Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 161 (2005) 585–597
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Invited Review

Selected combinatorial problems of computational biology

Jacek Błażewicz *,[1], Piotr Formanowicz, Marta Kasprzak

Institute of Computing Science, Poznań University of Technology, ul. Piotrowo 3A, 60-965 Poznań, Poland
Institute of Bioorganic Chemistry, Polish Academy of Sciences, Noskowskiego 12/14, 61-704 Poznań, Poland

Received 9 April 2003; accepted 3 October 2003
Available online 14 May 2004

Abstract

Recently we observe a great breakthrough in biology connected with the studies on genomes. These achievements would be impossible without an input from other sciences, combinatorial optimization being one of them. This study is devoted to a presentation of the most important (to our opinion) area of the computational biology, mostly connected with DNA studies, where combinatorial optimization impact was clearly visible. They include: sequencing DNA chains, assembling them, genome mapping and sequence comparison.

Keywords: DNA sequencing by hybridization; Sequence assembling; Genome mapping; Sequence alignment

1. Introduction

In recent years several spectacular events connected with genome studies have occurred, reading the human genome being one of them. Biological sciences got a great impact on many aspects of the everyday life. Since a discovery by Watson and Crick their double helix model of a DNA chain [40], biology has made a great progress in understanding foundations of life. The progress would have not been possible, however, without a help from other areas of science. Here, let us mention physics, chemistry and last but not least computing sciences. We will concentrate on the impact on biology of the latter science, especially its part connected with *operational research* and *combinatorial optimization*. Since ties between biology and combinatorial optimization may be found in very many fields of biology, we will concentrate on the most important (to our opinion) examples of the application of combinatorial optimization methodology in modeling and solving problems arising in the context of the computational biology. They include: sequencing DNA chains, assembling them, genome mapping and sequence comparison. The aim of the paper was to present these problems in a way that is interesting to those working in the field, as well as non-specialists which may

* Corresponding author. Address: Institute of Computing Science, Poznań University of Technology, ul. Piotrowo 3A, 60-965 Poznań, Poland.
E-mail address: blazewic@put.poznan.pl (J. Błażewicz).
[1] The paper was written while the first author was at Laboratoire Leibniz, Universite Joseph Fourier, whose help is greately appreciated.

0377-2217/$ - see front matter
doi:10.1016/j.ejor.2003.10.054

become interested by the new exciting applications of operational research. Thus, an up to date review of research in the field is complemented by many examples allowing one for a better understanding of the concepts introduced. An interested reader is referred also to the recent monographs, which describe the above and other problems in a greater detail: [18,20,30,33,39].

The organization of the paper is as follows. Section 2 contains a biological primer. Section 3 discusses the DNA sequencing problem, while Section 4 presents the assembling one. Genome mapping and sequence comparison are considered, respectively, in Sections 5 and 6.

2. Biological primer

Biology is a science that provides an understanding of the nature of all living things at different levels – from molecules to cells, individuals and populations. It is accepted that all living organisms are composed either of a single cell or a collection of them. At this stage of development a primary interest of biology lies in molecules and cells, thus, we have molecular biology. With the aid of computing science models and tools (combinatorial optimization being one of its main components) it creates so called computational molecular biology.

One of the main objects of the study of computational biology are *DNA chains*, coding genetic information of living organisms. DNA is a *string* composed of letters (*nucleotides*) being members of the alphabet {A, C, G, T}. Short single-stranded DNA molecules are called *oligonucleotides*. The entire DNA of the organism is called its *genome* and its length may reach billions of nucleotides (or base pairs). The same genome is contained in each cell of a given living organism (e.g. human beings have trillions of cells). DNA appears in a form of a *double helix*, i.e. a double strand, where A in one chain can be bound to T only, and C to G, respectively. This fundamental law of a DNA construction was discovered by Watson and Crick [40]. Knowing, thus, one strand of a DNA helix, the second (*complementary*) can be easily reconstructed. What is more, this property of a single strand (trying to bind to a complementary strand) called a *hybridization*, may be used in laboratories in many processes leading e.g. to a reconstruction of an unknown chain.

The genetic information contained in DNA is then used, as stated by the Central Dogma of Molecular Biology, to produce *RNA* and ultimately *proteins*. The latter are the main construction material of living organisms, deciding of their functioning as well.

In the following, we will concentrate on DNA analysis and reconstruction, since these processes require a considerable input from the combinatorial optimization side. We will consider combinatorial problems connected with DNA reading, sequence comparison and phylogenetic analysis.

Reading sequences of genomic DNA is usually a starting point for further molecular biology research. But reading DNA sequences itself is not a trivial task and may involve some sophisticated procedures. It follows from the fact that it is not possible to read a sequence of nucleotides directly, e.g. using a microscope. Hence, some indirect methods have to be used. The process of reading the sequence of a genome is usually divided into three stages: *mapping*, *assembling* and *sequencing*.

The process of reading a long piece of DNA usually starts with cutting it into smaller pieces of size about 100 000–1 000 000 nucleotides. (Let us note that biological experiments take effects usually on millions of copies of given types of molecules.) During the cutting process the information about the order of these pieces is lost. But, as one can guess, it is necessary to recover this information and this is done by mapping procedures. When mapped, the fragment is picked up and cut into much smaller parts of length about 40 000 nucleotides. Again, mapping is necessary to recover the order of the pieces obtained by the second-level cutting [20].

Since sequencing methods allow determining sequences of lengths not greater than 1000 nucleotides, fragments of lengths about 40 000 base pairs cannot be directly sequenced. So, it is necessary to break them into pieces of appropriate lengths. This is done randomly. At this stage one gets DNA fragments suitable for *sequencing*. After sequencing the short fragments are tried to be

ordered in such a way that they form the initial fragment of length about 40 000 in an *assembling process*.

At this moment let us point here to the very important problem of *sequence comparison* (*alignment*) occurring especially at the assembling level of the DNA reading process. It will be considered after a presentation of all the stages of DNA reading.

3. Sequencing

The most modern biochemical method allowing to get the information about a sequence of the considered DNA fragment is a *hybridization* experiment [17,27,36]. (In the labs DNA sequences are still sequenced by the gel electrophoresis approach which, however, leads to many errors.) The aim of this experiment is to detect all oligonucleotides of a given length l (usually 8–12 bases) composing a DNA chain (a few hundreds of bases). For this purpose the *oligonucleotide library*, consisting of all 4^l possible single-stranded DNA fragments of length l, is constructed in a form of bio-chip. Next, the library is compared (in the sense of hybridization) with many copies of the DNA chain. During the hybridization, oligonucleotides from the library complementary to fragments of the DNA chain join its copies. These oligonucleotides, written as words over the alphabet {A, C, G, T}, make a set called *spectrum*, which, of course, may contain erroneous data.

There are two types of error: a negative error, i.e. a deficit in a spectrum, and a positive error, i.e. an excess in a spectrum. Usually a spectrum contains both types of error. The *negative error* appears, for example, if an oligonucleotide exists more than once in an original sequence. Because spectrum is a set, only one of its elements corresponds to this oligonucleotide. An oligonucleotide complementary to the DNA chain may also be not detected as the element of a spectrum as a consequence of an incomplete hybridization reaction. The *positive error* is a non-complementary oligonucleotide which joins the chain, i.e. having not all its bases complementary to the considered DNA chain. Therefore, as a result of the hybridization experiment, one obtains a spectrum where not all words contained in the original sequence appear, and in which words not contained in the original sequence appear. Usually no additional information about spectrum is known. The length n of the original sequence can be detected using gel electrophoresis.

The computational phase of a sequencing process consists in the reconstruction of an original sequence on the basis of the spectrum. Older algorithms solving the sequencing problem assumed only the ideal spectrum, i.e. the one without errors. It consists of $n - l + 1$ different words and to reconstruct an original sequence, neighboring words should overlap on $l - 1$ letters. The first sequencing algorithm has been presented by Bains and Smith in [1]. It builds a search tree from the spectrum, where two oligonucleotides constituting its nodes are joined by an arc if the $l - 1$ last letters of the predecessor coincide with the $l - 1$ first letters of the successor. No element can be included twice into the current path. In the root, the element beginning the original sequence is placed, if it is known. If not, $|spectrum|$ trees differing by the root must be constructed. The solution is a path from the root to a leaf, containing all spectrum elements (Example 1). The next approach to the sequencing problem of Lysov et al. [25] used a transformation to a known problem from graph theory. In a directed graph, built from the spectrum, a Hamiltonian path is looked for. Each oligonucletide corresponds to a different vertex in this graph labeled by a string corresponding to this oligonucleotide. Two vertices u and v are joined by an arc (u, v) if the $l - 1$ last letters of the label of u coincide with the $l - 1$ first letters of the label of v (Example 1). In [15] the method of Drmanac et al. similar to the one of Bains and Smith, but avoiding excessive operations, has been proposed. Subpaths between branches of the search tree are fixed and represented by words of length l or longer. The nodes of the tree are joined if the corresponding words overlap on $l - 1$ letters (Example 1).

Example 1. Let us assume, that the hybridization experiment has been carried out without errors for the original sequence ACTCTGG, and that we know the starting element ACT. The ideal

spectrum is {ACT, CTC, CTG, TCT, TGG}. We see, that $|spectrum| = n - l + 1$, where $n = 7$ and $l = 3$. The method of Bains and Smith explores the tree from Fig. 1.

The lower path goes through all elements of the spectrum, so it is the solution. The original sequence can be reconstructed reading labels of the nodes from the root to the leaf.

The graph corresponding to the method of Lysov et al. [25], built on the basis of the same spectrum, has been presented in Fig. 2.

In the graph, there exists exactly one Hamiltonian path, what corresponds to one sequence of given length n, possible to reconstruct on the basis of the spectrum.

The method by Drmanac et al. searches for the solution in the tree shown in Fig. 3. Similarly, the solution is the path from the root to the leaf containing all the spectrum elements. Here, it is the path (ACT, CTCT, CTGG) corresponding to sequence ACTCTGG.

The three approaches accept as input data only the ideal spectrum, but despite this they have exponential complexity. The first and only polynomial time algorithm solving the sequencing problem without errors has been presented by

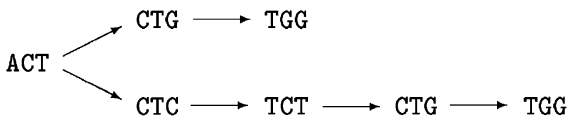

Fig. 1. The search tree from the method of Bains and Smith [1].

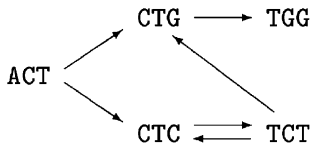

Fig. 2. The graph from the method of Lysov et al. [25].

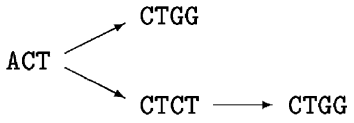

Fig. 3. The search tree for the method of Drmanac et al. [15].

Pevzner in [29]. In this method a directed graph is constructed, and an Eulerian path is looked for. Now, each oligonucleotide corresponds to a different arc, leaving the vertex labelled by its first $l - 1$ letters and entering the vertex labelled by its last $l - 1$ letters (Example 2). Thus, the number of vertices in the graph is equal to the number of all possible subwords of length $l - 1$ in the spectrum. In the original paper Pevzner did not prove the correctness of the transformation proposed. This problem was addressed later in [10], where the equivalence of the two representations was proved using the concept of directed line graphs. More formally, a directed line graph is *Hamiltonian* (Lysov model) if and only if its original directed graph is *Eulerian* (Pevzner model). As a byproduct, a new class of directed graphs, called *DNA graphs*, which are used for the representation of the results of the ideal (without errors) hybridization experiment, was defined. These graphs are induced subgraphs of de Bruijn graphs [14] and, moreover, Lysov graphs are the DNA graphs. Their properties were analyzed in a number of papers [6,10,28], but some problems remain open.

In his paper, Pevzner [29] proposed also a method for a spectrum with negative errors, but it not always lead to the solution of the problem and it may be treated only as a heuristic approach. (The sequencing problem in the presence of negative or positive errors in the spectrum has been proved to be strongly NP-hard in [12].) Each negative error causes the lack of one arc in the graph. Pevzner looks for the missing arcs solving a flow problem in a network based on a bipartite graph. The bipartite graph is constructed from the vertices of the original graph, having different numbers of entering and leaving arcs. The numbers of vertices used are equal to the values of these differences. Arcs in the bipartite graph leave the vertices with greater in-degree and enter the others. The arc costs are set to the value of a minimum shift corresponding to a maximum overlap of vertex labels. If the flow cost is equal to $n - l + 1 - |spectrum|$, the original graph is completed by the arcs used by the flow. Next, an Eulerian path can be looked for in the graph, under the condition that it is connected (Example 2).

Example 2. For the ideal spectrum from Example 1: {ACT, CTC, CTG, TCT, TGG}, the graph created by the Pevzner's method is presented in Fig. 4.

The Eulerian path in this graph corresponds to the original sequence ACTCTGG. Let us assume now, that as a consequence of an experimental error we loose the element CTG. To complete the graph we should find a flow of cost 1 in the network given in Fig. 5.

Following source s we have the vertices of greater in-degree in the original graph (i.e. the graph in Fig. 4 with arc CTG deleted), before sink t the vertices of greater out-degree. The arcs of costs greater than 1 represent in fact a sequence of arcs and vertices, not necessarily appearing in the original graph. Capacities of all arcs are equal to 1. There exists one flow pattern of value 1 and cost 1; it contains arc (CT,TG) corresponding to the missing vertex CTG in the original graph.

The first method assuming existence of positive errors in the spectrum was presented in [16]. However, the error model was very restricted. A positive error could appear only together with an associated negative error, and the improper letters could only be the first or the last ones. Three negative errors in this model could not appear consecutively. The algorithm in a few steps assembled subsequences into a larger contigs and removed obvious errors.

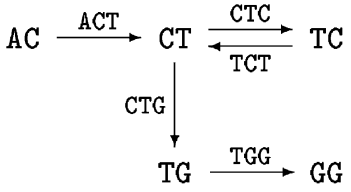

Fig. 4. The graph obtained by the Pevzner's method for the spectrum from Example 1.

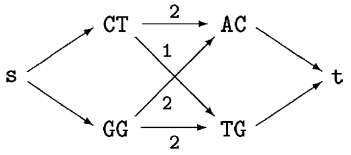

Fig. 5. The network obtained by the Pevzner's method for the graph given in Fig. 4.

The three following algorithms require additional information about spectrum elements. The first of them, described in [37], has to know a probability of a hybridization of all 4^l oligonucleotides (they could be read from the hybridization image and come from the brightness of its points). The method generates all possible 4^n sequences and associates with them values being the sums of the probability degrees of all oligonucleotides composing the sequences. The sequence of maximum value is taken as the solution. However, the authors admitted that the method does not work in practice for n greater than 20, because of its large consumption of time and memory. The algorithm from [24], besides the knowledge about error types in the spectrum, requires the knowledge of the percentage of errors within the spectrum (separately for positive and negative ones). Any negative errors are permitted in this method, the model of positive errors has been taken from [16]. The method has been derived from the Pevzner method for negative errors [29], thus cannot be treated as an exact one. The directed bipartite graph representing the whole spectrum is created, and to every arc the probability of existence in a solution is assigned. The best assignment in this graph, corresponding to the best matching of oligonucleotides, is looked for using the Hungarian method. The probabilities of arcs are then updated and a new assignment is chosen, until convergence of successive assignment values. The next algorithm [21] assumes that approximate numbers of appearances of all oligonucleotides in the original sequence are known. It accepts any negative errors in the spectrum. In the graph built on 4^{l-1} vertices, where arcs correspond to spectrum elements and have assigned the already mentioned approximate numbers, the most probable path is looked for.

The algorithms mentioned below do not require any additional information besides the spectrum and the length of the original sequence; they accept any error type and are based on the approach proposed in [8]. Its criterion function to be maximized is the number of spectrum elements composing a sequence not longer than n. The problem of DNA sequencing is reduced here to a variant of

prize collecting traveling salesman problem (known also as a selective traveling salesman problem) in a special graph constructed on the basis of the spectrum. The *prize collecting traveling salesman problem* (*PCTSP* in short) differs from the classical version of *TSP* by the existence of profit values on nodes, besides usual penalties (costs) on directed arcs [23]. The goal is to find the most profitable simple path (total profit is maximum) which does not exceed the assumed cost (penalty). In the case of the DNA sequencing problem, Lysov model is used but now all the directed arcs between vertices are possible, their cost being equal to a minimum shift between the two labels of the two vertices connected by the arc. A cost greater than one corresponds to a missing oligonucleotide(s). Profits of all vertices are set to 1. Now, one looks for a most profitable simple path (a maximum number of oligonucleotides from the spectrum will be chosen) of total cost not exceeding $n - l$ (the length of the constructed sequence will not exceed value n). Example 3 gives more details of this construction.

Example 3. Once more let us consider the spectrum from Example 1, but now let oligonucleotide CTC be lost, while a new (error) element GAT is recorded. Thus, the erroneous spectrum has the form {ACT, CTG, GAT, TCT, TGG}. The corresponding graph for the *PCTSP* is shown in Fig. 6.

In the graph given in Fig. 6 there are two paths being equivalent solutions for the corresponding *prize collecting traveling salesman problem*. One of them, i.e. (ACT, TCT, CTG, TGG), leads to the desired sequence.

In the original paper [8] an exact method based on branch and bound has been proposed. The same criterion function has been used in a few metaheuristic approaches: the tabu search algorithms [3,9] and the hybrid genetic algorithm [13]. The heuristic algorithm from [4] composes a solution from the most probable segments created by maximization of a thickness function.

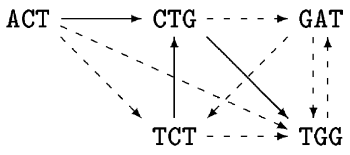

Fig. 6. A reduction from the erroneous spectrum to *PCTSP*. Solid arcs have costs equal to 1, while dashed ones equal to 2. For the sake of simplicity arcs with costs equal to 3 are not drawn.

To this end let us also mention a totally new approach to the hybridization stage of the sequencing problem, based on equal melting temperatures instead of lengths of oligonucleotides composing the library [7]. More stable duplexes should lead to spectra containing less experimental errors.

4. Assembling

As mentioned earlier, by using sequencing procedures it is possible to read DNA sequences of length up to 1000 nucleotides. Obviously, there is a need to assemble the sequenced short fragments into longer ones. This is probably the most difficult stage of reading genomic sequences. There are many problems very difficult to handle. One of them is how to manage errors following from subsequence repetitions. Known assembling algorithms usually treat long repetitive fragments as several occurrences of the same subsequence. It means that as a result the algorithms produce sequences containing only one copy of such a repetitive fragment.

In general, the assembling approach consists of three main steps: overlap detection, fragment layout and deciding the consensus [20].

In the first step, for each ordered pair of fragments it is determined how well they match each other. As the fragments for sequencing are obtained by random cutting of a great number of longer sequences, the short sequenced fragments should overlap. The goal of this step is to find the best overlap of each pair. To be more precise, for a pair of sequences (s, t) the goal is to find a suffix of s and a prefix of t whose similarity is maximal over all pairs of suffices and prefices of these two sequences [20]. The similarity is usually defined in a similar way to those used in determining sequence alignments (which are described later). Observe, that due to sequencing errors it is not enough to

find the largest suffix of the first sequence that exactly matches the prefix of the second one.

Example 4. Fig. 7 shows an example of assembling a few DNA fragments. Overlaps between the fragments usually are not exact, they may contain mismatches (when two overlapping letters are different) and gaps (when a letter appears against a space). Table 1 contains scores for selected overlaps of the fragments, with the assumed values of match $= +1$, mismatch $= -1$ and gap $= -1$. The consensus sequence for the assembly is written below the line on Fig. 7.

In the second step of the assembling procedure, an ordering of the short fragments is determined. At this stage of the approach greedy algorithms are usually used. One approach is as follows [20]. A pair of sequences with the highest suffix–prefix similarity is chosen. The two sequences are merged. Next, the second most similar (in "suffix–prefix" sense) pair is chosen and merged. At this point one may obtain one contig consisting of three fragments or two distinct contigs containing four fragments. Then, the third best matched pair of sequences is determined and merged, and so on.

In this approach only the information about similarity determined in the previous step of the approach is used. It means that if two contigs or one contig with a single fragment are merged, then the information about similarity of these strings is not used. This information would increase the quality of the layout obtained, but, on the other hand, the approach would be more time consuming.

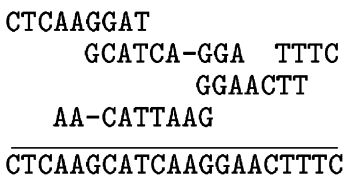

Fig. 7. Example of a DNA sequence assembly.

Table 1
Selected overlaps of the fragments from Fig. 7

s	*t*	Shift	Similarity score
CTCAAGGAT	GCATCAGGA	5	+2
CTCAAGGAT	AACATTAAG	3	+2
GCATCAGGA	GGAACTT	6	+3
GGAACTT	TTTC	5	+2
AACATTAAG	GCATCAGGA	2	+2
AACATTAAG	GGAACTT	8	+1

In the last step of the approach, a sequence of nucleotides of the assembled longer fragment is determined [20]. The layout determined in the second step assigns each nucleotide of every short fragment to a unique position in the target sequence. If all nucleotides assigned to a particular position are the same then the target sequence is recovered. Otherwise, when different nucleotides are assigned to a given position, one particular nucleotide has to be chosen. It may also happen that it is impossible to determine such a nucleotide because there is too much variety of different nucleotides assigned and none of them is dominant.

The simplest way to determine a proper nucleotide is to check the frequency of each character at each position of the layout and let the researcher decide how to use such information. Another approach is to determine regions of the layout where disagreements are large. For each such region the subsequences contained in it are multiple aligned. Observe that this may change the layout a bit because in the first step of the assembling procedure only pairwise comparisons were done. Another approach is based on remerging the subsequences in the layout. This is done similarly like in the second step, but this time each fragment is merged to its contig using an alignment of itself with a profile of fragments already present in the contig, which is less time consuming than making a multiple alignment.

At the end of this subsection let us shortly describe three particular assembling methods, which use some of the ideas described for the sequencing stage (cf. Section 3).

In [26] the problem was solved by a maximum-likelihood reconstruction algorithm. The multigraph is constructed, in which vertices correspond to DNA fragments being assembled and edges correspond to their overlaps. The overlaps are allowed between fragments from the same DNA strand or from different strands. Next, the graph is

reduced in a few steps. First, the vertices corresponding to fragments contained in longer ones are deleted. Next, the transitive edges are removed, i.e. the edge (u, v) while (u, w) and (w, v) exist in the graph. And finally, unique subpaths are replaced by single vertices. In the new graph, the shortest sequence corresponds to a simple path maximizing overlaps of its edges. The most likely solution is looked for by a branch and bound procedure.

The method proposed in [22] applied the method from [29] to the sequence assembly problem. Every input sequence of length n is represented there as a collection of $n - l + 1$ oligonucleotides of length l composing the sequence. Next, a graph is built from these collections and an Eulerian path is looked for. In order to save information about the input sequences, a large value of l is preferred. This method works properly only for instances without sequencing errors.

The next method, described in [31], solves the assembly problem for instances including several errors, and even for whole genomes including long repeats. It eliminates errors from an instance and looks for the Eulerian superpath in a graph. The error correction stage consists in a modification of the instance such that in a relatively small number of mutations in fragments, the maximum number of potential errors coming from the sequencing stage are eliminated. The effectiveness of the mutations is measured by the overall number of oligonucleotides in the instance: the change of one erroneous nucleotide usually eliminates a series of oligonucleotides. The graph is constructed in a similar way as in the Pevzner's method, and in addition a collection of paths in the graph, corresponding to the DNA fragments to be assembled, is stored. The goal is to find an Eulerian path containing all the stored paths as its subpaths.

5. Genome mapping

As we said, mapping is applied at the highest level of the process of reading genomic sequences. The goal of the mapping procedure is to recover an order of relatively long DNA fragments lost during cutting a DNA molecule. This goal is reached by determining the location of *markers*. The latter can be defined and then used to construct a physical genome map in two ways: either by *hybridization techniques* or by *restriction site analysis*.

In the first approach, one must verify whether or not some small markers (called *probes*) *hybridize* with the analyzed DNA fragment. In general, the mapped DNA molecule is broken up into fragments of different sizes. Each resulting fragment is cloned and as a result several copies (*clones*) are obtained, forming a *clone library*. A hybridization experiment is made to check if known probes hybridize with particular clones. The above information is then translated into the DNA molecule map, i.e. an *ordering of clones* (and hence probes) in the molecule. A very interesting way of representing the above hybridization process by means of special graphs was proposed by Benzer [2]. Here, each clone corresponds to a node of a graph and two nodes are joined by an arc if and only if there exists a probe hybridizing with two clones. In an ideal case, if the experiment had no errors, the resulting graph will be an *interval graph*, i.e. one where nodes correspond to *intervals* drawn in line and two nodes are joined by an undirected arc if and only if they correspond to intervals with a non-empty intersection (see example in Fig. 8). The inventor of interval graphs (S. Benzer) used them in the analysis of the genetic map of region II of phage T4.

What is more interesting, probes and clones can be described by matrix

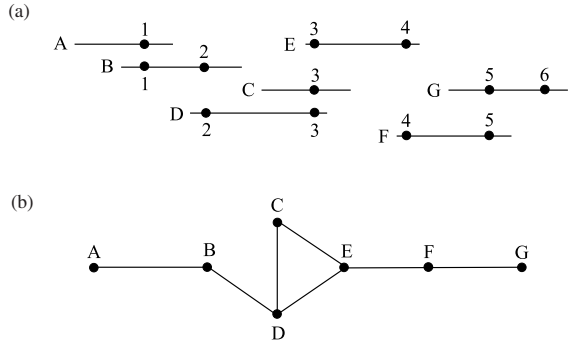

Fig. 8. An example of mapping by hybridization: (a) clones A, B, C, D, E, F and G with hybridizing probes 1, 2, 3, 4, 5 and 6; (b) the resulting interval graph.

$A = [a_{ij}]_{s\times t}$,

where

$$a_{ij} = \begin{cases} 1 & \text{if probe } j \text{ hybridizes with clone } i, \\ 0 & \text{otherwise,} \end{cases}$$

s – a number of clones,

t – a number of probes.

Finding a genetic map is now equivalent to finding a permutation of columns (probes) of matrix A, for which in each row *ones appear consecutively without breaks*, i.e. the matrix has the *consecutive ones property* (*C1P* for short). In case of an ideal hybridization experiment matrix A has C1P and such a permutation can be found in polynomial time (e.g. [19]).

Let us now come back to the second way of genome mapping, i.e. the restriction site analysis. The input data are in this case the lengths of the fragments and information about the method used to cut the long DNA by restriction enzymes. There are two main approaches known in the literature: *double digest method* and *single digest method* [33,39]. In the former one the target DNA molecule is cut by two enzymes, let us say α and β. As a result one gets three sets of DNA fragments, i.e. one coming from the cuts of the molecule by enzyme α, one from the cuts by enzyme β, and one from the cuts by both enzymes. As we said before, one knows only the lengths of the fragments. Let us formally define the *double digest problem* [39].

Let $\mathscr{A} = \{a_1, a_2, \ldots, a_m\}$ and $\mathscr{B} = \{b_1, b_2, \ldots, b_n\}$ be multisets of lengths of restriction fragments obtained by cutting the DNA molecule by enzymes α and β, respectively. Moreover, let $\mathscr{C} = \{c_1, c_2, \ldots, c_k\}$ be a multiset of lengths of fragments obtained by cutting the molecule by both enzymes. Let ψ and ω be permutations of elements from sets $\{1, 2, \ldots, m\}$ and $\{1, 2, \ldots, n\}$, respectively. By ordering multisets $\mathscr{A}$ and $\mathscr{B}$ according to permutations ψ and ω one obtains a set of possible locations of the cut sites:

$$\mathscr{S} = \left\{ s : s = \sum_{1\leqslant i\leqslant p} a_{\psi(i)} \vee s = \sum_{1\leqslant i\leqslant q} b_{\omega(i)}; \quad 0 \leqslant p \leqslant m,\ 0 \leqslant q \leqslant n \right\}.$$

Let us number the elements of set $\mathscr{S}$ such that for every pair $\{s_i, s_j\} \in \mathscr{S}$, $s_i \leqslant s_j$ iff $i \leqslant j$, for $0 \leqslant i, j \leqslant k$.

The double digest induced by a pair (ψ, ω) is multiset

$$\mathscr{C}(\psi, \omega) = \{c_i(\psi, \omega) : c_i(\psi, \omega) = s_i - s_{i-1}, \quad 1 \leqslant i \leqslant k\},$$

where for every $\{c_i(\psi, \omega), c_j(\psi, \omega)\}$, $c_i(\psi, \omega) \leqslant c_j(\psi, \omega)$ iff $i \leqslant j$, for $0 \leqslant i, j \leqslant k$.

The solution of the problem is pair (ψ, ω) such that $\mathscr{C} = \mathscr{C}(\psi, \omega)$.

Intuitively, a solution to the *double digest problem* consists of permutations of elements from the three sets such that if they were placed on a line the fragments from sets corresponding to enzymes α and β would "produce" the fragments from the third set as shown in Fig. 9.

Let us comment here on the complexity of the double digest problem. It is easy to notice that the decision version of the problem is a trivially easy problem, because if multisets $\mathscr{A}$, $\mathscr{B}$ and $\mathscr{C}$ come from real digestion experiments, the answer to the question concerning the existence of proper permutations ψ and ω is always positive. On the other hand, the search version of the double digest problem (i.e. finding permutations ψ and ω) is NP-hard in the strong sense. The latter result was obtained by a quite involved technique using polynomial Turing reduction from the *Partition* problem [11].

From the practical point of view the double digest problem besides its high complexity has also another drawback which is its high number of alternative solutions. It grows exponentially with the number of restriction sites [32].

In the single digest method only one enzyme is used but the resulting digested fragments of the

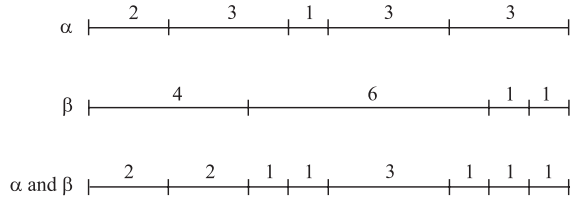

Fig. 9. An example of a double digest problem instance. A DNA sequence is cut by enzyme α, enzyme β and by both enzymes. Three multisets of fragment lengths are obtained: $\mathscr{A} = \{1, 2, 3, 3, 3\}$, $\mathscr{B} = \{1, 1, 4, 6\}$ and $\mathscr{C} = \{1, 1, 1, 1, 1, 2, 2, 3\}$.

target DNA are divided into several parts. In each of them different time spans for the chemical reaction is allowed. The spans are chosen in such a way that the shortest one is enough for the enzyme to cut DNA in one restriction site only, the second time span allows the enzyme to cut in two sites and so on. The longest time span is enough for the enzyme to cut the target DNA in all restriction sites. As a result a collection of restriction fragments is obtained. Similarly like in the double digest method, only lengths of the fragments are known. A main disadvantage of this approach is a difficulty to choose proper time spans, which in practice is almost impossible. Computational complexity of *single digest problem* is an open question, while the number of alternative solutions generated (i.e. maps) is bounded above by a polynomial in the number of restriction sites [34].

Recently, a new restriction mapping method called *simplified partial digest method* has been proposed [5]. In this method only one restriction enzyme is also used but in this case only two sets of target DNA are digested. For one of these sets the time allowed for the reaction is chosen in such a way that the molecules are cut in at most one site. The time span for the other set is long enough so that the enzyme cuts the molecules in all restriction sites. We see that from a biochemical point of view this approach is much more realistic than the single digest method.

Let us define formally the *simplified partial digest problem* [5]. Let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{2N}\}$ be a multiset of fragment lengths obtained from the short digest reaction (excluding the length of the whole DNA strand) and let $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_{N+1}\}$ be a multiset of fragment lengths coming from the long digest reaction, where N is the number of restriction sites in the target DNA. Furthermore, let us sort elements of multiset Γ in non-decreasing order. In this way we obtain list $A = \langle a_1, a_2, \ldots, a_{2N}\rangle$ of lengths of restriction fragments.

It is easy to see that in the ideal case (no experimental errors assumed) to each element a_i from A there corresponds element a_{2N-i+1} such that $a_i + a_{2N-i+1} = L$, where L is the length of the target DNA strand. We will call such fragments *complementary* and denote them by $\{a_i, a_{2N-i+1}\}$. Each pair of complementary fragments corresponds to one restriction site in the target DNA. Obviously, the real order of such fragments in the target strand is unknown. Let $P_i = \langle a_i, a_{2N-i+1}\rangle$ and $P_{2N-i+1} = \langle a_{2N-i+1}, a_i\rangle$ denote permutations of pair $\{a_i, a_{2N-i+1}\}$ and let us call a_i the *predecessor* in P_i. Let us denote by $Q = \{q_1, q_2, \ldots, q_N\}$ the set of complementary fragment permutations, where $q_i = P_i$ or $q_i = P_{2N-i+1}$ for $i = 1, \ldots, N$. Moreover, let $X = \langle x_1, x_2, \ldots, x_N\rangle$ be the sorted list (in non-decreasing order) of predecessors from each permutation in Q. To each set Q corresponds multiset $R = \{r_1, r_2, \ldots, r_{N+1}\}$ of integer numbers such that $r_1 = x_1$, $r_i = x_i - x_{i-1}$ for $i = 2, \ldots, N$ and $r_{N+1} = L - x_N$.

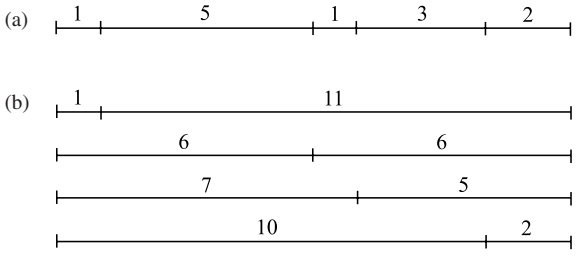

Fig. 10. An example of an instance of simplified partial digest problem: (a) DNA molecule with restriction sites to be found (the numbers indicate restriction fragments lengths observed in the biochemical experiment); (b) DNA fragments cut by the enzyme in short chemical reaction.

Now we can formulate the SIMPLIFIED PARTIAL DIGEST PROBLEM (SPDP) as follows:

Given multisets Γ and Λ of fragment lengths, find set Q such that the corresponding multiset R is equal to Λ.

Fig. 10 shows an example of an instance of simplified partial digest problem.

Complexity of the above problem is still an open question as well as the number of solutions generated. On the other hand, computational experiments proved a high efficiency of the approach proposed and its high robustness for the experimental errors [5].

6. Sequence comparison

The knowledge about genomic sequences without understanding their meaning would be almost

useless. So, after sequencing a new gene or genome, scientists usually try to understand information encoded. But this task is much more challenging than reading sequences and, unfortunately, it is also much less developed. No one knows the "language of DNA", so nowadays it is impossible to guess the function of a new gene simply looking at its nucleotide sequence. Instead, scientists try to find some analogies between this gene and genes which are known for some time as well as their functions. At this point a question may arise: what is the source of the knowledge about functions of the "old" genes? The answer is: it could be discovered in a similar way or in series of biochemical experiments. But the goal is to develop methods for automatic gene function discoveries, since biochemical experiments are time and money consuming and they often provide non-precise results. We see that *sequence comparison* is the most common method for looking for gene functions. And it is also one of the first problems in molecular biology which attracted the attention of mathematicians.

Sequence comparison is usually done by sequence alignment. The goal of such an alignment is to match two or more sequences in such a way that in the corresponding positions in these sequences there will be the same characters, i.e. nucleotides or aminoacids. Usually, it is necessary to insert some gaps (i.e. spaces) in order to make such an alignment better (see Fig. 11). Obviously, for a given set of sequences there is a number of possible alignments. One is usually interested in finding the best of them. But the question is: which one is the best? The answer depends on a biological context. But, in general, an objective function based on some scoring scheme is used. In the simplest form the scheme assigns some points to every column in the alignment, depending on what the column contains: match, mismatch or a gap. For example, the scheme may assign "+1" for a column with identical characters, "−2" for a column with different characters, and "−1" for a column containing a gap. A sum of the scores of all columns is a *similarity* of the sequences and it is a value of the objective function. Usually, one is interested in finding an alignment with the maximum value of similarity. There are different scoring schemes and there is no scheme which would be the best one for all applications. The scheme should be chosen depending on the biological application of the alignment.

Alignments of pairs of sequences are usually constructed by methods based on dynamic programming [33,39]. But in practice, researchers are interested in alignments of a greater number of sequences, which help them to find some similarities in sequences coming from families of organisms. Multiple sequence alignments are usually time consuming. One should also notice that not only alignments of nucleic acids are made. Very often aminoacid sequences are compared. Molecular biologists are often interested in discovering some important regions in polypeptide sequences. It is known that some subsequences in aminoacid strands play a crucial role in the functionality of a given protein. Discovering such regions is very important for understanding the mechanism of this functionality.

Some proteins are present in different species. Their aminoacid sequences need not be identical, but usually they contain some well-conserved regions identical or almost identical in all of them.

It should be also noticed that comparing aminoacid sequences is more complicated than comparing nucleotide ones. This is due to the fact that there are 20 types of aminoacids and only four types of nucleotides and some types of aminoacids may be thought of as equivalent in some sense. Hence, even if in an alignment there are different aminoacids at a given position, they may sometimes be treated as a match.

```
sequence a    CAAGC-A-CACTTG--ATGTACAGTCG
sequence b    CAA-CTAGCAGTTGTAATGT-CAG-CG

scores        +++-+-+-++#+++--++++-+++-++
```

Fig. 11. An example of an alignment of two sequences a = CAAGCACACTTGATGTACAGTCG and b = CAACTAGCAGTTGTAATGTCAGCG. In the third line are shown scores for a match ("+" corresponds to +1), a gap ("−" corresponds to −1) and for a mismatch ("#" corresponds to −2). The value of the objective function in this case is equal to 10.

...TCGGTC**TATAA**AGATCAG**GATTA**TCCAA...
...AA**TATATA**GCGGCTTACCACCC**GATA**CT...
...TCATACGAATAA**TATATA**CATCA**GATTA**...

Fig. 12. Example of some well-conserved motifs in DNA sequences.

The alignment discussed above is a *global alignment* – whole sequences are compared with each other. But in biological practice it is interesting to find some subsequences in a set of sequences that are the most similar to each other. As an example, we may consider the problem where one looks for some important regions in a protein family, as mentioned above. This kind of alignment is called a *local alignment*. It has also an application in evolutionary research. This is because related organisms have some common features, which have their origins in nucleotide and aminoacid sequences. Hence, some fragments of these sequences coming from different but related species have some similarities. Analyzing these similarities allows for finding degrees of affinity of the species. In this way their phylogenetic trees can also be constructed (we will say more about these problems in the next section).

A slight modification of the dynamic programming method proposed for the global alignment may be used for constructing local alignments [35].

Similar to the local alignment problem is *searching for motifs*. There are many variants of such searching procedures. In the simplest case, the motif is in fact a pattern which has to be found in every sequence from a given set. There are many well-known efficient algorithms for this problem (known for years in computer science community). In a more complicated variant the motif is described as a regular expression. In this case, the search can be also done in polynomial time, since parsing regular grammars is a computationally easy problem. But the most interesting, from a biological viewpoint, is the variant of the problem where the structure of the motif is unknown. One knows only that there should be some similar subsequences in a given set of sequences. It is often the case in searching for conserved regions in protein families or in searching for regulatory regions in DNA. In the latter case, one knows that in a set of DNA sequences coming from different organisms (all of them containing the same or a similar gene) there must also exist some important subsequences preceding the gene which controls a transcription of it. Here, an approach using *suffix trees* [38,41] is extremely useful. Fig. 12 shows an example of motifs in DNA sequences.

References

[1] W. Bains, G.C. Smith, A novel method for nucleic acid sequence determination, Journal of Theoretical Biology 135 (1988) 303–307.
[2] S. Benzer, On the topology of the genetic fine structure, Proceedings of the National Academy of Sciences of the USA 45 (1959) 1607–1620.
[3] J. Błażewicz, P. Formanowicz, F. Glover, M. Kasprzak, J. Węglarz, An improved tabu search algorithm for DNA sequencing with errors, in: Proceedings of the III Metaheuristics International Conference MIC'99, pp. 69–75.
[4] J. Błażewicz, P. Formanowicz, F. Guinand, M. Kasprzak, A heuristic managing errors for DNA sequencing, Bioinformatics 18 (2002) 652–660.
[5] J. Błażewicz, P. Formanowicz, M. Jaroszewski, M. Kasprzak, W.T. Markiewicz, Construction of DNA restriction maps based on a simplified experiment, Bioinformatics 17 (2001) 398–404.
[6] J. Błażewicz, P. Formanowicz, M. Kasprzak, D. Kobler, On the recognition of de Bruijn graphs and their induced subgraphs, Discrete Mathematics 245 (2002) 81–92.
[7] J. Błażewicz, P. Formanowicz, M. Kasprzak, W.T. Markiewicz, Method of sequencing of nucleic acids, The Patent Office of the Republic of Poland, Patent Application No. P 335786, 1999.
[8] J. Błażewicz, P. Formanowicz, M. Kasprzak, W.T. Markiewicz, J. Węglarz, DNA sequencing with positive and negative errors, Journal of Computational Biology 6 (1999) 113–123.
[9] J. Błażewicz, F. Glover, M. Kasprzak, DNA sequencing—tabu and scatter search combined, INFORMS Journal on Computing, in press.
[10] J. Błażewicz, A. Hertz, D. Kobler, D. de Werra, On some properties of DNA graphs, Discrete Applied Mathematics 98 (1999) 1–19.
[11] J. Błażewicz, M. Jaroszewski, M. Kasprzak, B. Paliświat, P. Pryputniewicz, On the complexity of the double digest problem, Control & Cybernetics (2004) in press.

[12] J. Błażewicz, M. Kasprzak, Complexity of DNA sequencing by hybridization, Theoretical Computer Science 290 (2003) 1459–1473.
[13] J. Błażewicz, M. Kasprzak, W. Kuroczycki, Hybrid genetic algorithm for DNA sequencing with errors, Journal of Heuristics 8 (2002) 495–502.
[14] N.G. de Bruijn, A combinatorial problem, Koninklijke Nederlandse Akademie van Wetenschappen 49 (1946) 758–764.
[15] R. Drmanac, I. Labat, I. Brukner, R. Crkvenjakov, Sequencing of megabase plus DNA by hybridization: Theory of the method, Genomics 4 (1989) 114–128.
[16] R. Drmanac, I. Labat, R. Crkvenjakov, An algorithm for the DNA sequence generation from k-tuple word contents of the minimal number of random fragments, Journal of Biomolecular Structure and Dynamics 8 (1991) 1085–1102.
[17] S.P.A. Fodor, J.L. Read, M.C. Pirrung, L. Stryer, A.T. Lu, D. Solas, Light-directed, spatially addressable parallel chemical synthesis, Science 251 (1991) 767–773.
[18] G.B. Fogel, D.W. Corne (Eds.), Evolutionary Computation in Bioinformatics, Morgan Kaufmann Publishers, San Francisco, 2003.
[19] D.R. Fulkerson, O.A. Gross, Incidence matrices and interval graphs, Pacific Journal of Mathematics 15 (1965) 835–855.
[20] D. Gusfield, Algorithms on Strings, Trees, and Sequences. Computer Science and Computational Biology, Cambridge University Press, Cambridge, 1997.
[21] J.N. Hagstrom, R. Hagstrom, R. Overbeek, M. Price, L. Schrage, Maximum likelihood genetic sequence reconstruction from oligo content, Networks 24 (1994) 297–302.
[22] R. Idury, M.S. Waterman, A new algorithm for DNA sequence assembly, Journal of Computational Biology 2 (1995) 291–306.
[23] M. Laporte, S. Martello, The selective travelling salesman problem, Discrete Applied Mathematics 26 (1990) 193–207.
[24] R.J. Lipshutz, Likelihood DNA sequencing by hybridization, Journal of Biomolecular Structure and Dynamics 11 (1993) 637–653.
[25] Yu.P. Lysov, V.L. Florentiev, A.A. Khorlin, K.R. Khrapko, V.V. Shik, A.D. Mirzabekov, Determination of the nucleotide sequence of DNA using hybridization with oligonucleotides. A new method, Doklady Akademii Nauk SSSR 303 (1988) 1508–1511.
[26] E.W. Myers, Toward simplifying and accurately formulating fragment assembly, Journal of Computational Biology 2 (1995) 275–290.
[27] A.C. Pease, D. Solas, E.J. Sullivan, M.T. Cronin, C.P. Holmes, S.P.A. Fodor, Light-generated oligonucleotide arrays for rapid DNA sequence analysis, Proceedings of the National Academy of Sciences of the USA 91 (1994) 5022–5026.
[28] R. Pendavingh, P. Schuurman, G.J. Woeginger, Recognizing DNA graphs is difficult, Discrete Applied Mathematics 127 (2003) 85–94.
[29] P.A. Pevzner, l-Tuple DNA sequencing: Computer analysis, Journal of Biomolecular Structure and Dynamics 7 (1989) 63–73.
[30] P.A. Pevzner, Computational Molecular Biology. An Algorithmic Approach, The MIT Press, Cambridge, MA, 2000.
[31] P.A. Pevzner, H. Tang, M.S. Waterman, A new approach to fragment assembly in DNA sequencing, in: Proceedings of the Fifth Annual International Conference on Computational Molecular Biology, RECOMB 2001, pp. 256–267.
[32] W. Schmitt, M.S. Waterman, Multiple solutions of DNA restriction mapping problems, Advances in Applied Mathematics 12 (1991) 412–427.
[33] J. Setubal, J. Meidanis, Introduction to Computational Molecular Biology, PWS Publishing Company, Boston, 1997.
[34] S.S. Skiena, G. Sundaram, A partial digest approach to restriction site mapping, Bulletin of Mathematical Biology 56 (1994) 275–294.
[35] T.F. Smith, M.S. Waterman, The identification of common molecular subsequences, Journal of Molecular Biology 147 (1981) 195–197.
[36] E.M. Southern, United Kingdom Patent Application GB8810400, 1988.
[37] E.M. Southern, U. Maskos, J.K. Elder, Analyzing and comparing nucleic acid sequences by hybridization to arrays of oligonucleotides: Evaluation using experimental models, Genomics 13 (1992) 1008–1017.
[38] E. Ukkonen, On-line construction of suffix trees, Algorithmica 14 (1995) 249–260.
[39] M.S. Waterman, Introduction to Computational Biology. Maps, Sequences and Genomes, Chapman & Hall, London, 1995.
[40] J.D. Watson, F.H.C. Crick, Genetic implications of the structure of deoxyribonucleic acid, Nature 171 (1953) 964–967.
[41] P. Wiener, Linear pattern matching algorithms, in: Proceedings of the 14th IEEE Symposium on Switching and Automata Theory, 1973, pp. 1–11.