Sequential algorithms for DNA sequencing

Jacek Błażewicz, Janusz Kaczmarek, Marta Kasprzak, Jan Węglarz

Institute of Computing Science, Poznań University of Technology
ul. Piotrowo 3a, 60-965 Poznań, Poland
{blazewic | janusz | marta | węglarz}@cs.put.poznan.pl

Wojciech T. Markiewicz

Institute of Bioorganic Chemistry, Polish Academy of Sciences
ul. Noskowskiego 12/14, 61-704 Poznań, Poland
markwt@ibch.poznan.pl

Abstract

Reconstruction of the original DNA sequence in sequencing by hybridization approach (SBH) due to a large number of possible combinations requires a computational support. In the paper, a new method of sequencing has been proposed. Two algorithms based on its idea have been implemented and tested: for the case of an ideal hybridization experiment (complete data) and for more general case, when some data are missing, like in the real experiment. Authentic DNA sequences have been used for testing. A parallel version of the second algorithm has been also implemented and tested. The quality of the reconstruction is satisfactory for the library of oligunucleotides of length 9, and 100, 200 and 300-bp long sequences. A way to a further decrease of the computation time is also suggested,
keywords: DNA sequencing, sequencing by hybridization, oligonucleotide hybridization, parallel computing

1. Introduction

A recognition of DNA sequences, as a part of a wide stream of activities aiming to complete our knowledge about biochemical basis of life, attracts researchers from various domains. Three main fields of activities can be distinguished in this area: DNA sequencing aiming to discover the exact sequence of nucleotides in a relatively short DNA segment (100-300 bp long), DNA assembling - which assembles the sequenced fragments into longer (1000-10000 bp long) contigs and DNA mapping - where one deals with whole chromosomes and tries to place marked DNA fragments (usually genes) on certain chromosome region. This paper deals with DNA sequencing. Our method is based on hybridization experiment, several versions of which have been proposed, e.g., by Drmanac et al. 1989, Khrapko et al. 1989. Southern et al. 1992 and Markiewicz et al. 1994.

Before presenting the method let us set up the problem more precisely. In general, the hybridization experiment is based on a property of single stranded nucleic acids to form a complex (antiparallel) with a complementary strand of nucleic acid. Shorter fragments of nucleic acids (oligonucleotides) are used in hybridization experiments and thus, a formation of a complex indicates the occurrence of a sequence complementary to oligonucleotide in nucleic acid. As a result of the experiment we obtain a set *S* of all *l*-long oligonucleotides which are known to hybridize with the investigated DNA sequence *N* of length *n* (i.e., they are substrings of string *N).* In case of ideal data we have thus $|S| = n - l + 1$. In some cases, however, we can obtain less then n -*l* +1 fragments, either due to experimental problems or as a result of the structure of sequence *N* (for example, it can contain repeated subsequences). Moreover, we assume no positive errors, i.e., each fragment (oligonucleotide) present in *S* is a part of the original sequence. The goal is to reconstruct the original sequence from the given set *S*.

Till now, several methods for assembling the original sequence have been proposed, e.g., by Khrapko et al. 1989, Pevzner 1989, and Bains 1991. Lysov suggests looking for a Hamiltonian path in a special graph (its vertices are elements of 5, an edge between vertices exists if and only if they overlap on length *l*- 1). Since the problem of the Hamiltonian path is known to be NP-hard (thus unlikely to admit polynomial time algorithms), Pevzner proposes an interesting transformation of such a graph. As a result of the transformation one gets a new graph in which the Eulerian path is looked for. This establishes the complexity of the problem since the problem of the Eulerian path can be solved in polynomial time. However, the Pevzner's method cannot deal correctly with cases when some data are missing. Bains' proposal is most similar to ours, but also does not take care of missing data. All these papers analyze sequential approaches, whereas significant speed-up can be obtained by applying parallel procedures.

In this paper a new method for DNA sequencing is proposed. Two algorithms based on its idea have beem developed and implemented: for the case of an ideal hybridization experiment (complete data) and for more general case, when some data are missing, like in the real experiment. A parallel version of the second (more general) algorithm is also discussed.

The structure of the paper is as follows. In Section 2 we present the sequential versions of the method for ideal data cases and for cases when data are incomplete, respectively. A modification of the method for long *l* is also presented. Some remarks on the implementation, test results and their analysis are included in Section 3. Section 4 contains a description of the parallel implementation of the second algorithm.

2. The method and its sequential implementation

Our method is rather a brute force one but thanks to the structure of the data it is expected to terminate fast in case when set *S* is complete, i.e., $|S| = n - l + 1$. In case it is not ($|S| = l - n + 1 - k, k > 0$, where A: is a number of oligonucleotides present in *N* missing in *S*), the method behaves less optimistically but 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 starttet 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 /) one tries to cut off (not to construct further) these branches that contain fragments not appearing in *S*. Computational experiments have shown that in the case when $|S| = n - l + 1$ 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 I.

The algorithm below (presented in a Pascal like language) describes this idea in a more detailed way. Initially the parameters of the procedure *Reconstruct* are as follows: S — the set of fragments obtained from the experiment. *Sequence* — the reconstructed sequence (initially empty), *level* — current level in the tree (initially 1). The result (or results, when the reconstruction is ambiguous) is output in the line **output** *[Sequence).*

Algorithm 1: the case of complete data (k = 0)

```
Reconstruct (S, Sequence, level)
begin
  for c in ['A', 'C', 'G', 'T'] do
  begin
    Sequence := Sequence + c;
    if level < 1 then
      Reconstruct (S, Sequence, level + 1);
    else
    begin
      if fragment of last 1 nucleotides
                           in Sequence = any x in S then
        if level = n then
          output (Sequence);
        else
          Reconstruct (S - x, Sequence, level + 1);
    end
    Sequence := Sequence - c;
  end
end
```

In the second case, where there are *k* defects in set *S*, i.e., $|S| = l - n + 1 - k$, the process of building the tree changes (although general idea of the method remains the same). Constructing the tree one allows each branch to contain at most *k* fragments of length *l* not included in *S*. Then each branch is checked against this property starting at level l. Thus, one may implement this idea in the form of Algorithm 2. (The parameter i is the number of accepted fragments not present in S.)

Algorithm 2: the case of incomplete data (k > 0)

```
Reconstructl (S, Sequence, level, i)
begin
  for c in ['A', 'C*, 'G', 'T'] do
  begin
    Sequence := Sequence + c;
    if level < 1 then
      Reconstructl (S, Sequence, level + 1, i);
    else
    begin
      if fragment of last 1 nucleotides
                            in Sequence = any x in S then
        if level = n then
          output (Sequence);
        else
          Reconstructl (S - x, Sequence, level + 1, i);
      else if i < k then
        if level = n then
          output (Sequence);
        else
          Reconstructl (S, Sequence, level +1, i + 1);
    end
    Sequence := Sequence - c;
  end
end
```

Let us illustrate these two algorithms with simple examples.

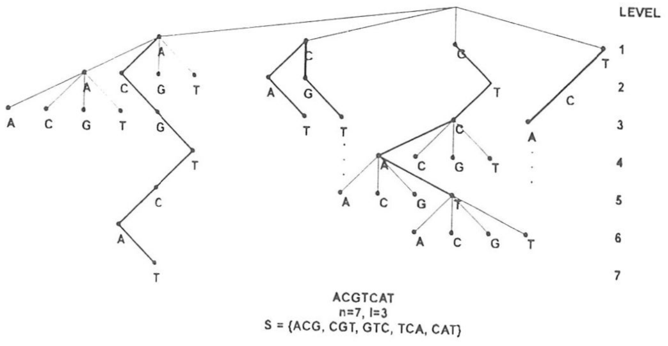

Figure 1. An exemplary tree for the original sequence ACGTCAT (Example 1) as constructed by Algorithm 1

Example 1:

In Fig. 1 a process of reconstructing a DNA sequence from the set of oligonucleotides *S* = **{ACG, CGT, GTC, TCA, CAT},** where l = 3, is presented. Now set *S* is complete, i.e., *k* = 0, and the original sequence **ACGTCAT** can be unambiguously reconstructed. Other branches has been cut off.

Example2:

Let us now consider the case when number *k* of oligonucleotides missing in set S is greater than 0 (cf. Fig. 2). We assume *N* = **ACGTCAT,** S = **{ACG, GTC, TCA, CAT},** *n* = 1, l = 3, *k* = 1. Let us trace how exemplary two branches of the tree have been constructed. First let us notice that *k* = 1, e.g., one and only one oligonucleotide of length l - 3 may be inserted while a branch is constructed. Consider branch **AACG.** The oligonucleotide **AAC** may be accepted. It does not appear in set S, but one is allowed to accept one oligonucleotide (since *k* = 1) not appearing in *S*. Introduction of such an oligonucleotide is marked as ①. Next, **ACG** is chosen from S since it is the only oligonucleotide in *S* which may be appended to the **AAC.** One is no longer allowed to choose any oligonucleotide from the outside of *S*. Moreover, it is no longer possible to extend the branch **AACG** since no oligonucleotide from *S* may be appended to this sequence. Now, consider branch **ACGTCAT** that is the only correct result of the reconstruction. First **CGT** (①) has been appended to **ACG** as the missing oligonucleotide. Thus, the limit of oligonucleotides from the outside of *S* has been exhausted. From now on only oligonucleotides from *S* may be appended to the constructed sequence. The following oligonucleotides have been appended in order: **GTC, TCA, CAT** resulting in the original sequence **ACGTCAT.** Other branches of the tree can be also considered although the tree presented in the Fig. 2 is not complete.

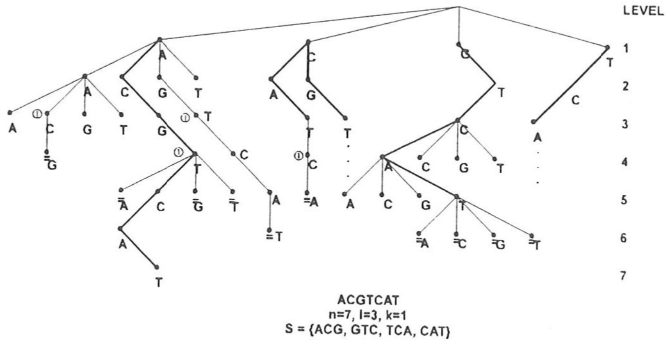

Figure 2. A tree for Example 2 (incomplete data)

As one can notice, the most time consuming part of the process is a construction of the complete tree down to level l (l + *k* in case of *k* defects). Thus, in order to speed up the computation, the starting fragment of the sequence to be reconstructed should be known. In fact, it is often known - especially when it was used as a primer (or part of it) in PCR amplification (Saiki et al. 1988). If, for some reasons, the starting fragment is not known, one could modify the algorithm. Branches of the tree before they reach level l would be cut off. At each level between k and $l + k$ one would have to check if the fragment generated can be a pail of any element of S. In other words, one would not wait until level $l + k$ is reached but would try to cut off the superfluous branches earlier.

3. Results of the computational experiment

A computational experiment has been carried out to evaluate our method from the computational point of view. We have rather tried to experimentally find out dependencies between various parameters than to perform a theoretical analysis. First, we have used real DNA fragments that are known to be less random than stochasticaly generated sequences. This causes all methods to behave worse than in case of randomly generated DNA. To test our method we have used 100, 200 and 300 bp long fragments taken from 40 various real DNA sequences. All the sequences come from GenBank (a database of investigated DNA fragments maintained by the National Institute of Health); the accession numbers are given in the appendix. From each sequence corresponding data sets S (as the results of a theoretical hybridization experiment) for $l \in [8,12]$ have been generated. Each point on charts presented below is an average of 40 results obtained for various sequences. All the results presented has been obtained on HP Apollo 9000/750 workstation in the Poznań Supercomputing and Networking Center.

When analyzing data from the GenBank we realized firstly, that the number of "bad" sequences, i.e., sequences for which $k > 0$ as a result of repeated fragments, not exactly meets the formula presented by Southern et al. 1992. This formula describes the probability with which all oligonucleotides occur in the sequence only once:

$$\log p_r \approx -\frac{r(r-1)}{2t}, \text{ if } r << t$$

where r = n - l + 1 is the number of oligonucleotides, 1 = 4^l.

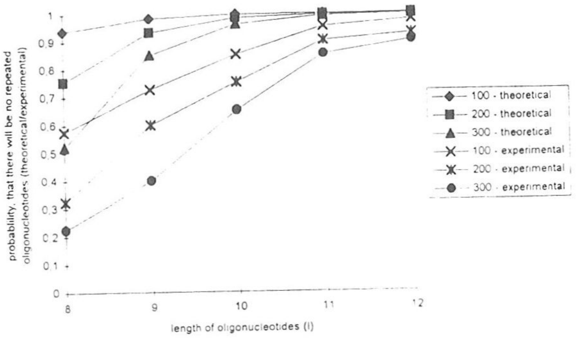

Figure 3. Comparison of the probability of the occurrence of repeated fragments in DNA sequences against theoretical estimations

The sequences we have used behave worse than expected because of the presence of repeated fragments. In Fig. 3 the behaviour of real DNA sequences against theoretical estimation is compared. The upper three lines in Fig. 3 illustrate the theoretical probability calculated from the above formula. The corresponding lower three series are calculated from real sequences probabilities. The real DNA sequences contain repeated fragments more frequently than we would expect. It follows that in real hybridization experiments defects will occur quite often.

The first set of experiments has been conducted for the case of no defects, i.e., $k = 0$. Suprisingly, although the full tree size grows exponentially, the time used by Algorithm 1 grows linearly only. This is illustrated in Fig. 4, where in Fig. 4a a global dependency of the computation time on the length of the original sequence (n) and the length of oligonucleotides (l) and in Fig. 4b the results for l = 8, 9, 10, are depicted more precisely.

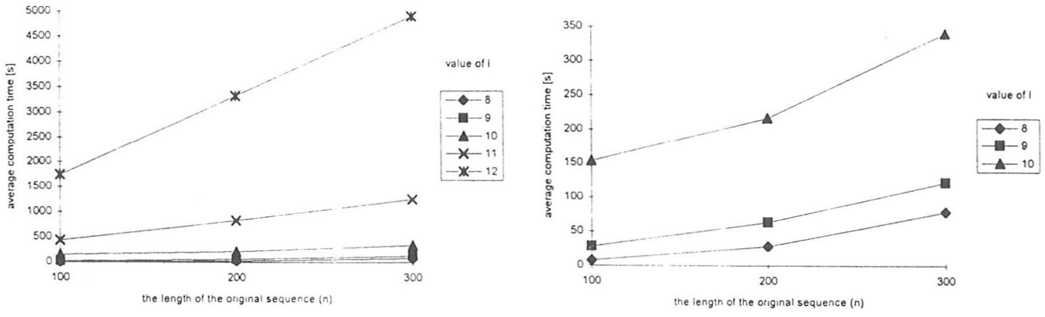

Figure 4a. 4b. Computation time of Algorithm 1 vs. the length of the original sequence (n) and the length of oligonucleotides (1)

In case of an existence of some defects one could expect that increasing k would cause our algorithm to consume exponentially more time. This is because the number of branches which cannot be cut off at level m is 4^m. In case of $k > 0$ the complete tree has to be constructed down to level $l + k$. The experiment has proved the above hypothesis. The increase of computation time in such a situation is presented in Fig. 5.

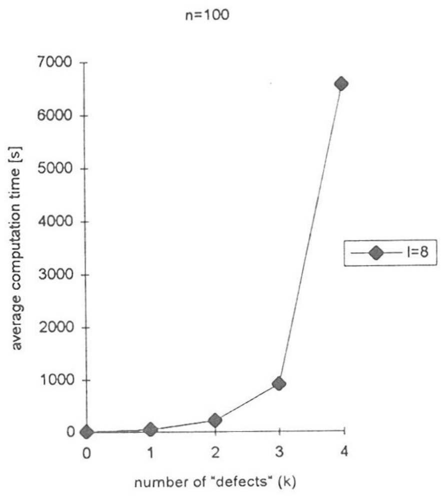

Figure 5. An impact of the number of "defects" k on the computation time for length n = 100

Another interesting factor measured by us concerns an average value of a number of defects (resulting from the structure of the sequences), i.e., $\overline{k}$. Of course. one is interested in possibly low $\overline{k}$, since bigger k is, longer branches in the tree one have to consider and bigger the computation time is. Figure 6 illustrates the dependency of $\overline{k}$ on different values of n and l. One can observe, that by increasing

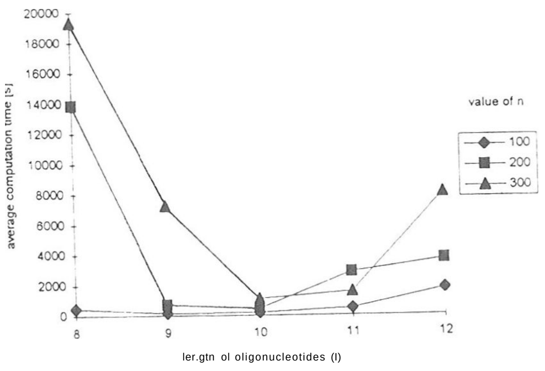

Figure 6. An average computation time (in sec.) vs. the length of oligonucleotides (for various lengths n of a reconstructed sequence)

length l of oligonucleotides one obtains a smaller $\overline{k}$ of oligonucleotides missing in S. This should result in decreasing average computation time. The observed behaviour of the average computation time is however somewhat suprising. As illustrated in Fig. 7 the average computation time (vs. the length of oligonucleotides l for various lengths n of a reconstructed sequence) initially decreases, what corresponds to the decreasing $\overline{k}$ (cf. Fig. 6). The following increase of the computation time corresponds in turn to the situation, where the increasing length of oligonucleotides l has no longer effect on the value of k but the depth of the tree to be built completely (down to the level $l + k$) still increases.

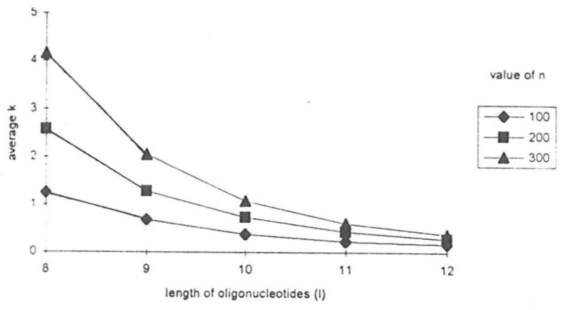

Figure 7. Increasing the length of oligonucleotides 1 results in lower average value of k

A very important factor in sequence reconstruction is ambiguity. One is usually interested in a selection of parameters assuring unambiguous reconstruction and at the same time possibly a limited experimental effort. Curves in Fig. 8 indicate that oligonucleotides of length 9 guarantee satisfactory results in most cases, while 10-base long oligonucleotides assure a very good reconstruction.

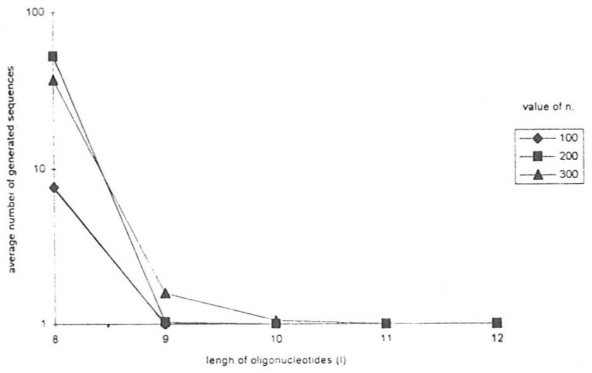

Figure 8. Number of generated sequences vs. the length of oligonucleotides (1) and the length of sequences (n)

The results of our experiments indicate, that a suitable length of oligonucleotides in a library used in the hybridization experiment would be 9 for 200 and even 300-base long sequences. For shorter fragments oligonucleotides of length 8 could be used but this can lead to an ambiguous reconstruction.

4. Parallel implementation of the method

Parallel implementation of the method can be constructed based on the sequential version described above. Below we present an outline of this strategy.

At certain level m of the tree one splits the computations of the algorithm among a number of processes. Splitting the work at level m results in generating 4^m chunks which can be treated independently from each other. The value of m should be less than l, otherwise most of the branches can be already cut off before reaching level m. A suitable number of potentially identical (regarding their processing times) processes facilitates the proper load balancing of the parallel system (i.e. a dynamic assignment of processes to processors in such a way that the processing time of the whole job is minimized).

The load balancing strategy we have used is very simple, but leads to satisfactory results. The whole computation process is controlled by a single process called master. It builds the tree down to level m and, after that, sends requests to slave processes. Each of the slave processes performs only one chunk of work at a time. Once it is ready with its task, it sends the results to the controlling master process. The master process sends the next piece of work to the free slave process if the computation is not yet completed.

During computational experiments we have tested the parallel version of Algorithm 2 (as the more general one). It has been developed using PVM system (Geist et al. 1994) and implemented on HP Apollo 9000/750 workstation. For the evaluation of an experiment we have not used a real distributed heterogeneous system, like usual PVM based systems, because the estimation of the speed-up and efficiency would not be simple in that case. Instead, we have run several PVM daemon processes on a single machine. Then the speed-up has been computed as the quotient of the CPU time of the sequential version of the algorithm divided by the maximum CPU time used by the slave processes plus the master process CPU time. We did not use the wall time to determine the speed-up because it is strongly connected with the current load of the machine. On a dedicated system we expect the results to be similar to these obtained in simulation assuming we can omit the communication time.

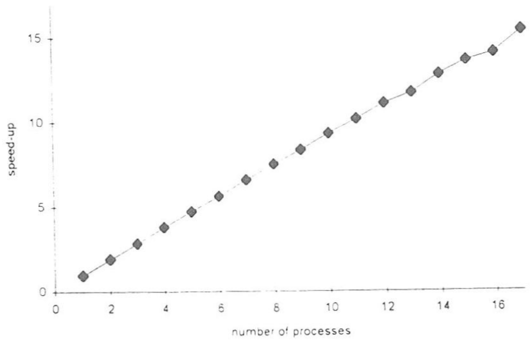

Figure 9. Speed-up obtained in simulation of the parallel version of the algorithm _

Figure 9 illustrates the results of the parallel simulation. Each point in the Fig. 9 and 10 represents an average value of 40 runs for different data sets. As one can notice, the speed-up obtained is nearly linear and does not depend on the number of processes applied. Such a good result has been obtained thanks to the proposed load balancing algorithm.

In Fig. 10 the efficiency (defined as the quotient of the speed-up divided by the number of processes performing the task) of the parallel version of the algorithm is presented. It goes worse with the increasing number of processes but remains still above 87%. This result was obtained by breaking the workload in a proper number of pieces making the efficient load-balancing possible.

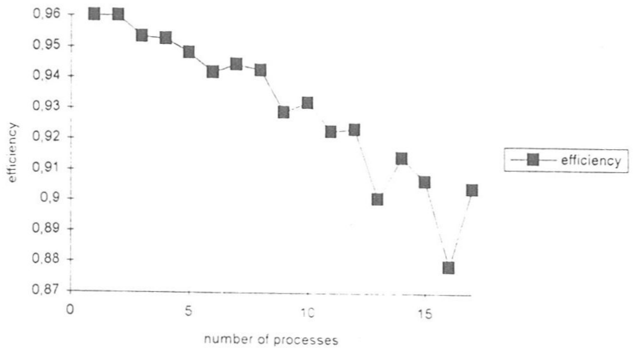

Figure 10. The efficiency of the parallel algorithm

5. Conclusions

We have presented a sequential and a parallel implementation of a new method for DNA sequencing from the data that come as a result of the hybridization experiment. The presented method finds all possible sequences that can be reconstructed from the provided data even if the data are incomplete. Having done a lot of computational experiments a reasonable trade-off between experimental and computational effort and unambiguity of reconstructed sequence has been pointed out.

Future work should be concentrated on two subjects. Firstly, an efficient method dealing with ambiguously reconstructed sequences should be developed. An interaction with existing knowledge about DNA structures could be used at the first stage. Experimental feedback at the second stage cannot be omitted. Secondly, the implementation of the parallel version of the method in real (heterogeneous, non-dedicated) environment should be investigated in more details.

References

Bains. W., and Smith, G.C. 1991. A Novel Method for Nucleic Acid Sequence Determination. *J. theor. Biol.* 135, 303-307

Bains. W. 1991. Hybridization Methods for DNA Sequencing. *Genomics* 11, 294-301

Geist. A., Beguelin, A., Dongarra, J., Jiang. W.. Manchek. R. and Sunderam, V. 1994. *PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing.* MIT Press

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

Khrapko, K.R.. Lysov, Yu.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

Markiewicz. W.T., Adrych-Rożek, K., Markiewicz, M.. Żebrowska, A. and Astriab, A. 1994. Synthesis of oligonucleotides permanently linked with solid supports for use as synthetic oligonucleotide combinatorial libraries. *Innovations in Solid Phase Synthesis, 1994: Biological and Biomedical Applications* (Ed. R. Epton). Mayflower Worldwide. 339-346.

Pevzner. P.A. 1989. 1-Tuple DNA Sequencing: Computer Analysis. *J. of Biomolecular Structure & Dynamics* 7. 63-73

Saiki. R.K., Gelfand. D.H.. Stoffel, S., Scharf. S.J., Higuchi. R., Horn. G.T., Mullis. K.B., and Erlich. H. 1988. Primer-Directed Enzymatic Amplification of DNA with Thermostable DNA Polymerase. *Science* 239. 487-491

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

Appendix

In Appendix the GenBank accession numbers of 40 sequences used for testing are listed.

1) D00723
2) D11428
3) X58377
4) X13440
5) X56088
6) X00351 J00074 M10278
7) X02994 K00509 K02567 M11816 M11817 M11818 M11819 M11820 M11821 M11822 M11823 M11824 M11825 M11826 X00427
8) X04350
9) X03350
10) X01098 K01177
11) Y00264
12) X53279
13) Y00649
14) X07577
15) X03663
16) X51841
17) X02160
18) X07173
19) X04772
20) Y00503
21) X07696
22) X14758
23) X03444
24) X03445
25) X02874 K00006
26) X13403
27) X06985
28) Y00093
29) X15610
30) X52104
31) X13097
32) X04217
33) X04808
34) X06374
35) X03795
36) X04741
37) X04412
38) X14034
39) X07743
40) X57398