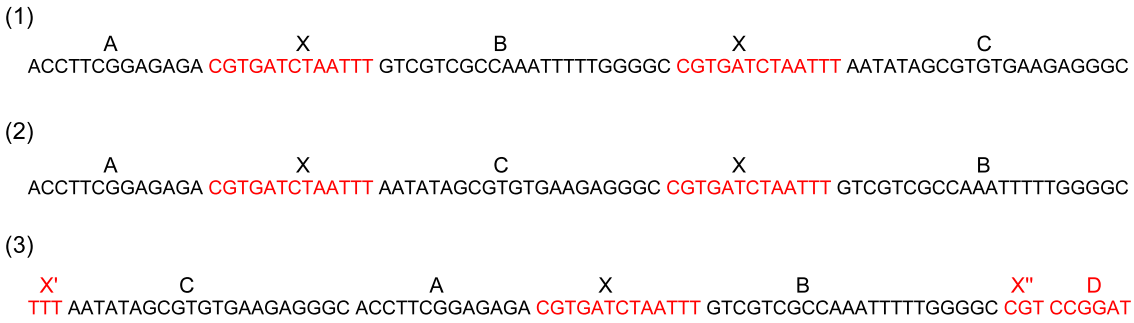
might be due to the different functions used to compare the intermediate solutions in the algorithms. The function is more sophisticated and time consuming for the genetic algorithm and although the algorithm works more slowly for easier instances, it is much more effective what results in shorter time of finding the best solutions for harder instances. In the tabu search algorithm, the function for evaluating intermediate solutions was the condensation, i.e., the number of oligonucleotides in the solution divided by the length of the produced sequence. In the genetic algorithm, different functions were tested (as explained in Section 4.2), including the condensation within the crossover operator. Among the functions that were tested, the best one appeared to be the combination of overlaps between neighboring oligonucleotides and the temperature of the sequence. It indirectly takes the length of oligonucleotides into consideration. This function provides very good results – almost all obtained sequences are optimal from the criterion function point of view, because the number of oligonucleotides is almost the same as the number of proper oligonucleotides in the spectrum. Moreover, they are also optimal from the biochemical point of view, because similarity to original sequences is almost 100% (cf. Table 4.7). For the tabu search algorithm the quality of the solution, although is very high (95%-97%), does not lead to high similarity of the obtained sequence, when compared to the original one.

At last, the algorithms were tested on the set B in which sequences contain repetitions of oligonucleotides. The case in which repetitions occur is more difficult to solve because there are more possibilities of obtaining solutions which are of the same value of the criterion function (they are equal in mathematical sense) but sequences can be very different from the original one (see Example 4.3 for details illustrating

that phenomenon).

Example 4.3: *Let a sequence be composed of some fragments which we label as A, X, B, X, and C (see Figure 4.6 (1)). Notice that fragment X is repeated twice in*

Fig. 4.6: An example of an erroneous reconstruction of a sequence in case of repetitions of oligonucleotides in the sequence.



the sequence. Let in the spectrum be all the oligonucleotides that are contained in the original sequence – the repeated oligonucleotides appear only once in the spectrum. Now, during the reconstruction process the algorithm finds a sequence in which fragments B and C are swapped (Figure 4.6 (2)). The solution is composed of almost all oligonucleotides from the spectrum (depending on how long the oligonucleotides are). Unfortunately, the similarity is low (only the first part of the sequence composed of A and X well matches the original sequence). Suppose that in the spectrum some errors appear. Now the reconstruction of a sequence is presented on Figure 4.6 (3). The second repeated part X could not be fully reconstructed due to repetitions and lack of some oligonucleotides in the spectrum. Part X' which is the end of X and together with C is now in the beginning of the solution. Next parts A, X, B, and X" come, X" being the beginning of part X. The reconstructed sequence is shorter than n and additionally another element D (which is a positive error) joins the solution. The usage of elements from the spectrum in the solution is now greater than 100%, because among all the proper oligonucleotides from the spectrum, additional one, enters the solution. The similarity value of the obtained sequence to the original one is again very low. □

Analyzing the above example we can see that some solutions which are wrong for the biologists are preferred while concerning the global criterion function. Without an additional experiment it is not possible to choose the solution which is similar to the original sequence.

The preliminary tests for the spectra with repetitions were made for the genetic algorithm with isothermic libraries (sets $B.1_T$ and $B.2_T$) and for the tabu and scatter search algorithm [BGK04] for the standard libraries (sets $B.1_M$ and $B.2_M$). The genetic algorithm was chosen for the isothermic libraries because it was better in the case of no repetitions. It was tested once again ten times for every instance to start from different random initial solutions (cf Table 4.9). More detailed tests are presented in Tables C.9 and C.10 in Appendix C.

Tab. 4.9: Results of tests for sets B.1 – sequences of length 600 with repetitions of oligonucleotides without additional errors in spectra – and for sets B.2 – sequences of length 600 with repetitions of oligonucleotides and additional up to 5% of negative and 5% of positive errors

Algorithm		Set B.1	Set B.2
Tabu and scatter search	Usage of oligonucleotides [%]	99.86	99.55
Standard libraries	Similarity [%]	88.45	82.63
	No. of optimal solutions	33/40	23/40
	No. of original sequences	14/40	10/40
	Running time [s]	84.30	129.95
Genetic algorithm	Usage of oligonucleotides [%]	99.99	100.00
Isothermic libraries	Similarity [%]	78.12	78.44
	No. of optimal solutions	39/40	35/40
	No. of original sequences	9/40	6/40
	Running time [s]	90.63	108.67
	Deviation	59.38	61.65

Analyzing the results from Tables 4.7 and 4.9, we can state that the genetic algorithm with isothermic libraries deals very well with the instances where no repetition of oligonucleotides appears, while in the case with repetitions the tabu and scatter

search method with standard libraries works better. This might be for the following reason: isothermic libraries cause so many repetitions that solving the problem is much more difficult. In the description of the preparation of set B (sequences with repetitions of oligonucleotides) it was mentioned that the average number of repetitions in set $B.1_M$ is 4 for standard libraries and in set $B.1_T$ is 16 for isothermic libraries for the same sequences. Although isothermic libraries cause fewer experimental errors, they give more repetition errors than standard libraries, because an isothermic library consists of elements shorter than 10 nucleotides for the GC rich sequences. We may notice that the genetic algorithm solved most of the sequences at least once optimally (see no. of optimal solutions). However, the similarity is very low – 78.12% – and only for 9 instances the algorithm found the original sequence. Although the usage of oligonucleotides in the solution for tabu and scatter search method is similar to the usage for genetic algorithm and less solutions were solved optimally, the similarity of obtained sequences to the original ones were 10% greater and the number of the original sequences also increased.

The results for set B.2, where positive and negative errors were added, are slightly worse for both of the algorithms. Surprisingly, in case of the genetic algorithm the usage of oligonucleotides is equal 100% but only 35 out of 40 instances were solved optimally. This follows the single presence of many repeated oligonucleotides in the spectrum. The solutions found by the algorithm were composed of more oligonucleotides than the optimal number of elements from the spectrum, what means that some positive errors were included in the solution. Looking at the more detailed results of the genetic algorithm tests in Table C.10 in Appendix C, one might notice that, for example, for sequences No. 11, 17, 36 and 39 the usage of oligonucleotides is more than 100% and the similarity is on average equal 65%. The deviation is also very high what suggests that the sequence obtained by the algorithm is very distant from the original sequence, while the solution is better than the optimal one comparing the criterion function. This situation did not occur in set $B.1_T$ because there were no positive errors and the algorithm could use only the proper oligonucleotides.

The value of the deviation of the similarity is higher for sets $B.1_T$ and $B.2_T$ than for set A_T. This follows the fact that obtaining the proper (original) sequence strongly depends on the instance and sometimes more than one sequence with very low similarity was found (compare for example in Table C.9 in Appendix C sequences

No. 21, 23, 33, and 34 where some sequences had 100% similarity but the average value was 85% or lower). In Tables C.9 and C.10 only in one case (No. 13) the original sequence was obtained in all ten runs of the algorithm. In most of the cases the algorithm never found the sequence or it found it only in one run (cf. sequence No. 2 and 9 in Table C.9). Thus, there is a quite low probability that the algorithm may find the original sequence. Only an additional biochemical experiment can reveal the ambiguities.

Following the above discussion, one might suppose that connecting the standard libraries causing less repetitions with the hybrid genetic algorithm would give the best results. Although there were a few different genetic algorithms applied to this problem [BKK02, BY04], a new one, which outperforms the existing algorithms, is proposed in the next chapter. The revised hybrid genetic algorithm is based on the main pattern of the approach described in Section 4.2 but adopted to handle standard libraries.

5. COMPARISON OF TWO APPROACHES TO SEQUENCING BY HYBRIDIZATION: THE STANDARD AND ISOTHERMIC ONES

In the previous chapter the isothermic approach was presented. The tests of the new algorithms proposed for this approach showed that this approach is prone to errors coming from repetitions (especially in the sequences rich in G/C content, where the length of the oligonucleotides decreases). The genetic algorithm designed for isothermic libraries outperformed the existing algorithms for standard sequencing in the case of the sequences without repeated regions. On the other hand, in the case of the repetitions in the spectra the algorithm rarely found original sequences although the maximization of the criterion function, which is the number of oligonucleotides in the solution, reached optimum. It cannot be surely stated what is the reason of the worse results for isothermic libraries in the case of repetitions in the spectra. Is it a higher repetition rate in case of isothermic libraries or is the algorithm not working well enough to solve the problem? This chapter answers both questions. In order to compare in reliable way the two approaches to sequencing by hybridization: the standard and the isothermic ones, once again a genetic algorithm was implemented. This time the aim was to keep the new algorithm as similar to the algorithm from Section 4.2 as possible, but adopted for the equal-length oligonucleotide libraries.

In this chapter the genetic algorithm devoted to standard SBH problem (cf. Definition 2.2) is described (Section 5.1). In Section 5.2 the algorithms for different approaches are compared together in tests on the same sequences.

5.1 Revised hybrid genetic algorithm for standard sequencing by hybridization problem

The revised hybrid genetic algorithm is based on the main pattern of the approach described in Section 4.2 but adopted to handle standard libraries (see also [BGS+06]).

As before the *data* coming from the hybridization experiment are the spectrum S – here a set of oligonucleotides of the same length l – and the length of the original sequence n.

Initial population consists of s randomly generated individuals. s is the population size, and it is kept constant during computations.

Individual (chromosome) is a permutation of $|S|$ indices of oligonucleotides from the spectrum. Every permutation is decoded into a sequence, usually longer than n, and its best subsequence of length not greater than n, i.e., including the largest number of oligonucleotides from the spectrum, is a potential solution.

Fitness (objective function) is the number of oligonucleotides from the spectrum used to form the best subsequence, which is not longer than n.

Selection. Individuals from a current population are selected on the basis of their fitness values to form the mating pool for the reproduction step. This selection is performed by using the part sum selection procedure in the implementation.

Reproduction. New individuals (offspring) in next generations are obtained by applying the operators: structured crossover and mutation to the mating pool obtained in the previous step, with the probability, respectively, c and m. Thus, in the next generation $c \cdot s$ new individuals will be created with structured crossover and at most $m \cdot s \cdot |S|$ new individuals will come from mutation. The probabilities (c, m) were determined during our preliminary tests to set their best combination.

Structured crossover. Two parents are chosen randomly from the mating pool. The first oligonucleotide in the offspring is chosen randomly. This oligonucleotide, o_i, is identified in both of the parents. The construction of the child from this starting point departs from that of classical crossover operations, and uses a strategic design that accords with the concept of *structured combinations* as discussed in Section 4.2. The construction proceeds as follows. The successors of o_i and the predecessors

of o_i in the parent individuals are considered. The one that fits better in front of o_i (for the predecessors in the parent chromosomes) or at the end of o_i (for the successors in the parent chromosomes) is placed at the proper position in the offspring. The new oligonucleotide together with o_i in the offspring now form a *block*. In the next steps instead of o_i, the terminal oligonucleotides of the block are considered and predecessors of the first oligonucleotide of the block and successors of the last oligonucleotide of the block are checked in the parent individuals. If there is neither unused successor nor unused predecessor in the parent individuals, then the best fit oligonucleotide from the remaining individuals is chosen. The best fit oligonucleotide is the one with the highest value of the ratio of the number of oligonucleotides in the block to the length of the sequence. This in fact means that the newly built sequence should be the shortest. The procedure of creating the individual is stopped when all oligonucleotides from the spectrum are in the block. In this formulation one child is created from two parents. As previously mentioned, the process of creating this new offspring is different from the classical genetic algorithm, where most of the process goes randomly. New structured crossover operator incorporates the idea of structured combination by treating the individuals as *vectors* to establish precedence relationships between oligonucleotides. Selection of an oligonucleotide at each step can be compared to choosing the *best vote* from two vectors (parents). Hence, the offspring inherits the best characteristics (votes) from parents (vectors).

Mutation can occur both in the parents and in the offspring population with the probability m. The individual to be mutated is selected randomly. In this individual the oligonucleotide with the lowest overlap degree is found (if more than one exist, the first one is chosen). This oligonucleotide is swapped with its neighbor that has lower overlap degree with it. If the selected oligonucleotide is the first (last) one in the individual, then it is swapped with the last (first) oligonucleotide. The new individual then replaces the old one.

Creation of the next generation. Apart from the newly created individuals, the best individuals from the parent population go on to the next generation.

Stopping criterion is a chosen number of generations in which no improvement occurs in the objective function.

After the preliminary tests a set of best parameter values was established. The cardinality of the population has been set to the half of the length of the target

sequence. For the shortest sequences s is equal to 100 individuals, and for the longest sequences 300 individuals. The probability that mutation of each oligonucleotide in the chromosome occurs is $m = 0.001$, and the probability that structured crossover occurs for any two individuals is $c = 0.9$, where the stopping criterion has been set to be 50 generations without improvement of the objective function.

5.2 Computational experiment

The revised hybrid genetic algorithm was tested on the same sets A and B as the algorithms for standard sequencing in the previous chapter. The algorithm is compared with other algorithms dedicated to standard sequencing: tabu and scatter search presented in [BGK04], enhanced genetic algorithm presented in [BY04], and the algorithm presented in [ZWZ03], and the results of the comparison are presented in Table 5.1. Test of the first two algorithms are performed on set A_M and of the two other algorithms – on the sets prepared under similar conditions – sequences do not contain any repeated oligonucleotides. The algorithm from [ZWZ03] assumes additional knowledge of consecutively missing oligonucleotides, which was not the case for the other algorithms.

Analyzing the results of tests performed with the spectra prepared from the sequences without repetitions of oligonucleotides, one can notice that the revised hybrid genetic algorithm works very well even for very hard instances – for sequences of length 600 nucleotides and 20% of error rate. In almost all of the cases it finds target sequences, composed of 100% of the proper oligonucleotides. It happens that a high rate of usage of oligonucleotides in the solution does not mean that the obtained sequence is similar to the examined one, due to the existence of more than one optimal solution. However, analyzing more detailed results placed in Appendix C in Tables C.11 - C.18 we can be sure that if the solution was composed of the proper number of elements from the spectrum, then at least one sequence was the original one or with high similarity (above 99.8%) to the original one. For such high results only few oligonucleotides are wrong or wrongly ordered, usually at the ends of sequences. Also the deviation shows that there are only small differences in the similarity values. The highest value is for the case of 200-nucleotide sequence and $\pm 20\%$ of errors in the spectrum. When comparing this result with Table C.15 one can see that for a

Tab. 5.1: Comparison of results of different algorithms for standard sequencing. Sequences have no repetitions – set A_M

length of the sequence	200		400		500		600	
error rate	±5%	±20%	±5%	±20%	±5%	±20%	±5%	±20%
Revised hybrid genetic algorithm								
PC Pentium 4, 2.0 GHz, 512 MB RAM								
Usage of oligonucleotides [%]	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.98
Similarity [%]	100.00	99.84	100.00	100.00	99.85	99.80	99.87	99.81
No. of optimal solutions	40/40	40/40	40/40	40/40	40/40	40/40	40/40	40/40
No. of original sequences	40/40	40/40	40/40	40/40	40/40	39/40	40/40	39/40
Deviation	0.00	11.77	0.00	0.00	0.59	0.39	0.41	2.31
Running time [s]	4.3	6.3	11.9	20.5	18.1	27.2	25.7	44.2
Tabu and scatter search [BGK04]								
PC Pentium 4, 2.0 GHz, 512 MB RAM								
Usage of oligonucleotides [%]	99.93	99.93	99.90	99.67	99.83	99.64	99.84	99.36
Similarity [%]	99.87	98.44	98.96	95.70	95.76	92.41	95.82	88.50
No. of optimal solutions	39/40	39/40	37/40	28/40	33/40	27/40	32/40	20/40
No. of original sequences	39/40	38/40	37/40	28/40	32/40	27/40	32/40	19/40
Running time [s]	5.6	9.0	47.4	68.5	82.9	125.5	127.6	186.3
Enhanced genetic algorithm [BY04]								
PC Pentium 4, 2.4 GHz, 512 MB RAM								
Similarity [%]	-	97.60	-	92.90	-	92.00	-	-
No. of optimal solutions	-	26/40	-	13/40	-	13/40	-	-
Running time [s]	-	1.5	-	8.6	-	15.1	-	-
Method by Zhang et al. [ZWZ03]								
SGI Origin 2100								
Similarity [%]	100.00	-	100.00	-	-	-	-	-
No. of original sequences	90/90	-	80/80	-	-	-	-	-
Running time [s]	1.6	-	19.0	-	-	-	-	-

sequence number 13 only in 4 out of 10 runs the algorithm gave as a result the original sequence (100% similarity) while in the other runs sequences of 55% similarity composed of the same number of oligonucleotides were returned.

The overall results are improved significantly as compared to the previous algorithms, especially in the number of generated original sequences. The algorithm in [ZWZ03] generates as the output all possible solutions for an instance and one of the output sequence is the original one, but the authors do not provide that information (cf. the comment for this algorithm in Section 4.6). Another case, which was mentioned in Section 3.1.1 in the description of the algorithm is that the spectrum is extended with artificial elements such that all the negative errors are eliminated and only the positive errors are present in the spectrum. For all the pairs of oligonucleotides for which the distance (the shift) d is not greater than δ, $1 < d \leqslant \delta$ the path of length d is constructed, where shifts between consecutive oligonucleotides are equal 1. For every such a pair of oligonucleotides the spectrum is extended with $d-1$ oligonucleotides from the path. The authors assume that $\delta \leqslant 3$, but they did not show how the computation time would change if δ is increased to l. The time could grow drastically and there would be too many output sequences due to many positive errors in the extended spectrum.

In Table 5.2 the genetic algorithms for the two approaches to sequencing by hybridization are compared for set A, the revised hybrid genetic algorithm and the genetic algorithm, which was presented in Chapter 4.2. The numbers of optimal solutions are the same for both genetic algorithms – at least in one run for each algorithm the optimal usage of oligonucleotides was reached. The average values of the usage are similar and in most of the cases the optima are reached. Differences start when comparing the similarity of the obtained sequences to the original ones. For the standard libraries the genetic algorithm usually solved each instance optimally and the differences between the sequences, if appeared, were very small (see the deviation values). In the case of isothermic libraries at least one of the sequence was never solved optimally despite a few runs for every instance. However, the average similarity values are also very high and they are better than for the other algorithms dedicated to standard or isothermic libraries, see Table 4.7.

Next, the algorithms were tested on the instances from sets $\mathrm{B.1_M}$ and $\mathrm{B.2_M}$, where sequences contain repetitions, and the possibility of obtaining more than one

Tab. 5.2: Comparison of results of genetic algorithms for different approaches to SBH. Sequences have no repetitions – set A

length of the sequence	200		400		500		600	
error rate	±5%	±20%	±5%	±20%	±5%	±20%	±5%	±20%
Revised hybrid genetic algorithm for standard libraries PC Pentium 4, 2.0 GHz, 512 MB RAM								
Usage of oligonucleotides [%]	100.00	100.00	100.00	100.00	100.00	100.00	100.00	99.98
Similarity [%]	100.00	99.84	100.00	100.00	99.85	99.80	99.87	99.81
No. of optimal solutions	40/40	40/40	40/40	40/40	40/40	40/40	40/40	40/40
No. of original sequences	40/40	40/40	40/40	40/40	40/40	39/40	40/40	39/40
Deviation	0.00	11.77	0.00	0.00	0.59	0.39	0.41	2.31
Running time [s]	4.3	6.3	11.9	20.5	18.1	27.2	25.7	44.2
Genetic algorithm for isothermic libraries PC Pentium 4, 2.0 GHz, 512 MB RAM								
Usage of oligonucleotides [%]	100.00	100.00	100.00	99.96	100.00	99.98	100.00	99.93
Similarity [%]	99.53	99.15	99.47	98.95	99.60	98.89	98.47	96.08
No. of optimal solutions	40/40	40/40	40/40	40/40	40/40	40/40	40/40	40/40
No. of original sequences	39/40	38/40	39/40	39/40	39/40	37/40	39/40	37/40
Deviation σ [%]	1.72	0.39	5.87	11.92	3.01	8.57	23.68	35.01
Running time [s]	6.5	8.5	23.9	30.8	46.5	53.6	80.9	91.6

solution with the same value of the criterion function increases. The results of tests performed on the instances from sets B are presented in Table 5.3. For the tabu and scatter search algorithm and for the genetic algorithm they are the same as in Table 4.9. Comparing these results, we can notice that the revised hybrid genetic algorithm is much better than the other algorithms, especially for the number of optimal solutions and the number of original sequences found. Also the similarity is a few per cent higher than the results of tabu and scatter search algorithm. The usage of oligonucleotides is very high for all the algorithms, however, optimal solutions were more frequently reached by two genetic algorithms – for standard and isothermic libraries.

For the revised hybrid genetic algorithm the usage of oligonucleotides in the solution is on average equal to 100% and the instances were solved optimally. For

Tab. 5.3: Results of the computational tests of the algorithms for standard and isothermic sequencing. Set B.1 – sequences with repetitions only. Set B.2 – sequences with repetitions and additional negative and positive errors up to 5%

	Set B.1	Set B.2
Revised hybrid genetic algorithm with standard libraries		
Usage of oligonucleotides	100.00	100.00
Similarity [%]	92.75	91.23
No. of optimal solutions	40/40	40/40
No. of original sequences	34/40	30/40
Deviation	72.63	63.99
Running time	37.65	41.85
Tabu and scatter search [BGK04] with standard libraries		
Usage of oligonucleotides [%]	99.86	99.55
Similarity [%]	88.45	82.63
No. of optimal solutions	33/40	23/40
No. of original sequences	14/40	10/40
Running time [s]	84.30	129.95
Genetic algorithm with isothermic libraries		
Usage of oligonucleotides [%]	99.99	100.00
Similarity [%]	78.12	78.44
No. of optimal solutions	39/40	35/40
No. of original sequences	9/40	6/40
Deviation	59.38	61.65
Running time [s]	57.42	64.78

the detailed computational results of the experiment see Tables C.19 and C.20 in Appendix C. In Table C.20 only for two sequences No. 16 and 20 the usage of oligonucleotides exceeded 100%. In these cases the obtained sequence was never found although the similarity is very high 97%-98%. It is the result of holes in the seqeunce, which could not be reconstructed due to negative errors coming from repetitions, and they were filled with some oligonucleotides being positive errors in the spectrum. The rest of the sequence was properly obtained. For the genetic algorithm dedicated to isothermic libraries the situation in which the usage of oligonucleotides exceeded 100% took place more often (cf. 6 spectra in Table C.10). For these cases the similarity was on average equal to 73%. The reason of this is a higher repetition rate in these spectra, and what follows, more possibilities to construct the solutions and including positive errors.

For standard libraries there were many spectra for which the reconstruction of a sequence from one spectrum led in every run to the same original sequence: in Table C.19 – 13 spectra and in Table C.20 – 9 spectra. For isothermic libraries there is only one spectrum (no. 13) for which genetic algorithm resulted in the original sequence in all 10 runs (cf. Tables C.9 and C.10). Moreover, it was also more difficult to find the optimal solution in case of isothermic libraries, than in the case of standard ones. In Table C.10 we can see that there are 10 instances which were not solved optimally at least once, among them there are 5 spectra for which the optimal solution was never found.

Summing up the results of the computational experiment for two different approaches to sequencing by hybridization, performed with genetic algorithms based on the similar scheme, we can state that optimal solutions were found with both algorithms, slightly more frequently for standard libraries, when in the spectra repetitions occur. However, for the isothermic libraries the reconstruction led very often to many different sequences, what was noticeable by high deviation values. The genetic algorithm found then different solutions with the same criterion function for different runs of the algorithm. Usually, at least one of the sequences was the original one. In the case of repeated parts in the original sequences, both algorithms had problems in the reconstruction, even though the solutions were optimal from the point of view of the criterion function. The genetic algorithm for isothermic libraries only in less than a quarter of all the instances obtained the proper sequence. This is the result

of the property of the isothermic libraries which contain shorter oligonucleotides in case they are rich in 'G', 'C' nucleotides. This in turn causes more repetitions in the parts of sequences rich in G and/or C.

Following the above remarks one may say that the revised hybrid genetic algorithm for standard libraries is the best suited for solving the SBH problem, especially for sequences with repetitions. It overcomes by far other approaches known from the literature.

To this end let us also mention that the sequence which is the result of the algorithm should be confirmed with additional biochemical experiment. Sometimes it may be necessary to use some longer oligonucleotides in the reconstruction part – or when the sequence is very repetitive, combine the isothermic approach with the multistage one.

6. ASSEMBLING – LARGE SCALE SEQUENCING

The assembling problem was formulated in Section 2.3 in two versions: in case of no errors in the input sequences and in general case, when the sequences may come from two strands of the DNA molecule and there may be some sequencing errors in the input fragments. In the next section, the algorithm is proposed, which incorporates the novel approach to sequencing – pyrosequencing: the input fragments are shorter than in the classical assembling and the vector of quality scores for every nucleotide in the sequence is provided (see Section 2.2 for the description). In Section 6.2 the results of computational test are presented.

6.1 New algorithm for the assembling problem

As the input to the assembling problem we have a multiset of fragments over the alphabet {A, C, G, T}. The lengths of the fragments differ depending on the biochemical method of sequencing which was chosen and are in the range from a few dozens or so (Solexa's approach, see at: www.solexa.com) up to a few hundreds of nucleotides (Sanger's approach [SNC77]) and there may exist the inclusion relation between them. The algorithm presented in this section is designed especially for the sequences of rather high quality and of length about 100 nucleotides, which are coming from 454 Sequencer. But in general the method may be used for sequences with different lengths. The output is a sequence having all the input fragments as the subsequences. The criterion for the evaluation of the solution can be its length, which is minimized, or its likelihood, which has to be maximized.

If there are no errors coming from the experiment and the fragments are from one strand of a DNA helix then we have an ideal case. This case is not considered here in

more details, since it is only of a theoretical nature. In general, the input sequences contain errors: insertions, deletions and substitutions of nucleotides. The fragments may also come from different strands of a DNA helix. The problem becomes more difficult because imperfect matches have to be allowed when aligning two fragments.

Unfortunately, both of these problems are NP-complete. The first problem without errors in the input set is the shortest common superstring problem. Errors in the input fragments make the second problem even more difficult to solve. As the output one usually gets a number of contigs (the contig being a few up to a few thousand fragments overlapping together) instead of one sequence.

The proposed algorithm solving the DNA sequence assembling problem is divided into three parts. The first part builds an overlap graph for all the input sequences. It requires as the parameters, values of the minimum overlap between two sequences and of the error bound, the latter being the percentage of the mismatches allowed in the overlap of two sequences. The comparison is done also for the reverse complementary sequences to the input ones, due to the possibility that the fragments may come from both strands of a DNA helix. In the second phase, a graph is constructed with the fragments as the vertices and two vertices are connected with an arc if there is a feasible overlap between the two fragments. Next, a path is searched for which passes through all the vertices (either the original fragments or their reverse complementary ones). Usually, it is not possible to make one path in the graph and as the output several paths are returned. At the end, in the third part, the *consensus sequence* is determined by voting of fragments on every position of the alignment.

Building the overlap graph requires a comparison of all the pairs of fragments which is $O(n^2)$, where n is the number of input fragments. Determining a possible overlap of two fragments with the Smith-Waterman algorithm ([Wat95], cf. also Needleman-Wunsch algorithm [NW70] in Section 4.6) takes time $O(k^2)$, k being the length of the longer sequence. The complexity of this problem makes it impossible to determine whole overlap matrix for big instances. For example, if we want to recover the genome of a bacteria which is of length around 2 Mbp with the fragments of 100bp and average overlap being 15, then we get around $3 \cdot 10^5$ fragments. The time of computations is extremely long, and there is need to limit the comparison of time consuming Smith-Waterman algorithm only to the promising pairs of sequences. In order to narrow the range of compared fragments, the sequences are divided into

smaller subsequences shifted by 1 nucleotide, called *windows* of the length l and then coded with hashing function. Only the windows which are in the *minimum overlap area* (it is a parameter for the algorithm) are considered and only the ones which are of good quality. The quality of a window is calculated on the basis of quality score which is obtained from the flowgram by the 454 Sequencer. If the quality of at least one nucleotide does not exceed the *minimum quality* (another parameter of the method) value, then such a window is not reliable and is rejected. If the number of the same windows for a pair of sequences exceeds a bound given by a parameter, and the order of the windows is the same, then such a pair is marked as *promising*. Next, for all the promising pairs of fragments, the overlap together with an error of mismatches are calculated. If there is the inclusion relation between two sequences (with some mismatches allowed) then this fact is remembered.

The alignment of two sequences is calculated according to the Smith-Waterman algorithm. For every match of nucleotides the value m is added to the score, for every mismatch the penalty d is added to the score and for every gap, g. Generally $m = 1$, $d = -1$, $g = -1$. However, the gaps usually appear in bunches and should be scored more accurately with the convex gap function: $\gamma(n)$: for all n, $\gamma(n+1) - \gamma(n) < \gamma(n) - \gamma(n-1)$, where n is the number of gaps. The first gap is punished with the highest penalty and for every next one the smaller punishment is given. Such an approach is difficult to achieve in practice because it requires on every position checking the number of gaps which is currently given. At every iteration we have to go back n steps in the worst case and the complexity of counting the score is $O(N^2M)$, where N and M are the lengths of the sequences, and $N > M$. Instead, affine gaps are commonly used in practice, which try to approximate the convex gap function. In this approach, the first gap is punished with the highest penalty and every next one has the same, smaller, penalty. The function is defined as follows: $\gamma(n) = d + (n-1) \cdot e$, where d is the value of the opening gap and e is the value of the extended gap. In this case, to compute the optimal alignment, we have to remember at the similarity matrix entry (i, j) the best score if the gap is open, and the best score if the gap is not open. We increase the memory space because we have to remember additional matrices, but do not increase polynomially the time complexity, which was the main problem for the convex gap function.

Having the score of alignment: the offset of overlapping of two sequences and the

error rate, we can construct now a graph in which the vertices are the fragments and the arcs connect the vertices for which the alignment score was calculated and the values of the offset and the error rate are in the range of feasible values. Next, some additional operations has to be done within the graph, which help to lengthen the path, which will be searched for in the graph. The hashing algorithm could have omitted some pairs of sequences, not marking them as promising. These three steps are necessary to repair the lack of arcs or the excess of some vertices:

- If there is the inclusion relation between sequence a and b, such that $a \subseteq b$, then a is deleted from the input set, together with sequence $\vec{a}$ (the reverse and complementary sequence to sequence a). The arcs from the predecessors of a and to the successors of a are becoming the predecessors and the successors of b.

- If there is an arc $a \longrightarrow b$, then there should also be an arc between the reverse and complementary fragments to sequences a and b: $\vec{b} \longrightarrow \vec{a}$. If the arc $\vec{b} \longrightarrow \vec{a}$ is not present, then the alignment between $\vec{b}$ and $\vec{a}$ is calculated and the appropriate arc is added to the graph.

- If vertex a has got two successors $a \longrightarrow b$ and $a \longrightarrow c$, then the relation $b \; ? \; c$ is also checked.

After recovering the missing arcs and deleting the unnecessary vertices in the graph, we can now search for a path in this graph. Starting from any vertex, we construct a path by joining as long as possible the vertices with the smallest shift and the highest similarity between the fragments. The path is constructed in both directions: by adding the predecessors and successors of the path. Such a path is constructed for every vertex as the starting point. The longest path is chosen, and all the vertices from the path (and their reverse complementary counterparts) are deleted. If there are still some vertices left in the graph, another paths are constructed, until the length of a path is greater than the *minimum path length*. Now, we are having a set of paths and some spare (or 'not fixed') sequences. The algorithm performs two steps as long as any changes can be made in the set of paths.
Step 1. Check whether it is possible to connect two paths obtained in the previous part of the algorithm. Calculate the alignments between the extreme fragments in

the paths, and, if possible, connect two paths. For example, checking for the possible matching of two paths z and y, for which the beginning and ending sequences are respectively a, b for path z and c, d for path y, requires checking of the alignment of the pairs of sequences: $(b, c), (d, a), (b, \vec{d}), (\vec{c}, a)$. If there is a feasible alignment of sequences:

- (b, c), then join paths z and y
- (d, a), then join paths y and z
- $(b, \vec{d})$, then join paths z and $\vec{y}$
- $(\vec{c}, a)$, then join paths $\vec{y}$ and z

Step 2. Match one of the fragments, which were not included in any paths, to any of the already built paths on both of its sides. If the lengthening of the path with fragment a is possible, then join a with the path, delete a and $\vec{a}$ from the set of unused fragments, and perform Step 1. Otherwise, stop.

The last part of the algorithm consists in reconstruction of the consensus sequence (or the parts of the final sequence, if there is more than one path) from the paths obtained in the previous step. In order to this all the sequences have to be aligned due to the offset which was calculated at the beginning. On every position of the consensus sequence, the fragments, which overlap that position, are voting on the proper nucleotide, and the one (A, C, G or T) with the greatest support wins.

As the result the algorithm returns the consensus sequence or shorter contigs, being the parts of the sequence, which was looked for. Usually, the sequencing errors and the lack of sufficient coverage on every position in the sequence, do not allow to reconstruct the whole sequence. Instead, the solution is composed of contigs which are not overlapping each other. To reconstruct the whole genome usually next phase is required, called *finishing*. It may be based on searching in the database of known genomes, for example in GenBank, for a sequence similar to the searched one and if it is found, the contigs may be put on the proper place. The holes in between the contigs can be read with additional biochemical experiment. If the DNA sequence of an organism is not similar to any known sequences, then extra actions have to be carried out, like marking the places of some characteristic points of the sequence (see: mapping, in Section 2.2) or detecting the distance between some of the fragments from the sequencing phase.

6.2 Computational experiment

The algorithm was tested on the data which come from the real experiment performed with 454 Genome Sequencer in the Lawrence Livermore National Laboratory in Joint Genome Institute. The genome which was recovered in the experiment is the genome of bacteria *Prochlorococcus marinus*. The length of the genome is about 2 M*bp*. The sequencing step in 454 Genome Sequencer resulted in 150 000 sequences of length about 100 nucleotides. Together with the sequences, one gets the quality of every nucleotide in the sequence. The fragments contain sequencing errors and are coming from both strands of DNA helix. Thus, at the beginning, the algorithms added to the input set the reverse and complementary sequences, and the cardinality of the set was doubled. The most often type of sequencing errors is the result of erroneous reading of the intensity signal on the CCD camera, and what follows the erroneous length of uniform nucleotide string, for example, instead of AAAA the sequencer reads it as AAAAA. But insertions, deletions and substitutions also appear in the sequences.

For the computational experiment, smaller sets of input sequences were prepared to compare the behavior of different methods designed for the genome assembling problem. For a large set of fragments most of the methods does not perform correctly, either they did not appear to finish the computations in reasonable time or the result was very miserable. Thus, the cardinalities of the input sets were equal to 500, 1000, 2000, 5000, 10000, 20000, and 50000 sequences. In the cases up to 10000 input fragments, the sequences were coming from one contig. In the latter cases the fragments were composing at most a few contigs. The final sequence was known, it was published in [CAK+06]. Although, it was used only for the comparison of the contigs which were the results of execution of different programs.

The new algorithm was implemented in Ansi C language. The test of all the algorithms were carried out on SUN Fire 6800 in the Poznan Supercomputing and Networking Center. The assembling programs which were used for comparison are widely known, although not all of them have the documentation about the algorithm which was implemented. The compared methods are: PHRAP (http://www.phrap.org/ /index.html), CAP3 [HM99], TIGR (http://www.tigr.org/software/assembler/) and ASM [BKJ+04].

The measures which could compare the methods are the number of contigs ob-

tained by the method, the quality of the largest contig, the coverage, and the time of computations. The *quality* of the output sequence was calculated by the Smith-Waterman algorithm [Wat95, SM97], which searches for the best alignment of two sequences: the input one and the largest obtained contig. This algorithm was described in more details in Section 4.6. The *coverage* is the percentage of the length which was covered by the largest contig. The results are presented in Tables: 6.1 (PHRAP and ASM applications), 6.2 (CAP3 and TIGR applications), and 6.3 (the new algorithm). For the cases where the algorithms resulted in more than one contig, the coverage and quality values are presented for three longest contigs.

Tab. 6.1: Results of the computational experiment for algorithms PHRAP and ASM. The data are coming from a real sequencing experiment. * number of contigs, which are necessary to cover the whole sequence

	PHRAP				**ASM**			
No. of sequences	Quality [%]	Coverage [%]	No. of contigs*	Time [s]	Quality [%]	Coverage [%]	No. of contigs*	Time [s]
500	96.50	92.73	5	1.6	94.43	76.5	8	7.7
	88.85	7.98			84.26	25.73		
	91.91	5.13			97.78	5.34		
1000	97.18	96.40	10	3.4	96.23	96.46	6	31.8
	98.56	6.12			80.38	5.05		
	88.85	3.95			97.92	2.11		
2000	99.66	71.80	14	7.0	97.20	48.86	18	128.0
	99.69	25.28			90.15	26.86		
	50.00	4.12			97.53	23.94		
5000	99.54	20.82	36	17.1	96.82	19.74	28	794.6
	99.61	20.80			97.20	19.79		
	99.56	18.14			97.08	17.92		
10000	99.72	24.11	72	34.3	96.91	17.68	74	3380.9
	99.75	10.92			96.82	9.66		
	99.56	8.87			97.20	9.68		
20000	94.69	11.46	150	78.2	97.42	10.72	147	11783.3
	99.63	7.18			90.83	9.23		
	99.72	6.47			97.04	8.31		

PHRAP and ASM algorithms were tested for the instances with up to 20000 input

Tab. 6.2: Results of the computational experiment for algorithms CAP3 and TIGR. The data are coming from a real sequencing experiment. * number of contigs, which are necessary to cover the whole sequence

	CAP3				**TIGR**			
No. of sequences	Quality [%]	Coverage [%]	No. of contigs*	Time [s]	Quality [%]	Coverage [%]	No. of contigs*	Time [s]
500	99.50	41.42	7	8	98.7	22.34	31	2.6
	99.33	26.44			97.87	16.02		
	88.08	24.9			98.52	14.01		
1000	91.55	27.26	11	17.3	98.56	12.78	52	6.6
	99.93	21.97			99.26	11.91		
	99.50	20.48			98.74	11.05		
2000	99.91	23.86	16	34.5	99.06	12.95	90	13.1
	99.80	15.04			99.26	8.44		
	99.93	11.01			99.21	7.42		
5000	99.91	9.67	33	88.6	99.00	5.68	198	44.4
	99.83	8.8			99.06	5.25		
	99.34	8.59			99.11	4.02		
10000	99.67	6.55	76	179.7	99.00	2.78	394	140.3
	99.70	5.32			99.09	2.57		
	99.91	4.73			99.56	2.34		

sequences. The results of tests for CAP3 and TIGR algorithms are presented here only for the instances with up to 10000 sequences. This is because of poor results – for the largest input set the algorithms gave as output many contigs for which the coverage is lower than 7%. These methods were beaten by ASM, PHRAP and the new algorithm, thus further results are not considered. The new algorithm is still in testing phase and the results presented here are for the instances of size up to 50000 input sequences. However, preliminary tests were performed also for the complete spectrum of all 150000 input fragments, from which one can obtain the whole genome of *Prochlorococcus marinus* bacteria (but divided into contigs). The results seem very promising, but the algorithm still need further research.

Analyzing the results in Tables 6.1-6.3 one may notice very easily, that the new algorithm produces one contig for the instances up to 10000 fragments, which cover

Tab. 6.3: Results of the computational experiment for the new algorithm. The data are coming from a real sequencing experiment. * number of contigs, which are necessary to cover the whole sequence

	The new algorithm			
No. of sequences	Quality [%]	Coverage [%]	No.of contigs*	Time [s]
500	91.93	100	1	2.0
1000	93.49	100	1	5.0
2000	94.29	100	1	10.0
5000	94.83	100	1	31.0
10000	94.89	100	1	113.0
20000	94.20	42	10	746.0
	96.23	20		
	95.03	20		
50000	95.18	10	31	5926.0
	92.56	10		
	97.27	9		

the entire sequence. The quality of the obtained contig is slightly worse than for other methods, and it is around 92%-95%. The lower quality value is caused by imperfect creation of the consensus sequence from the alignment, and can be easily corrected by a group of experts during their finishing work. A part of the algorithm that concerns forming the consensus sequence from the alignment will be soon further developed.

Usually, with the increase of the cardinality of input sets, there are more contigs given as the output of the methods execution. PHRAP was the fastest method for every tested case and gave the contigs of high similarity to the sequence that was looked for. Moreover, for the small input sets, the longest contig of the consensus sequence covered almost whole target sequence (92%-96%). ASM resulted also in long contigs for these sets, with high quality, and for the other sets only slightly worse than for PHRAP program. However, the time of computations exceeded by far any of the tested algorithms. The method, if tested for the instance of whole genome (e.g. genome of bacteria: 150000 sequences), would not finish in reasonable time.

The other methods, CAP3, TIGR, and the new algorithm, work in similar time, but give very different results. TIGR terminated with the outcome of many small contigs. CAP3 produced the contigs of very high quality but much shorter than for

the PHRAP or the new algorithm. The new algorithm outperforms other methods in both: the coverage and the number of generated contigs. On the other hand, the quality is slightly lower than for other ones and still need some improvements.

To conclude, there is no universal method, which is good for every input data. Each of the algorithms, except TIGR, surpass the other methods on one of the aspects. All produce data which are acceptable and need some additional work of experts in finishing stage. However, only the new algorithm could manage to cover the entire sequence, which was searched for (in the case of up to 10000 input fragments). The time consuming, last phase of the algorithm of ordering and joining the contigs did not cause high increase of time – it can be still comparable with PHRAP.

This also proved, that the algorithm can be used in the real case of the DNA assembling problem.

7. CONCLUSIONS

The aim of this thesis was to design new efficient algorithms for the problems of sequencing in small and in large scales, which are time-cost effective and produce results close to optimum. The proposed two algorithms solving the sequencing by hybridization problem with isothermic libraries, namely tabu search and hybrid genetic algorithms, satisfy these conditions. Moreover, the genetic algorithm obtained sequences of high similarity to the target sequences despite of many errors in the spectrum.

The case of repetitions in sequences was considered in more details. It has turned out that isothermic libraries, however diminish the number of experimental errors, increase the errors coming from repetitions. It has been showed, that this type of error is difficult for solving the problem properly. The use of the entire number of proper oligonucleotides in the solution does not assure that the sequence will be properly reconstructed. Although the solutions were of high quality, the similarity of the obtained sequences to the original ones was quite low, for a few runs of the program it was around 60%. The ambiguities, which sequence is correct, can only be resolved with the help of biochemist experts.

Another hybrid genetic algorithm has been designed for the SBH problem with standard (equal-length) libraries. This time the algorithm worked very well in case of both experimental errors and the errors coming from repetitions. For the latter, for more than 75% of tested instances the algorithm returned the original sequence.

The comparison of the approaches to DNA sequencing: the standard one and the isothermic one, has been performed with genetic algorithms based on a similar scheme. Optimal solutions were found with both algorithms, slightly more frequently for standard libraries when repetitions occur in the spectra. For the isothermic libraries there were more possibilities of reconstructing different sequences from the same number of oligonucleotides and this fact resulted in lower average similarity values and higher deviation of the similarity values than for the standard libraries.

In the case of repeated parts in the original sequences, both algorithms had problems in finding original sequences, even though the solutions were optimal – the usage of oligonucleotides was 100%. The genetic algorithm for isothermic libraries obtained the original sequence only for less than a quarter of instances, while the one for standard libraries found the original sequence in 30 out of 40, and 34 out of 40 tested instances, depending on the error rate in the spectra. Worse results for isothermic libraries can be explained by the occurence of shorter oligonucleotides (rich in 'G' and 'C' nucleotides) and what follows higher repetition rate than for the standard libraries. The revised hybrid genetic algorithm for equal-length libraries is the best suited for solving SBH problem, especially for sequences with repetitions and it surpasses by far other algorithms known from the literature.

Novel ideas applied in the genetic algorithms, like the crossover operator based on structured combination, can be also of value for other problems, for which the solution can be represented as a permutation of vertices of a graph. An example of applying the genetic algorithm scheme to the Traveling Salesman Problem has been presented.

And finally, the new approach to sequencing – pyrosequencing – has been studied in details. The data, coming from a real experiment performed on the bacteria Prochlorococcus marinus, on 454 sequencer, were used in the computational tests. The first tests, carried out on a part of the whole genome instance, showed that the proposed algorithm can solve real data, giving as the result one contig of high similarity to the reconstructed sequence.

BIBLIOGRAPHY

[Agi06] Agilent ChIP-on-chip genome-wide location analysis. Technical report, Agilent Company - brochure no.5989-3426EN, http://www.chem.agilent.com/, 2006.

[AH04] A.E. Abbas and S.P. Holmes. Bioinformatics and management science: some common tools and techniques. *Oper. Res.*, 52:165–190, 2004.

[AL97] E.H.L. Aarts and J.K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley and Sons, Chichester, UK, 1997.

[BANZ95] D.E. Bergstrom, P.C. Andrews, R. Nichols, and P. Zhang. 3-nitropyrrole nucleoside. US Patent No. 5,438,131, 08/01/95.

[BF05] J. Błażewicz and P. Formanowicz. Multistage isothermic sequencing by hybridization. *Comput. Biol. Chemistry*, 29:69–77, 2005.

[BFBM86] K.J. Breslauer, R. Frank, H. Blöker, and L.A. Marky. Predicting DNA duplex stability from the base sequence. *Proc. Natl. Acad. Sci. USA*, 83:3746–3750, 1986.

[BFK+99] J. Błażewicz, P. Formanowicz, M. Kasprzak, W.T. Markiewicz, and J. Węglarz. DNA sequencing with positive and negative errors. *J. Comput. Biol.*, 6:113–123, 1999.

[BFK+00] J. Błażewicz, P. Formanowicz, M. Kasprzak, W.T. Markiewicz, and J. Węglarz. Tabu search for DNA sequencing with false negative and false positives. *Euro. J. Oper. Res.*, 125:257–265, 2000.

[BFKM99] J. Błażewicz, P. Formanowicz, M. Kasprzak, and W.T. Markiewicz. Method of sequencing of nucleic acids. Polish Patent Application P335786, 1999.

[BFKM00] J. Błażewicz, P. Formanowicz, M. Kasprzak, and W.T. Markiewicz. Isothermic oligonucleotide libraries. In S. Miyano, R. Shamir, and T. Takagi, editors, *Currents in Computational Molecular Biology*, pages 97–98. Universal Academy Press, Tokyo, 2000.

[BFKM04] J. Błażewicz, P. Formanowicz, M. Kasprzak, and W.T. Markiewicz. Sequencing by hybridization with isothermic oligonucleotide libraries. *Discrete Appl. Math.*, 145:40–51, 2004.

[BFS+04] J. Błażewicz, P. Formanowicz, A. Świercz, M. Kasprzak, and W.T. Markiewicz. Tabu search algorithm for DNA sequencing by hybridization with isothermic libraries. *Comput. Biol. Chemistry*, 28:11–19, 2004.

[BGK04] J. Błażewicz, F. Glover, and M. Kasprzak. DNA sequencing - tabu and scatter search combined. *INFORMS J. Comput.*, 16:232–240, 2004.

[BGK05] J. Błażewicz, F. Glover, and M. Kasprzak. Evolutionary approaches to DNA sequencing with errors. *Annals of Operational Research*, 138:408–415, 2005.

[BGS+06] J. Błażewicz, F. Glover, A. Świercz, M. Kasprzak, W.T. Markiewicz, C. Oğuz, , and D. Rebholz-Schuhmann. Dealing with repetitions in sequencing by hybridization. *Comput. Biol. Chemistry*, 30(5):313–320, 2006.

[BHKW99] J. Błażewicz, A. Hertz, D. Kobler, and D.de Werra. On some properties of DNA graphs. *Discrete Appl. Math.*, 98:1–19, 1999.

[BK03] J. Błażewicz and M. Kasprzak. Complexity of DNA sequencing by hybridization. *Theoretical Comput. Sci.*, 290:1459–1473, 2003.

[BK05] J. Błażewicz and M. Kasprzak. Computational complexity of isothermic DNA sequencing by hybridization. *Discrete Appl. Math.*, 154:718–729, 2005.

[BKJ+04] J. Błażewicz, M. Kasprzak, P. Jackowiak, D. Janny, D. Jarczyński, M. Nalewaj, B. Nowierski, R. Styszynski, L. Szajkowski, and P. Widera.

ASM: DNA assembly application. Technical report, Poznan Supercomputing and Networking Center internal report, no. RA-001/2004, 2004.

[BKK+97] J. Błażewicz, J. Kaczmarek, M. Kasprzak, W.T. Markiewicz, and J. Węglarz. Sequential and parallel algorithms for DNA sequencing. *Comput. Appl. Biosci.*, 13:151–158, 1997.

[BKK02] J. Błażewicz, M. Kasprzak, and W. Kuroczycki. Hybrid genetic algorithm for DNA sequencing with errors. *J. Heuristics*, 8:495–502, 2002.

[Bła88] J. Błażewicz. *Złożoność obliczeniowa problemów kombinatorycznych.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1988.

[BOSW06] J. Błażewicz, C. Oğuz, A. Świercz, and J. Węglarz. DNA sequencing by hybridization via genetic search. *Oper. Res.*, 54(6):1185–1192, 2006.

[BS88] W. Bains and G.C. Smith. A novel method for nucleic acid sequence determination. *J. Theoretical Biology*, 135:303–307, 1988.

[BY04] T.N. Bui and W.A Youssef. An enhanced genetic algorithm for DNA sequencing by hybridization with positive and negative errors. *Lect. Notes Comput. Sci.*, 3103:908–919, 2004.

[CAK+06] F. Chen, J. Alessi, E. Kirton, V. Singan, and P. Richardson. Comparison of 454 sequencing platform with traditional Sanger sequencing: A case study with de novo sequencing of Prochlorococcus marinus NATL2A genome. In *Plant and Animal Genomes Conference*. 2006.

[CL86] G. Chartrand and L. Lesniak, editors. *Graphs & Digraphs.* Wadsworth & Brooks /Cole, Pacific Grove, CA, 1986.

[CLRS01] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, editors. *Introduction to Algorithms, Second Edition.* MIT Press, Cambridge, MA, 2001.

[CPT04] M. Chaisson, P. Pevzner, and H. Tang. Fragment assembly with short reads. *Bioinformatics*, 20(13):2067–2074, 2004.

[End04] T.A. Endo. Probabilistic nucleotide assembling method for sequencing by hybridization. *Bioinformatics*, 20:2181–2188, 2004.

[FC03] G.B. Fogel and D.W. Corne, editors. *Evolutionary Computations in Bioinformatics*. Morgan Kaufman, San Francisco, CA, 2003.

[FRP+91] S.P.A. Fodor, J.L. Read, M.C. Pirrung, L. Stryer, A. Lu, and D. Solas. Light-directed spatially addressable parallel chemical synthesis. *Science*, 251:767–773, 1991.

[GJ79] M. R. Garey and D.S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness*. Freeman, San Francisco, 1979.

[GL93] F. Glover and M. Laguna. Tabu search. In C.R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[GL97] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, MA, 1997.

[Glo94] F. Glover. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Appl. Math.*, 49:231–255, 1994.

[Gus97] D. Gusfield. *Algorithms on Strings, Trees and Sequences. Computer Science and Computational Biology*. Cambridge University Press, New York, NY, 1997.

[HGP03] Human Genome Project. Genomics and its impact on science and society: A 2003 primer. Technical report, U.S. Department of Energy, http://www.ornl.gov/sci/techresources/HumanGenome/publicat/ /primer2001/index.shtml, 2003.

[HGR] Technical report, National Human Genome Research Institute, http://www.genome.gov/Education/.

[HHHS03] E. Halperin, S. Halperin, T. Hartman, and R. Shamir. Handling long targets and errors in sequencing by hybridization. *J. Comput. Biol.*, 10:483–497, 2003.

[HM99] X. Huang and A. Madan. CAP3: A DNA sequence assembly program. *Genome Res.*, 9:868–877, 1999.

[Hol75] H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, MI, 1975.

[HPY03] S.A. Heath, F.P. Preparata, and J. Young. Sequencing by hybridization by cooperating direct and reverse spectra. *J. Comput. Biol.*, 10:499–508, 2003.

[IW95] R. Idury and M. Waterman. A new algorithm for DNA sequence assembly. *J. Comput. Biol.*, 2:291–306, 1995.

[JL96] T. Jiang and M. Li. DNA sequencing and string learning. *Mathematical Systems Theory*, 29:387–405, 1996.

[KM95] J.D. Kececioglu and E. W. Myers. Combinatorial algorithms for DNA sequence assembly. *Algorithimica*, 13:7–51, 1995.

[Kru98] S. Kruglyak. Multistage sequencing by hybridization. *J. Comput. Biol.*, 5:165–171, 1998.

[LB94] D. Loakes and D.M. Brown. 5-nitroindole as an universal base analogue. *Nucl. Acids Res.*, 22:4039–4043, 1994.

[LFK+88] Y.P. Lysov, V.L. Florentiev, A.A Khorlin, K.R. Khrapko, V.V Shik, and A.D. Mirzabekov. Determination of the nucleotide sequence of DNA using hybridization of oligonucleotides. A new method. *Dokl. Akademii Nauk SSSR*, 303:1508–1511, 1988.

[MEA+05] M. Margulies, M. Egholm, W.E. Altman, S. Attiya, et al. Genome sequencing in microfabricated high density picolitre reactors. *Nature*, 437:376–380, 2005.

[MG77] A.M. Maxam and W. Gilbert. A new method for sequencing DNA. *Proc. Natl. Acad. Sci. USA*, 74:560–564, 1977.

[Mic96] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* 3rd rev. and extended ed. Springer-Verlag, Berlin, Germany, 1996.

[MLB+96] G. McGall, J. Labadie, P. Brock, G. Wallraff, T. Nguyen, and W. Hinsberg. Light-directed synthesis of high-density oligonucleotide arrays using semiconductor photoresists. *Proc. Natl. Acad. Sci. USA*, 96:13555–13560, 1996.

[NW70] S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities of the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[Pev89] P.A. Pevzner. l-tuple DNA sequencing: computer analysis. *Journal of Biomolecular Structure and Dynamics*, 7:63–73, 1989.

[Pev00] P.A. Pevzner. *Computational Molecular Biology: An Algorithmic Approach*. MIT Press, Cambridge, MA, 2000.

[PFU99] F.P. Preparata, A.M. Frieze, and E. Upfal. On the power of universal bases in sequencing by hybridization. In *Proc. 3rd Ann. Int. Conf. Comput. Mol. Biol.*, pages 295–301, 1999.

[PO04] F.P. Preparata and J.S. Oliver. DNA sequencing by hybridization using semi-degenerate bases. *J. Comput. Biol.*, 11(4):753–765, 2004.

[PS00] P.V. Parthasarathy and R. Sudarshan. *GALiLEO Theory Manual*. Department of Civil Engineering, Indian Institute of Technology Madras, Chennai, 2000.

[PS01] V.T. Phan and S. Skiena. Dealing with errors in interactive sequencing by hybridization. *Bioinformatics*, 17:862–870, 2001.

[PTW01] P.A. Pevzner, H. Tang, and M.S. Waterman. A new approach to fragment assembly in DNA sequencing. In *Proc. 5th Ann. Inter. Conf. Res. Comput. Molecular Biology (RECOMB)*, pages 256–267, Montreal, 2001. ACM Press.

[PU01] F.P. Preparata and E. Upfal. System and methods for sequencing by hybridization. United States Patent Application US 2001/0004728, 21/07/2001.

[RKP+96] M. Ronaghi, S. Karamohamed, B. Pettersson, M. Uhlen, and P. Nyren. Real-time DNA sequencing using detection of pyrophosphate release. *Anal. Biochem.*, 242(1):84–89, 1996.

[SK95] S. Schulze-Kremer. *Molecular Bioinformatics: Algorithms and Applications*. de Gruyter, Berlin, Germany, 1995.

[SM97] J. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, MA, 1997.

[SNC77] F. Sanger, S. Nickelen, and A.R. Coulson. DNA sequencing with chain-terminating inhibitors. *Proc. Natl. Acad. Sci. USA*, 74:560–564, 1977.

[Sou88] E.M. Southern. United Kingdom Patent Application GB8810400, 1988.

[Swi02] A. Swiercz. *Algorytmy przybliżone sekwencjonowania łańcuchów DNA z wykorzystaniem bibliotek izotermicznych*. Master Thesis, 2002.

[VMOR98] S. Voss, S. Martello, I. Osman, and C. Roucairol, editors. *Meta-Heuristics - Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, Boston, MA, 1998.

[Wat95] M.S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes*. Chapman & Hall, London,UK, 1995.

[WJH+81] R.B. Wallace, M.J. Johnson, T. Hirose, T. Miyake, E.H. Kawashima, and K. Itakura. The use of synthetic oligonucleotides as hybridization probes. II. Hybridization of oligonucleotides of mixed sequence to rabbit beta-globin DNA. *Nucleic Acids Res.*, 9(4):879–894, 1981.

[ZWZ03] J-H. Zhang, L-Y. Wu, and X-S. Zhang. Reconstruction of DNA sequencing by hybridization. *Bioinformatics*, 19:14–21, 2003.

A. APPENDIX. BIO-PORTAL

A WWW bio-portal was created and you may find it at: http://bio.cs.put.poznan.pl/.

You may find there some details about the sequencing by hybridization approach (the main page "SBH"), different libraries of oligonucleotides which may be used in the biochemical experiment (page "library of oligonucleotides"), and the list of articles where this issue was considered in more details by the bioinformatics group from the Institute of Computer Science at Poznań University of Technology (page "literature").

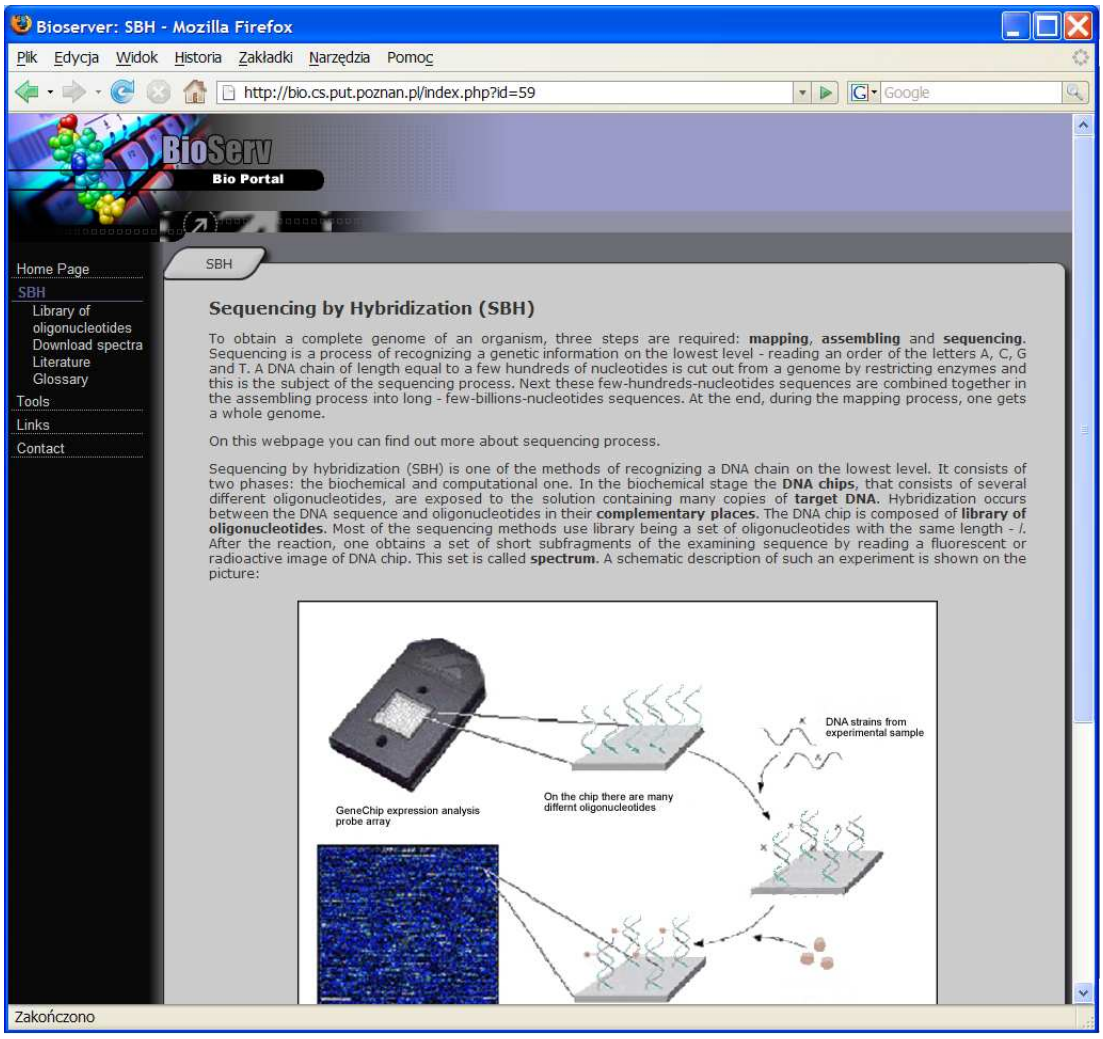
On the main page one can read about the details of the hybridization experiment. The standard approach is described, which uses the libraries of oligonucleotides of the same length l. Also the computational complexity of the SBH problem is specified there, for the case when in the input set of oligonucleotides (spectrum) there are no errors, and in the case of positive and negative errors in the spectrum (see Figure A.1).

From the page "Download spectra" (see Figure A.2) one may obtain spectra from the computational experiment done within this work. The spectra were divided into two parts (Sections: "Spectra derived from real DNA sequences for standard and isothermic sequencing" and "Spectra for standard DNA sequencing derived from real and random DNA sequences"). The first group of spectra was prepared from real DNA sequences, taken from GenBank (the accession numbers of these sequences are placed in Appendix B). Some of the sequences did not contain repetitions of oligonucleotides both for standard and for isothermic libraries (Set A), and the others did (Set B), (see also Figure 4.4). The instances, created on the basis of the sequences were used in the computational experiments from [BFS$^+$04, BOSW06, BGS$^+$06].

The second group of spectra ("Spectra for standard DNA sequencing derived from real and random DNA sequences") was prepared from real DNA sequences as well as from random sequences, generated according to the uniform distribution. The sequences also could contain repetitions of oligonucleotides. The results of computational experiments for these instances were presented in [BKK02, BGK04, BGK05].

All of the sequences were composed of at least 600 nucleotides. The prefixes of the sequences are of length 100, 200, 300, 400, 500 and 600, and are called original sequences (the length of the prefixes differs for two groups of spectra). From these original sequences

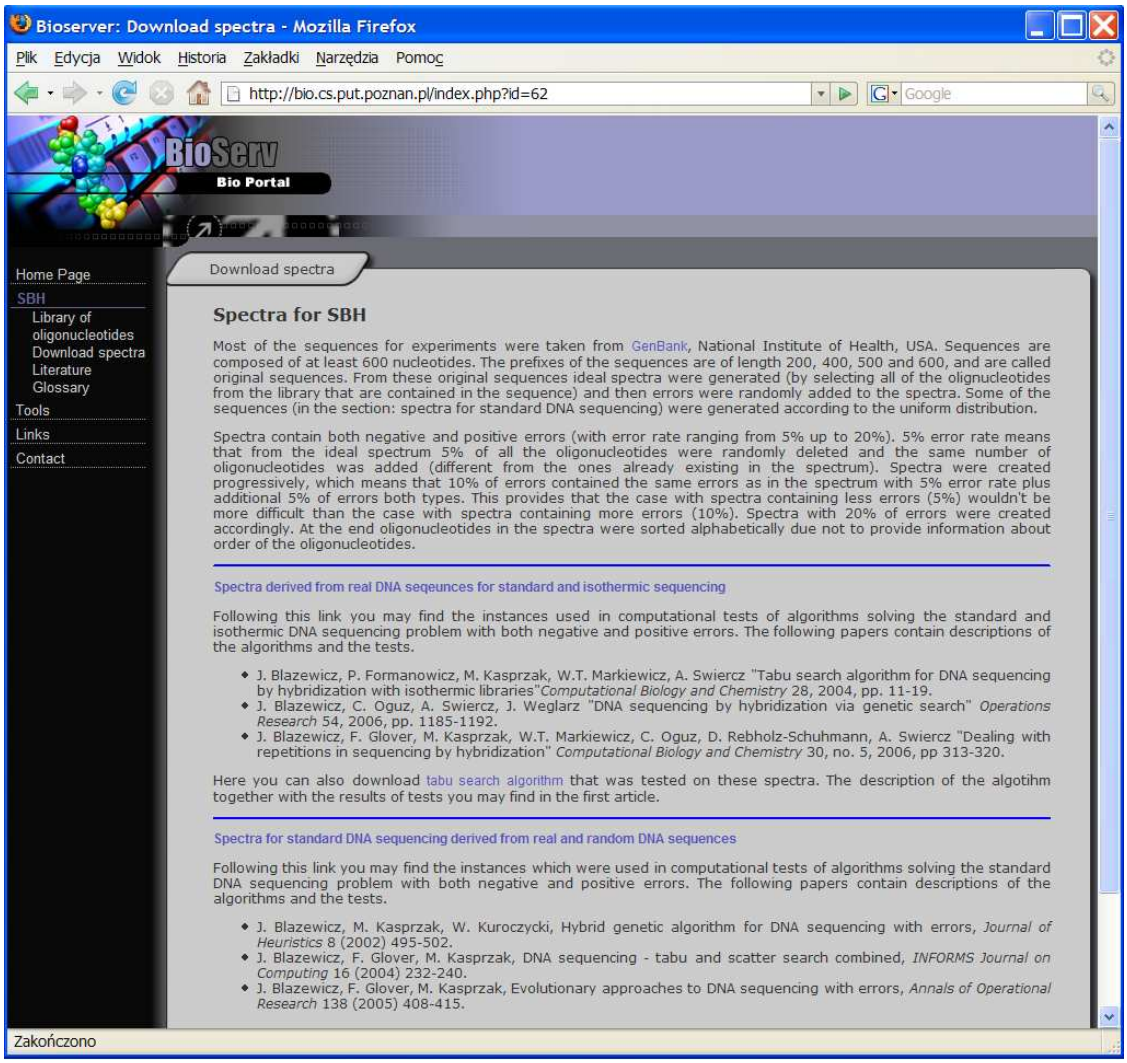
Fig. A.1: A screenshot of the site on the bio-portal entitled: "Sequencing by hybridization (SBH)".



ideal spectra were generated (by selecting all of the oligonucleotides from the library that are contained in the sequence) and then errors were randomly added to the spectra.

Spectra contain both negative and positive errors (with error rate ranging from 5% up to 20%). 5% error rate means that from the ideal spectrum 5% of all the oligonucleotides were randomly deleted and the same number of oligonucleotides was added (different from the ones already existing in the spectrum). Spectra were created progressively, which means that the one with 10% of errors contained the same errors as the spectrum with 5% error rate, plus additional 5% of errors of both types. This provides that the case with spectra containing less errors (5%) would not be more difficult than the case with spectra containing more errors (10%). Spectra with 20% of errors were created accordingly.

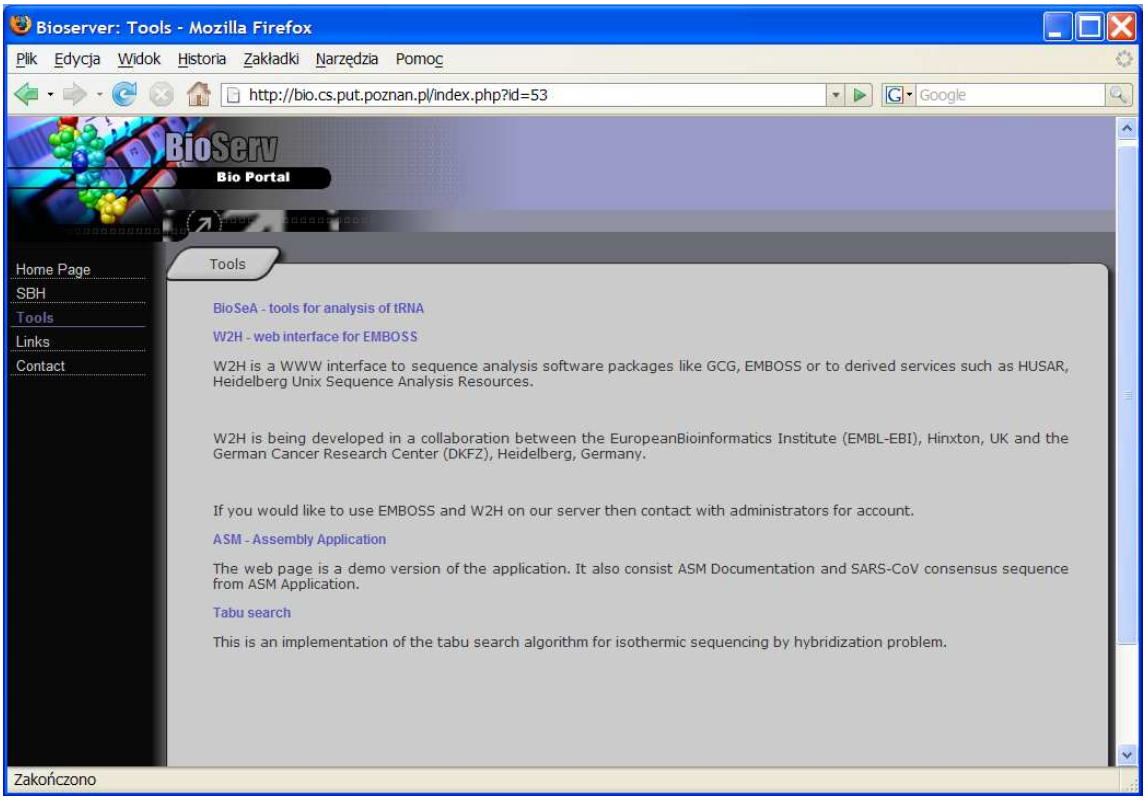
Fig. A.2: A screenshot of the site on the bio-portal entitled: "Download spectra".



At the end oligonucleotides in the spectra were sorted alphabetically due to not provide information about order of the oligonucleotides.

At the WWW bio-portal you may also find some interesting tools, which were developed in the bioinformatics group from the Institute of Computer Science, and can be downloaded from the site. Among others, there are two programs related to the current work, the implemented tabu search algorithm for SBH problem presented in this work, and ASM application, which was one of the programs tested for comparison with the new algorithm for the assembling problem. A screenshot of the page you may find in Figure A.3.

Fig. A.3: A screeshot of the site on the bio-portal entitled: "Tools".



B. APPENDIX. ACCESSION NUMBERS OF SEQUENCES USED IN TESTS.

Accession numbers of the sequences used for computational tests. Set A – sequences have no errors coming from repetitions.
NM_016055, NM_016080, NM_152373, NM_002938, BC040844, NM_152763, NG_002692, BC044213, NG_002660, NG_002481, BC007770, BC015575, BC004538, BC063108, NG_002361, NG_001569, BC056270, NM_153834, HSA519841, BC053904, BC062471, BC062325, BC057825, NG_001151, NG_001292, NM_005337, NM_032293, NM_032292, NM_198197, NM_032423, NM_021807, NM_015435, NM_024622, NM_030633, NM_172366, NM_177959, NM_005337, NM_003318, AF435957, AF497481

Accession numbers of the sequences used for computational tests. Set B – sequences contain errors coming from repetitions.
NM_002052, NM_003008, NM_183353, BC008923, BC012982, NG_002363, BC009854, NM_194310, NM_006402, NM_194300, NM_020690, NM_005745, BC000774, NM_182498, BC047640, NM_004902, NM_020713, BC062620, NM_032866, NG_000980, BC026171, NM_000321, BC063041, BC005805, NM_144767, BC001077, NM_032349, BC053852, NM_021934, NM_015698, BC050425, NM_018046, NM_004663, BC007398, NM_016428, NM_177423, BC026078, NM_031205, BC041372, NM_033318

C. APPENDIX. ADDITIONAL COMPUTATIONAL RESULTS

More detailed results of the computational experiment of the genetic algorithm for isothermic and standard libraries are presented in this appendix. The length of tested sequences varies from 200 to 600 nucleotides and the error rate in the spectrum is equal to $\pm 5\%$ and $\pm 20\%$. The algorithm was tested with 40 instances for every case, 10 times for each spectrum. The results in the tables in the second and in the third columns are the average values of 10 runs for each instance. The values in the column 'usage of oligonucleotides' say how many proper oligonucleotides are used to compose the solution. 100% means that the number of oligonucleotides in the solution is the same as the cardinality of the spectrum with negative errors only. The 'similarity' value is counted according to the Needleman-Wunsh algorithm and is described in Section 4.6 in more details. The value in the column 'optimal usage' means how many instances were solved with 100% of usage of oligonucleotides out of 10 tested instances, while in the column 'optimal similarity' there is the number of instances solved with 100% of similarity. In the last row of the tables the average values of the usage of oligonucleotides and the similarity are presented.

The results of the tests are discussed widely in Sections 4.6 and 5.2.

Tab. C.1: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 200 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	86,35	56,87	10/10	0/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	94,75	11,81	10/10	3/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**99,53**	**1,72**		

Tab. C.2: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 400 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	95,47	11,90	10/10	0/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	85,19	219,48	10/10	5/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	98,13	3,52	10/10	5/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**99,47**	**5,87**		

Tab. C.3: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 500 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	93,15	0,42	10/10	0/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	99,94	0,01	10/10	7/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	92,89	117,95	10/10	7/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	98,20	2,16	10/10	4/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**99,60**	**3,01**		

Tab. C.4: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 600 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	93,82	0,20	10/10	0/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	96,10	22,82	10/10	6/10
11	100,00	92,10	93,62	10/10	6/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	82,25	315,06	10/10	5/10
20	100,00	98,25	1,31	10/10	3/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	77,88	489,52	10/10	5/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	98,35	24,50	10/10	9/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**98,47**	**23,68**		

Tab. C.5: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 200 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	83,20	1,41	10/10	0/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	86,00	0,00	10/10	0/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	97,00	13,50	10/10	6/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	99,90	99,60	0,64	8/10	8/10
average	**100,00**	**99,15**	**0,39**		

Tab. C.6: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 400 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	99,65	98,80	6,39	6/10	6/10
2	100,00	99,70	0,36	10/10	8/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	95,16	18,81	10/10	1/10
5	100,00	100,00	0,00	10/10	10/10
6	99,56	96,25	76,89	7/10	7/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	82,22	210,70	10/10	4/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	99,25	0,00	10/10	0/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	98,13	3,52	10/10	5/10
21	99,71	91,68	149,56	4/10	4/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	99,40	96,75	10,61	5/10	5/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	99,90	99,87	0,06	8/10	8/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,24	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**99,96**	**98,95**	**11,92**		

Tab. C.7: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 500 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	79,60	92,16	10/10	0/10
4	99,75	93,22	2,72	5/10	0/10
5	99,34	92,79	102,40	2/10	2/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	99,92	0,01	10/10	6/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	95,26	89,87	10/10	8/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	98,20	2,16	10/10	4/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	99,96	99,07	7,78	9/10	9/10
33	100,00	99,70	0,00	10/10	0/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	99,98	97,75	45,56	9/10	9/10
40	100,00	100,00	0,00	10/10	10/10
average	**99,98**	**98,89**	**8,57**		

Tab. C.8: Results of the computational experiment with the genetic algorithm for isothermic spectra coming from sequences of length 600 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	99,94	99,90	0,04	8/10	8/10
2	99,11	77,75	160,01	2/10	0/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	95,69	1,67	10/10	0/10
5	100,00	100,00	0,00	10/10	10/10
6	99,37	98,70	1,13	4/10	4/10
7	100,00	100,00	0,00	10/10	10/10
8	99,81	77,71	154,09	6/10	2/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	92,20	15,21	10/10	2/10
11	100,00	90,13	97,52	10/10	5/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	82,25	315,06	10/10	5/10
20	100,00	97,55	11,52	10/10	4/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	99,66	60,96	177,22	2/10	0/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	90,10	65,34	10/10	4/10
34	99,95	100,00	0,00	9/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	99,20	80,21	401,77	5/10	5/10
average	**99,93**	**96,08**	**35,01**		

Tab. C.9: Results of the computational experiment with the genetic algorithm for isothermic spectra, which come from sequences of length 600 with repetitions

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	64,66	106,23	10/10	0/10
2	100,00	74,05	102,65	10/10	1/10
3	100,00	94,52	46,01	10/10	0/10
4	100,00	95,43	59,37	10/10	7/10
5	100,00	69,30	13,62	10/10	0/10
6	100,00	91,46	70,14	10/10	0/10
7	100,00	68,91	93,92	10/10	0/10
8	99,97	72,91	24,17	9/10	0/10
9	100,00	74,21	122,30	10/10	1/10
10	100,00	74,46	8,92	10/10	0/10
11	100,00	75,88	105,34	10/10	0/10
12	100,00	82,44	76,84	10/10	0/10
13	100,00	100,00	0,00	10/10	10/10
14	99,94	64,12	15,91	6/10	0/10
15	100,00	99,66	0,00	10/10	0/10
16	100,00	85,30	27,74	10/10	0/10
17	100,00	62,87	15,42	10/10	0/10
18	99,91	99,60	0,04	2/10	2/10
19	99,90	91,30	39,07	10/10	0/10
20	100,00	99,66	0,00	10/10	0/10
21	100,00	85,75	135,38	10/10	4/10
22	100,00	79,98	54,78	10/10	0/10
23	100,00	85,95	131,64	10/10	4/10
24	100,00	61,74	40,38	10/10	0/10
25	99,88	97,38	6,55	0/10	0/10
26	100,00	67,67	12,11	10/10	0/10
27	100,00	68,50	22,78	10/10	0/10
28	100,00	71,76	27,38	10/10	0/10
29	100,00	66,76	43,58	10/10	0/10
30	99,96	73,36	24,38	10/10	0/10
31	100,00	81,18	104,51	10/10	0/10
32	100,00	66,29	10,68	10/10	0/10
33	100,00	83,35	192,13	10/10	3/10
34	100,00	76,18	284,03	10/10	3/10
35	100,00	71,61	36,76	10/10	0/10
36	100,00	63,17	128,70	10/10	0/10
37	100,00	76,39	22,99	10/10	0/10
38	100,00	68,77	15,87	10/10	0/10
39	100,00	63,03	31,37	10/10	0/10
40	100,00	75,27	121,65	10/10	0/10
average	**99,99**	**78,12**	**59,38**		

Tab. C.10: Results of the computational experiment with the genetic algorithm for isothermic spectra, which come from sequences of length 600 with repetitions, with additional positive and negative errors up to ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	99,97	79,92	93,13	9/10	0/10
2	100,00	83,98	216,61	10/10	2/10
3	100,00	91,99	54,74	10/10	0/10
4	100,00	85,80	134,46	10/10	4/10
5	100,00	66,13	12,71	10/10	0/10
6	100,00	85,04	25,61	10/10	0/10
7	100,00	66,77	45,17	10/10	0/10
8	99,56	69,48	36,47	0/10	0/10
9	100,00	72,39	84,51	10/10	0/10
10	99,99	73,68	15,37	9/10	0/10
11	100,32	66,50	120,11	10/10	0/10
12	100,00	78,56	29,87	10/10	0/10
13	100,00	100,00	0,00	10/10	10/10
14	99,81	63,09	20,43	0/10	0/10
15	100,00	99,00	0,00	10/10	0/10
16	100,34	83,99	138,77	10/10	0/10
17	100,29	67,35	73,60	10/10	0/10
18	99,88	99,50	0,00	0/10	0/10
19	100,00	98,50	0,00	10/10	0/10
20	100,00	99,66	0,00	10/10	0/10
21	100,00	88,13	141,02	10/10	5/10
22	100,26	92,50	0,86	10/10	0/10
23	100,00	85,95	131,64	10/10	4/10
24	100,00	65,36	44,62	10/10	0/10
25	99,87	96,88	6,89	0/10	0/10
26	99,91	68,47	11,98	6/10	0/10
27	99,91	67,20	20,71	6/10	0/10
28	100,00	69,77	54,02	10/10	0/10
29	99,94	76,31	124,29	10/10	0/10
30	99,86	73,08	17,37	0/10	0/10
31	99,98	76,45	182,55	8/10	0/10
32	100,00	67,06	15,86	10/10	0/10
33	100,00	72,81	50,68	10/10	0/10
34	100,00	80,40	289,31	10/10	4/10
35	100,00	68,27	42,73	10/10	0/10
36	100,14	66,34	113,37	10/10	0/10
37	100,00	76,10	56,85	10/10	0/10
38	100,00	70,03	10,65	10/10	0/10
39	100,14	63,19	30,64	10/10	0/10
40	100,00	82,02	18,28	10/10	0/10
average	**100,00**	**78,44**	**61,65**		

Tab. C.11: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 200 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**100,00**	**0,00**		

Tab. C.12: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 400 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**100,00**	**0,00**		

Tab. C.13: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 500 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	94,06	23,52	10/10	4/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**99,85**	**0,59**		

Tab. C.14: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 600 and the error rate ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	99,88	0,01	10/10	3/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	95,05	16,34	10/10	4/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**99,87**	**0,41**		

Tab. C.15: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 200 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	73,45	470,61	10/10	4/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**99,34**	**11,77**		

Tab. C.16: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 400 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	100,00	0,00	10/10	10/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	100,00	0,00	10/10	10/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**100,00**	**0,00**		

Tab. C.17: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 500 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	100,00	0,00	10/10	10/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	92,08	15,68	10/10	2/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	100,00	0,00	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	100,00	100,00	0,00	10/10	10/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	99,80	0,00	10/10	0/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**100,00**	**99,80**	**0,39**		

Tab. C.18: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 600 and the error rate ±20%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	100,00	0,00	10/10	10/10
3	100,00	100,00	0,00	10/10	10/10
4	100,00	100,00	0,00	10/10	10/10
5	100,00	100,00	0,00	10/10	10/10
6	100,00	99,90	0,01	10/10	4/10
7	100,00	100,00	0,00	10/10	10/10
8	100,00	96,70	16,34	10/10	6/10
9	100,00	100,00	0,00	10/10	10/10
10	100,00	100,00	0,00	10/10	10/10
11	99,94	99,31	4,31	9/10	9/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	100,00	0,00	10/10	10/10
14	100,00	100,00	0,00	10/10	10/10
15	100,00	100,00	0,00	10/10	10/10
16	99,43	99,62	0,02	1/10	1/10
17	100,00	100,00	0,00	10/10	10/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	100,00	0,00	10/10	10/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	100,00	0,00	10/10	10/10
23	100,00	100,00	0,00	10/10	10/10
24	99,94	97,18	71,83	9/10	9/10
25	100,00	99,83	0,00	10/10	0/10
26	100,00	100,00	0,00	10/10	10/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	100,00	0,00	10/10	10/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	100,00	0,00	10/10	10/10
32	100,00	100,00	0,00	10/10	10/10
33	100,00	100,00	0,00	10/10	10/10
34	100,00	100,00	0,00	10/10	10/10
35	100,00	100,00	0,00	10/10	10/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	100,00	0,00	10/10	10/10
38	100,00	100,00	0,00	10/10	10/10
39	100,00	100,00	0,00	10/10	10/10
40	100,00	100,00	0,00	10/10	10/10
average	**99,98**	**99,81**	**2,31**		

Tab. C.19: Results of the computational experiment with the genetic algorithm for equal-length spectra coming from sequences of length 600 with repetitions

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	87,55	91,94	10/10	3/10
3	100,00	91,29	75,86	10/10	5/10
4	100,00	87,03	75,16	10/10	2/10
5	100,00	85,70	136,40	10/10	4/10
6	100,00	91,46	173,94	10/10	0/10
7	100,00	85,00	150,00	10/10	4/10
8	100,00	90,54	89,49	10/10	5/10
9	100,00	94,85	106,09	10/10	8/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	91,77	55,39	10/10	10/10
12	100,00	100,00	0,00	10/10	10/10
13	100,00	84,76	91,35	10/10	1/10
14	100,00	89,50	110,25	10/10	5/10
15	100,00	93,72	65,15	10/10	0/10
16	100,00	93,80	6,44	10/10	0/10
17	100,00	90,17	81,38	10/10	0/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,00	94,05	167,86	10/10	0/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	97,18	7,20	10/10	4/10
23	100,00	85,95	131,64	10/10	4/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	94,56	16,05	10/10	3/10
26	100,00	84,75	232,56	10/10	5/10
27	100,00	100,00	0,00	10/10	10/10
28	100,00	90,92	82,54	10/10	5/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	100,00	75,46	120,63	10/10	10/10
32	100,00	92,18	26,20	10/10	3/10
33	100,00	95,52	80,43	10/10	8/10
34	100,00	73,67	301,09	10/10	3/10
35	100,00	97,70	21,16	10/10	8/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	93,07	192,32	10/10	8/10
38	100,00	87,63	153,14	10/10	5/10
39	100,00	98,55	0,29	10/10	1/10
40	100,00	91,87	63,19	10/10	0/10
average	**100,00**	**92,75**	**72,63**		

Tab. C.20: Results of the computational experiment with the genetic algorithm for equal-length spectra, coming from sequences of length 600 with repetitions, with additional positive and negative errors up to ±5%

seq. no.	usage of oligonucleotides [%]	similarity [%]	similarity deviation	optimal usage	optimal similarity
1	100,00	100,00	0,00	10/10	10/10
2	100,00	70,60	79,44	10/10	0/10
3	100,00	93,03	72,83	10/10	6/10
4	100,00	82,82	52,95	10/10	1/10
5	100,00	85,70	136,40	10/10	4/10
6	100,00	94,12	172,98	10/10	0/10
7	100,00	80,00	100,00	10/10	2/10
8	100,00	90,54	89,49	10/10	5/10
9	100,00	84,55	159,14	10/10	4/10
10	100,00	100,00	0,00	10/10	10/10
11	100,00	94,64	46,59	10/10	0/10
12	100,00	94,50	3,45	10/10	1/10
13	100,00	87,37	104,07	10/10	2/10
14	100,00	93,70	92,61	10/10	7/10
15	100,00	96,33	25,01	10/10	0/10
16	100,34	96,88	0,09	10/10	0/10
17	100,00	92,36	69,36	10/10	0/10
18	100,00	100,00	0,00	10/10	10/10
19	100,00	100,00	0,00	10/10	10/10
20	100,18	98,25	0,00	10/10	0/10
21	100,00	100,00	0,00	10/10	10/10
22	100,00	97,40	6,79	10/10	5/10
23	100,00	90,63	131,64	10/10	6/10
24	100,00	100,00	0,00	10/10	10/10
25	100,00	95,00	14,07	10/10	2/10
26	100,00	81,70	223,26	10/10	4/10
27	99,86	96,12	47,97	6/10	6/10
28	100,00	85,46	52,82	10/10	2/10
29	100,00	100,00	0,00	10/10	10/10
30	100,00	100,00	0,00	10/10	10/10
31	99,77	73,84	190,93	5/10	1/10
32	100,00	93,30	29,94	10/10	4/10
33	100,00	88,79	125,66	10/10	5/10
34	100,00	61,15	5,61	10/10	0/10
35	100,00	89,65	11,90	10/10	1/10
36	100,00	100,00	0,00	10/10	10/10
37	100,00	82,67	300,50	10/10	5/10
38	100,00	90,10	147,02	10/10	6/10
39	100,00	98,79	0,22	10/10	0/10
40	100,00	89,16	67,01	10/10	0/10
average	**100,00**	**91,23**	**63,99**		