CMST 2(1) 17-32 (1996) DOI:10.12921/cmst.1996.02.01.17-32

A COMPARISON OF TWO DNA SEQUENCING METHODS

Jacek Błażewicz, Łukasz Gwóźdź, Marta Kasprzak, Marcin Przysucha

Institute of Computing Science, Poznań University of Technology
ul. Piotrowo 3a, 60-965 Poznań, Poland
e-mail: blazewic@poznlv.put.poznan.pl

Abstract

In the paper the problem of DNA sequencing is considered. The sequencing method by Pevzner [8] is analyzed and its generalization, allowing for finding all acceptable solutions, is described. It is then compared with another method based on a generation of a solution tree [4],
keywords: DNA sequencing, hybridization method, complexity analysis.

1. Introduction

The importance of decoding an information hidden in DNA cannot be overestimated. One of the basic problems connected with genetical experiments is identification of the structure of the DNA chain obtained as a result of the performed experiment. The first thing that has to be done to accomplish it is the recognition of the sequence of nucleotides the chain is build of. Because of the fact that in DNA helix the only pairs that can face each other are A—T and C—G (where A stands for adenine, T for tymine and so on) it is enough to know the sequence in one of the fibres, since there can be only one sequence in the other fibre that matches the recognized order.

The process of rebuilding the original sequence of nucleotides in the chain is called DNA sequencing. The input data for it is usually a set of l-tuples (of nucleotides) found in the examined chain, called later spectrum. The object is to find the order of l-tuples with the property that each following oligonucleotide covers the previous one at all but one positions, that is the last l-1 nucleotides (represented by appropriate letters) of the preceding l-nucleotide make the same sequence as l-1 first nucleotides of the following one. The length of the DNA chain one tries to reconstruct is usually denoted by n. Let's notice that in the ideal situation we have |spectrum|=n-l+1. Then, if the oligonucleotides obtained from experiment are correct, one can reconstruct the DNA sequences precisely. If |spectrum|<n-l+ 1, the difference (n-l+ l-|spectrum|) is called a defect and it is usually denoted by k.

We see, that DNA sequencing problem belongs to the search class of combinatorial problems. Several different methods of sequencing have been invented, but none of them can be called the best one. The methods found out by Bains [1], Bains and Smith [2], Drmanac et al. [6] and Khrapko et al. [7] require an error-free experiment and complete information about all the l-tuples the DNA chain is build of (|spectrum|=n-l+1), which is very hard to achieve. Apart from it, the Drmanac's method requires a change in experiment. It suggests to add a l-1-long fragments to the original chain before the hybridization experiment. This change brings an advantage, but the need of modifying the experiment makes that method less practical. All of these methods are of exponential time complexity. There's also Southern et al. [9] method, which can handle experiment errors quite well, but it generates 4^n possible sequences, which makes it totally useless in case of longer chains. This paper presents an implementation of Pevzner's methods [8], which seem to be more reasonable than previous methods. The original Pevzner's methods seem to concentrate on finding one solution. Of course, it makes the methods less practical, because when there are more than one possible solution and only one is found, we cannot be sure it is the correct one. In fact, there are cases when the methods don't give proper results even in case of searching for only one solution. In the implementation Pevzner's methods are used in an attempt to find all acceptable solutions. Some conclusions about Pevzner's article and methods presented there are included in the text. Then, they are compared with another method based on a generation of a solutions tree. The text is organized in the following way. Section 2 is a detailed description of ideas and implementation of Pevzner's methods. Section 3 describes the new method. In the last section results of performed tests and comparisons are presented.

2. Pevzner's methods

An idea of using graph theory approach for DNA sequencing came from Khrapko et al. [7], They suggested to find a Hamiltonian path in graph H constructed on n-l+1 vertices, each vertex being equivalent to one l-tuple in spectrum. Vertices v and w were joined by arc (v, w) if the last l-1 nucleotides of v were the same as the first l-1 nucleotides of w. This approach to the problem has serious disadvantages: reconstruction from incomplete spectrum (that is, containing less than n-l+1 l-tuples) is impossible, and time complexity of finding Hamiltonian path in the graph is exponential in the number of nodes, i.e. l-tuples.

Pavel A. Pevzner in 1989 [8] presented a modification of this method. In his approach a graph G, containing vertices corresponding to l-1-tuples, is created. In this graph, vertices v and w are connected by arc (v, w) if there exists l-tuple in spectrum, for which the following condition is satisfied: the first *l*-1 nucleotides of *l*-tuple correspond to *v* and the last *l*-1 nucleotides of *l*-tuple correspond to *w*. Fig. 1 presents an example for constructing the graphs H and G.

It is apparent that each arc of graph G corresponds to one vertex of graph H. Therefore problem of finding Hamiltonian path in graph H can be reduced to finding Eulerian path in the corresponding graph G. Efficient algorithm for finding Eulerian path is known, so in this case it is possible to use polynomial algorithm for determining full DNA sequence. The following criterion for the existence of Eulerian path is known:

- $\text{in}(v) = \text{out}(v) \qquad v \neq s,t$
- $\text{out}(s) - \text{in}(s) = 1$
- $\text{out}(t) - \text{in}(t) = -1$

where *s* is the first vertex in the Eulerian path, *t* is the last one, and in(*v*) and out(*v*) are indegree and outdegree of vertex *v*, respectively. Note that if only the first condition is satisfied, an Eulerian cycle also exists in graph G.

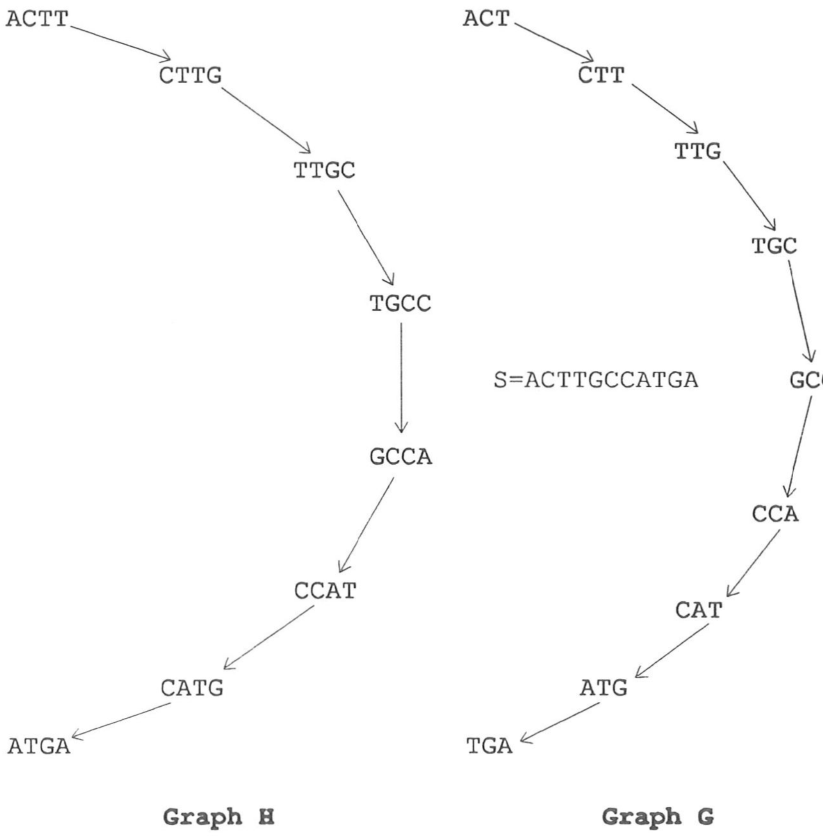

Fig. 1. Construction of graph H and G for given DNA chain S and *l*=4

However, the main aim of Pevzner's method is to find the only possible and therefore definitely correct answer. Consequently, the criterion for the existence of only one Eulerian path has to be defined. Assuming that the above criterion is satisfied, the number of Eulerian paths in a graph can be obtained by using the following formula given by de Bruijn—van Aardenne-Ehrenfest-Smith-Tutte theorem [8]:

$$D = C \cdot \prod_{v \in V} (d(v) - 1)!$$

where D is the number of Eulerian paths in a graph, V is the vertex set of graph G, and d(*v*)=in(*v*)=out(*v*) is the semi-degree of vertex *v* (an artificial arc *(t, s)* is added, so that condition in(*v*)=out(*v*) is satisfied for every vertex of a graph). C is the common value of all signed minors of the Kirchhoff matrix.

We see that D=1 if and only if C=1 and d(*v*) $\leq$ 2 for all vertices. While checking the second condition is trivial, calculating the value of C is a more difficult problem. In order to simplify checking the second condition one may only check if C=l. As stated in [8], C=1 if every vertex in the graph G is a part of no more than one cycle.

Note, that sometimes there are two possible different Eulerian paths leading to the same DNA sequence, as in the following example (this is possible only in case of incomplete spectrum, when there exist multiple *l*-tuples represented by only one entry in input data).

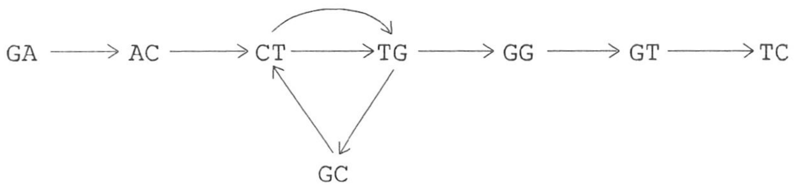

Fig. 2. Example of graph G containing two equivalent Eulerian paths

After checking these conditions it is easy to find an Eulerian path in graph G and to determine full DNA sequence. Unfortunately, spectrum obtained from experiment is not always complete. Even assuming that the experiment results are always accurate (which is a very strong assumption), defect can be caused by the fact, that the same *l*-tuple may exist in the sequence more than once, and it will be represented in the spectrum only once. Of course, the longer *l*-tuples are, the lower the number of missing sequences in spectrum is. By using the described method it is possible to determine the full DNA sequence even when the spectrum is incomplete. In this case graph G is also incomplete and finding Eulerian path in it has to be preceded by adding missing arcs.

For the dcfect k spectrum will contain $(n-l+1-k)$ l-tuples. In some cases it is possible to reconstruct the missing k l-tuples, so that the graph G can be corrected. Of course, in the incomplete graph G multiple variants of adding the missing arcs are possible, and it is necessary to find the optimal one. The method suggests adding arcs joining only vertices, for which $\mathrm{in}(v) \neq \mathrm{out}(v)$. One has to consider all the possible combinations, assigning to each possible arc its cost. Cost of an arc joining vertices v and w is the minimal c satisfied the following condition: the last l-1-c nucleotides of vertex v are the same as the first l-1-c nucleotides of vertex w. Of course, value of c can be in the range <1, l-l>. The problem is to find the combination of arcs so that the exact identification of DNA sequence is possible.

In Pevzner's method it is proposed to consider all possible missing connections, for which the conditions for Eulerian path are satisfied, and to choose the optimal one - for which the total cost equals k. Again, one is interested in finding only one acceptable connection, because it is the condition of uniqueness of the Eulerian path.

To find the correct answer to the problem of finding missing arcs, it is suggested to construct graph $K_{m,\,m}$ with vertices:

$$\mathrm{div}(v) = \mathrm{out}(v) - \mathrm{in}(v)$$
$$X = \{v \in V\colon \mathrm{div}(v) < 0\}$$
$$Y = \{v \in V\colon \mathrm{div}(v) > 0\}$$
$$m = -\sum_{v\in X} div(v) = \sum_{v \in Y} div(v)$$

The vertices with $|\mathrm{div}(v)|>1$ are duplicated $|\mathrm{div}(v)|$ times. Each vertex v in X is connected with each vertex u in Y by directed arc (v, u). This arc is assigned the cost of connection as defined earlier, and constant capacity equal to 1. In such a way a complete bipartite graph is created. Then, additional four vertices *s1*, *s2*, *t1* and *t2* are added with $2m+2$ arcs in the following way:

- arc *(s1, s2)* and arc *(t1, t2)* with capacity m-l and cost 0
- arcs *(s2, x)* for each vertex x in X and arcs $(y,\ t1)$ for each vertex y in Y with capacity 1 and cost 0

Fig. 3 presents an example of graph K constructed on the basis of spectrum from Fig. 1. The arcs (CCA, CAT) and (TTG, TGC) has been removed from graph G. The capacities of arcs are shown in circles.

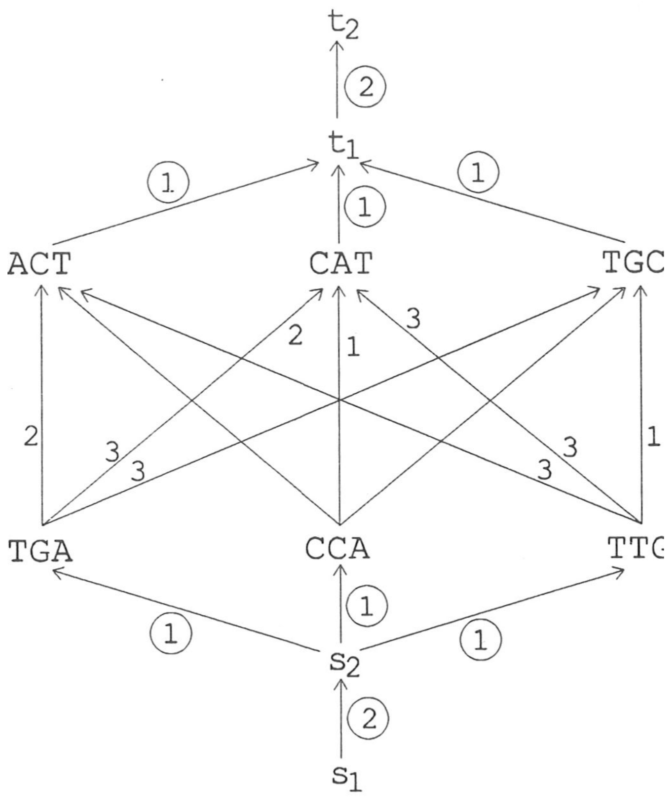

Fig. 3. Example of graph K

Now, to find an optimal connection for incomplete graph G one has to find a maximal flow of a minimal cost in graph K. After determining the arcs with a non-zero flow in K, new arcs are added to graph G, so that for each arc of cost *c* in graph K, *c* new arcs are added to graph G. For example, for arc (TGA, ACT) of cost 2, arcs (TGA, GAC) and (GAC, ACT) are added to the Eulerian graph. Then, if graph G is complete and there exists an Eulerian path in it, solution is constructed as if the spectrum was complete. Of course, if graph G is still incomplete (does not satisfy conditions for the existence of Eulerian path), the problem cannot be solved.

Polynomial in time algorithms for finding maximal flow of minimal cost are known, so - in fact - the time complexity of the whole method is polynomial. There is however one condition to be checked: the uniqueness of the constructed flow. This topic is another one described very shortly and generally, so it is difficult to determine the real Pevzner's algorithm. Below implementation of this method is described.

The method has been modified to give all possible answers. This modification caused changes in approaches to finding Eulerian paths and optimal flows, but the general idea of the method was left unchanged.

Let's assume that we have to construct full DNA sequence for a spectrum with defect k (the particular case, when $k=0$, can be solved by bypassing unnecessary finding of the optimal flow). If k is greater than 0, graph K is constructed. It is however constructed in slightly different way than in the original Pevzner's method. Note that originally we had to search for the maximal flow of minimal cost. The first major change in our implementation is that all possible optimal flows should be found, and we are not interested in the problem of a uniqueness of the flow. Classical algorithms (polynomial in time) are designed to search for only one possible maximal flow of a minimal cost. The problem is that the polynomial algorithm for finding all optimal flows is not known, and therefore we had to implement algorithm, which is exponential in time, losing by it the main advantage of Pevzner's method. We will describe algorithm we used later, after discussing other necessary changes to the construction of the graph and finding needed optimal flows.

The second thing in finding the maximal flow of minimal cost that causes doubts, is the assumption of minimal cost of the flow, which is not always equal k. The assumption was needed in order to apply the classical algorithm, as efficient algorithms for finding the maximal flow of constant cost are not known. But again, solving only cases for which minimal cost was equal k limits the number of cases, for which positive solutions to the problem can be found. Once abandoning the idea of using a classical algorithm, we can now easily change the problem to searching for all maximal flows of constant cost k, which will result in greater efficiency of the method.

Let's consider the problem of „maximal cost" now. Again we can see that the maximal possible cost of the flow is already determined at the beginning by capacity of arc $(s1, s2)$ or arc $(t1, t2)$, both set to m-1. Therefore it is clear, that we really search for the flow of value m-1. There is also one condition: every vertex can be used in the constructed flow only once (note that the vertices $v \in V$ with $|\mathrm{div}(v)| > 1$ were duplicated $|\mathrm{div}(v)|$ times); in other words, there cannot exist two arcs $(x1, y1)$ and $(x2, y2)$, where $x1, x2 \in X$ and $y1, y2 \in Y$, such that $x1$-$x2$ or $y1$-$y2$. In original Pevzner's method this is ensured by adding arcs $(s2, x)$ and $(y, t1)$ for each $x \in X$ and $y \in Y$, all with capacities 1. So, from our point of view, those additional vertices $s1, s2, t1, t2$ and all arcs connected with them are not necessary. We only have to remember, that the same vertex cannot be used twice, as mentioned before. Deleting vertices $s1, s2, t1, t2$ leads to simpler construction of graph K. The following example shows a reduction of graph K from Fig. 3.

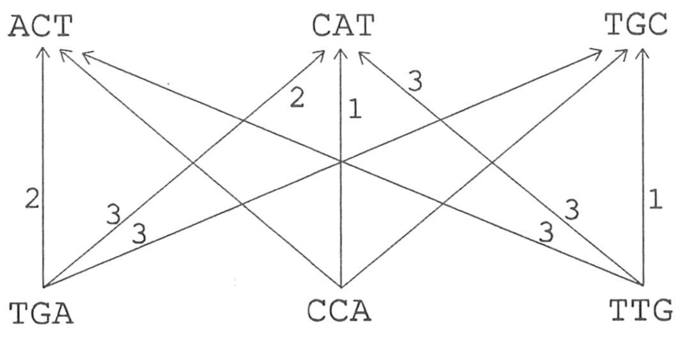

Fig. 4. Example of reduced graph K used in the implementation

In that way we transformed the problem of finding maximal flow of minimal cost to the problem of finding all possible combinations of *m*-1 arcs in graph K, for which the following conditions are satisfied:

- total cost of all arcs equals *k*
- if one arc connects vertices *x* and *y*, other arcs cannot use *x* nor *y*

Note, that we still have to find constant number of arcs - *m*-1, because combinations which contain less or more arcs, even if they satisfy the other conditions, cannot lead to successful reconstruction of graph G. We have *2m* vertices in graph G for which div(v) $\neq$ 0, and every arc selected in graph K, independently of its cost, reduces that number by 2. Therefore, to keep the number of vertices for which in and out degrees are different at the level of 2 (begin and end vertices in Eulerian path), we must select exactly *m*-1 arcs.

Finding all possible combinations for which all necessary conditions are satisfied is a difficult problem, and methods for solving it in polynomial time are not known. We can only use exponential in time algorithms and introduce some ideas to speed them up. Fortunately, even for large DNA sequences, the value of *m* is relatively small, and therefore even not very efficient algorithms will be good enough to solve the problem in a reasonable time. Moreover, as was shown in test results, finding all the combinations took very little time (very often technically unmeasurable) in comparison to searching for all Eulerian paths in graph G; in fact, less than 1% of total time.

In our algorithm a graph is represented in a form of the list of arcs (in fact it is a dynamically allocated table of size *m*x*m*). For each entry in the table the following data are stored:

- the cost of arc - *c*
- vertices connected by the arc - two indices within the range from 1 to *m*

Each vertex *x* in X and *y* in Y has its unique index, which is also used in tables „usedX" and „usedY", both containing *m* entries. These tables are used for remembering which vertices have already been allocated for the combination and cannot be used any more. There is also one table named „flow" of size m-1, which contains currently constructed combination.

The main idea of the algorithm is to construct all possible combinations with quick backing up, if currently considered arcs do not give any chances for creating correct combination out of them. The list of arcs is sorted in increasing order by the cost of arcs, so that when an arc with too big cost value is encountered, all arcs left can be skipped, as they cannot have lower cost. When the summary cost of the answer is equal *k* and number of arcs used is exactly *m*-1, procedure for reconstructing Eulerian graph is called, what will be described later. Procedure detects moments, when further developing of current answer is of no use, and then exits to give way for another combinations.

When a possible combination is found, all arcs that correspond to arcs in graph K are added to graph G, and instantly procedure for finding Eulerian paths is called. Searching for another combination is resumed only after the procedure is finished.

The construction of the graph from given spectrum is done in the following way:

- a *l*-tuple is taken from the spectrum;
- from each *l*-long piece the first *l*-1 and the last *l*-1 letters are taken;
- then, the already existing graph is searched to check whether any of vertices has identical code as one of the parts taken in previous point;
- if such vertices are found, a connection between them is added, otherwise a new vertex or vertices are added and connected properly;
- if there are any *l*-tuples left in the spectrum, the procedure is repeated;

After the graph has been constructed, the criterion for an existence of the Eulerian path in it is checked. Note that if

$$\forall_{v \in V} out(v) = in(v)$$

then the graph is an Eulerian graph (an Eulerian cycle exists in it). It means that without an information about the starting *l*-tuple it is impossible to find one unique solution, so the case would not be solved by original Pevzner's method. In our implementation it is taken into account. When an Eulerian path is found and we know that a cycle exists in the graph, all possible paths are generated. We just treat the path found as a cycle in which the first vertex in the path follows the last one. So it is enough to generate corresponding DNA sequences starting from each vertex in the path and ending at the preceding one.

If the criterion for an existence an Eulerian path or cycle is satisfied, the searching begins. It is done by a classical algorithm.

Since G is a directed graph, the path would be stored in reversed order. That's why in our case when a vertex is taken out from the stack it is put on the top of another stack. After the procedure has been completed, the vertices are taken from the stack and put into a queue, which holds vertices in the order one has to follow to find Eulerian path in graph G and a DNA sequence that corresponds to it.

When a path has been found, a proper DNA sequence is created. Each of the vertices in created graph has its representation as a sequence of letters. The sequence of the first vertex is taken as the beginning. Then each of vertices adds one letter to the sequence. The process continues and at the end the sequence of demanded length is given.

Of course, the method presented above allows to find only one path if there's no cycle in the graph. It was enough in Pevzner's method, but since we want to generate all possible sequences that are acceptable, we have to check if there is more than one Eulerian path in created graph.

It can be done by modifying the original algorithm. In that algorithm, when one of the connections of the current vertex has been used and removed from the graph and there are more of them, another connection is taken. The order of taking the connections doesn't matter when we want to find only one Eulerian path, but it is important for when generating all possible paths.

Example

Let's suppose we are given:
spectrum = {AAC, ACC, ACT, AGA, CCT, CTA, GAC, TAG},
length of the DNA chain to be reconstructed: n=10.

As we can see, the length of tuples is l=3, and we've got 8 of them. Let's notice:

$$|spectrum| = 10 = n\text{-}l+1,$$

so the defect k=0. The constructed graph:

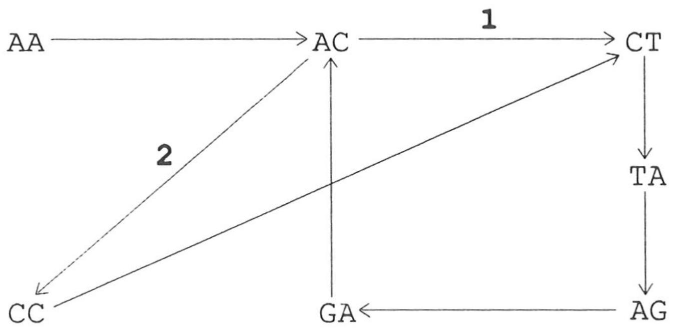

Fig. 5. Two possible Eulerian paths

If we choose the arc marked as 1 as the first one to be taken in the algorithm, the sequence we will find will be AACTAGACCT. Otherwise it will be AACCTAGACT. There are two (and only two) different acceptable sequences.

As we can see, whenever we consider a vertex whose outdegree is greater than 1, there are chances that different order of arcs taken can lead to different sequences in a final solution. Remembering about our goal, which is generating all acceptable solutions, we have to consider every possible order of arcs taken.

When analyzing carefully a behavior of Pevzner's algorithm we see that the following cases cannot be handled in a proper way by this method [5]:

1) One of the ends (an appropriate l-mer) of the sequence is not included in the spectrum. Let us consider the example sequence given in Fig. 6. After the elimination of the *l*-tuple corresponding to arc 1, the information about the first vertex is lost, and cannot be reconstructed by the discussed method (similarity with the last vertex).

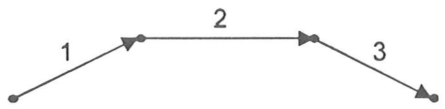

Fig. 6. An example for information loss as described in 1)

2) The algorithm omits vertices having the same number of input and output edges, although the existence of an arc joining such vertices in full spectrum case is possible. In the graph in Fig. 7, if arc 5 is missing, there will be no vertices with different numbers of entering and leaving edges, and graph $K_{m,m}$ will not be constructed. Thus, no solution will be generated.

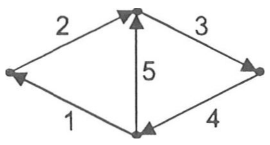

Fig. 7. An example for information loss as described in 2)

3) It is impossible to read the sequence when *l* successive subchains are missing. The maximal number of arcs inserted in succession into the graph is equal to *l*-1 - this is the maximal cost which can be assigned to an edge in graph K. Thus, there is no possibility of inserting missing nucleotides between the existing ones (in the investigated DNA chain).

4) The reconstruction of the missing *l*-tuple when it consists of identical nucleotides is also impossible. The algorithm does not allow for inclusion of the same vertex into both subsets of vertices of the bipartite graph K. Thus, it makes the reconstruction of loops in graph G impossible. („ATTTTCTT" with *l=3* causes defect „TTT" in spectrum, which cannot be detected.)

5) Sometimes the maximal flow in graph K has the cost less than *k*, although none of the above four cases occurs. Pevzner counts such case among unsolvable ones. („GCTTCTTCA" with *l=3* can be an example.)

In order to construct DNA sequence even for the cases when Pevzner's algorithm fails, a new method has been proposed [3][4], It is briefly described in the next section.

3. The new method

The proposed method is rather a brute force one but thanks to the structure of the data it is expected to terminate fast in case spectrum is complete, i.e., |spectrum| = n-l+1. In case it is not (|spectrum| = n-l+1-k, k>0), the running time of the method is longer but it always generates a correct sequence (in case only one is possible) or sequences.

The basic idea behind it is as follows. Suppose k=0. Then a complete tree of all possible sequences of length *n* is started to be built. The tree is constructed in the depth first search way. At each level *m* there are 4^m nodes, due to 4 nucleotides present in DNA. As soon as it is possible (at level *l*) one tries to cut off (not to construct further) these branches which contain fragments not appearing in spectrum. Computational experiments have shown that in the case when |spectrum| = n-l+1 and the sequence can be reconstructed unambiguously (in some cases, although the data are complete, due to the data structure the original sequence cannot be reconstructed unambiguously) an average depth of the tree built (except for the correct branch) is close to *l*. Suppose now that there are *k* defects in spectrum. The process of building the tree changes now (although the general idea of the method remains the same). The origin of the defects - whether introduced by repeated fragments or resulting from experimental errors - does not have any influence on the method. The described method deals correctly with any type of false negative errors. Constructing the tree one allows each branch to contain at most *k* fragments of length *l* not included in spectrum. Then each branch is checked against this property starting at level l+k.

4. Computational experiments

We have tested both methods using 40 samples of DNA sequences taken from the GenBank by splitting them to [8-12]-tuples (generating full spectrum containing each l-tuple only once). The experiments have been carried out on SGI Power Challenge at Poznań Supercomputing and Networking Center. We have used DNA sequences of length 100, 200, 300 and 400. The new method, described in Section 3 has been also tested and the results of experiments are gathered in [3],

Below both methods are compared. Only cases with number of solutions > 0 are taken into consideration (the others were not correctly solved by the algorithm).

The timing

The main advantage of Pevzner's method is that it reduces the problem of finding a correct DNA sequence to the Eulerian path finding problem, for which a polynomial in time algorithm exists. However, finding all Eulerian paths in the constructed graph G and all flows of a given cost in graph K makes the time complexity much worse. Time of computations increases exponentially with the increase of *n* and *k* and the decrease of l. In spite of that, the time results are much better than those of the new method.

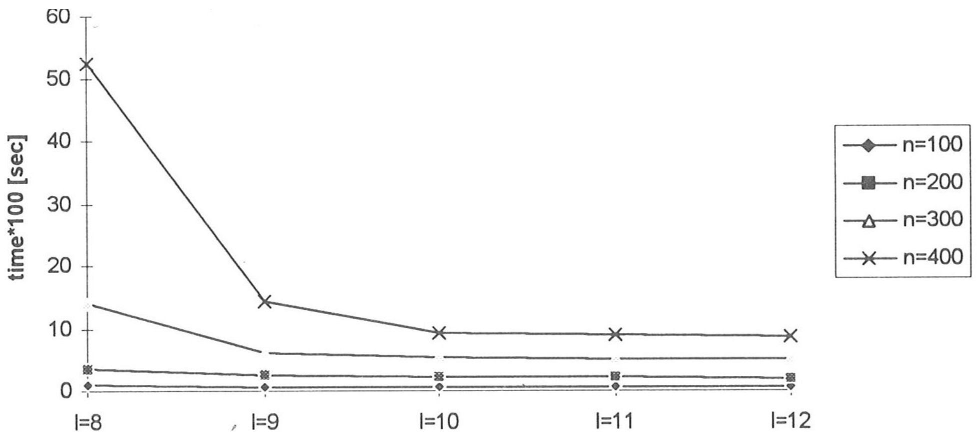

Fig. 8. Average time used by the Pevzner's method vs. the length of the oligonucleotides. The value of *k* is equal to 0.

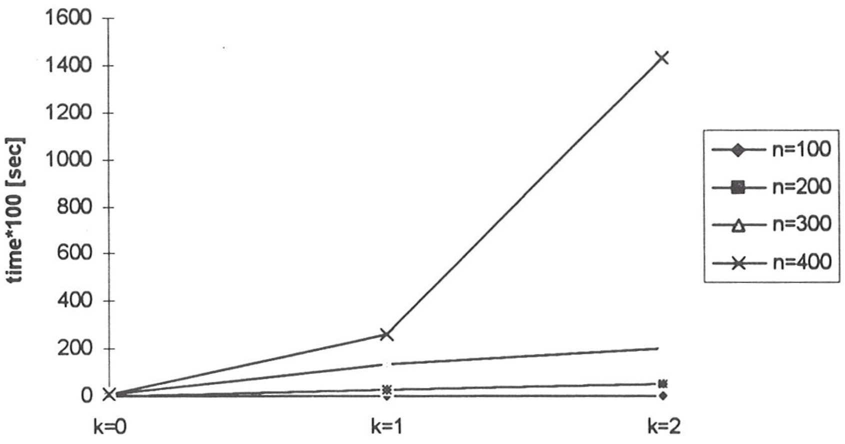

Fig. 9. Average time used by the Pevzner's method vs. the value of defect.

In Figures 8 and 9 the average behaviour of the Pevzner's method is given. They do not show any comparisons with the new algorithm because of the big difference between the two methods (e.g. the average time for the new one, for n=400, *l*= 12 and k= 0, was 70 second, while the time for the Pevzner's one was 0,09 second). But the nature of the time functions is the same. We can safely assume that the Pevzner's method is much better than the new method as far as time is concerned.

The efficiency

When the new method generates a solution, it is always complete. But waiting for computational results may be time-consuming. The Pevzner's method generates results quickly but - as opposed to the new algorithm - it is not clear when given solution is correct. In the performed tests it either found all possible solutions or none. Below both methods are compared. The efficiency for the new method means percentage of not-much-time-consuming cases, for the Pevzner's one - percentage of properly solved cases.

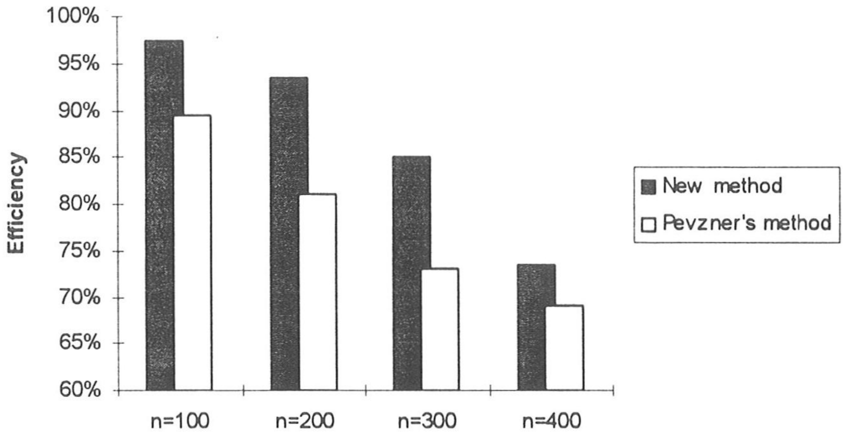

Fig. 10. The average efTicicncy of Pevzner's method in comparison to the new one.

The chart on Fig. 10 shows the average efficiency of the Pevzner's method. The number of cases where the solution was found decreases with increasing n, as it is shown. As we can see, Pevzner's method may become useless for sequencing long DNA chains due to its descending efficiency, although it gives good results for n lower than 200.

On the basis of the obtained results it can be assumed that if, using our modification of Pevzner's method, a solution is gained, it can be treated with big probability as a complete one. However, when no solution is found, we cannot be sure that there is not any. It can be noticed that in most cases when the Pevzner's method gave no answer the new method worked very long. This leads us to the conclusion that the new method is most useful for verifying the results of Pevzner's method. Reducing the time consumption of the new method would make it probably the best DNA sequencing method known so far.

References

1. Bains, W. 1991. Hybridization Methods for DNA Sequencing. *Genomics* 11, 294-301.
2. Bains, W., and Smith, G.C. 1988. A Novel Method for Nucleic Acid Sequence Determination. J. *Theor. Biol.* 135, 303-307.
3. Błażewicz, J. (ed.) 1995. Szeregowe i równoległe algorytmy sekwencjonowania łańcuchów DNA. *Report, Poznań Supercomputing and Networking Center* 1.
4. Błażewicz, J., Kaczmarek, J., Kasprzak, M., Markiewicz, W.T., and Węglarz, J. 1996. Sequential algorithms for DNA sequencing. *Computational Methods in Science and Technology, Poznań Supercomputing and Networking Center* 1, 31-42.

5. Błażewicz, J., Kaczmarek, J., Kasprzak, M., Markiewicz, W.T., and Węglarz, J. 1996. A note on *l*-tuple DNA sequencing in incomplete Spectrum case. Submitted to *J. Comput. Biol.*
6. Drmanac, R., Labat, I., Brukner, I., and Crkvenjakov, R. 1989. Sequencing of Megabase Plus DNA by Hybridization: Theory of the Method. *Genomics* 4, 114-128.
7. Khrapko, K.R., Lysov, Y.P., Khorlyn, A.A., Shick, V.V., Florentiev, V.L., and Mirzabekov, A.D. 1989. An oligonucleotide hybridization approach to DNA sequencing. *FEBS Letters* 256, 118-122.
8. Pevzner, P.A. 1989. *l*-Tuple DNA Sequencing: Computer Analysis. *J. Biomol. Struct. Dyn.* 7, 63-73.
9. Southern, E.M., Maskos, U., and Elder, J.K. 1992. Analyzing and Comparing Nucleic Acid Sequences by Hybridization to Arrays of Oligonucleotides: Evaluation Using Experimental Models. *Genomics* 13, 1008-1017.