A package of fast tools for genomic sequence analysis

Boris J. Steinberg, Jumana M. Abu-Khalil, Mikhail G. Adigeyev, Andrey A. Bout, Anton V. Kermanov, Evgeny A. Pshenichnyy, Galina V. Ramanchauskayte, Alena P. Kroshkina, Alexandr V. Gutnikov, Natalia S. Ponomareva, Anatoliy E. Panich and Tatiana P. Shkurat

Abstract — Here we present a current progress towards the develepment of nucleotide sequence analysis software package produced as a result of bioinformatics initiative at SFedU. Different modules included in the package allow to perform wide range of sophisticated operations including sequence alignment, motif search and *de novo* discovery, primer design etc. The particular advantage of the package modules is the reduced time of task performing coupled with optimization for memory usage both leading to improved performance. The latter was reached due to using special data structures, parallel running, etc. Widely used in bioinformatics pairwise alignments are represented by two pairwise alignment algorithms (parallel block and parallel block with optimal memory usage) adjusted to running on multi-core processors and accelerators. The performance tests have confirmed that the former one is faster than Needleman-Wunsch algorithm by ~60% and EMBOSS tool by ~30% . The latter one aligns long sequences faster than EMBOSS Stretcher by 40%. The motif discovery time was decreased tenfold for some conditions when Gibbs sampling method with truncated generalized suffix trees was introduced into algortihm. A VP-tree based approach offered for locating of genome origins of replications allowed to lower time complexity comparing with the naïve algorithm.

***Keywords*—** Bioinformatics, Dynamic programming, Suffix tree, Vp-tree.

This research was supported by project of the Ministry of Education and Science of Russia №6.703.2014/K «The search of new targets for predictive diagnostics of the reproductive system diseases». Analytical work was carried out on the equipment of Center for collective use of Southern Federal University «High Technology», grant RFMEFI59414X0002.

B. J. Steinberg is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, Rostov-on-Don, Russia (e-mail: borsteinb@mail.ru).

J. M. Abu-Khalil is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, Rostov-on-Don, Russia (e-mail: jumana.abukhalil@gmail.com).

M. G. Adigeyev is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, Rostov-on-Don, Russia (e-mail: madi@math.sfedu.ru).

A. A. Bout is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, Rostov-on-Don, Russia (e-mail: a-bout@yandex.ru).

A. V. Kermanov is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, and Research Institute for Plague Control, Rostov-on-Don, Russia (e-mail: av-kermanov@mail.ru).

E. A. Pshenichnyy is with the Southern Federal University, Research Institute of Biology, Rostov-on-Don, Russia (e-mail: pshenichniy.eugene@gmail.com).

G. V. Ramanchauskayte is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, Rostov-on-Don, Russia (e-mail: galinka@lastbit.com).

A. P. Kroshkina is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, Rostov-on-Don, Russia (e-mail: kroshkina.alena@yandex.ru).

A. V. Gutnikov is with the Southern Federal University, Institute of Mathematics, Mechanics and Computer Science, Rostov-on-Don, Russia (e-mail: zeratul477@gmail.com).

N. S. Ponomareva is with the Southern Federal University, Institute of high technology and piezotechnique, Rostov-on-Don, Russia (e-mail: nsponomareva@sfedu.ru).

A.E. Panich is with the Southern Federal University, Institute of high technology and piezotechnique, Rostov-on-Don, Russia (e-mail: panich@sfedu.ru).

T. P. Shkurat is with the Southern Federal University, Research Institute of Biology, Rostov-on-Don, Russia (e-mail: tshkurat@sfedu.ru).

I. INTRODUCTION

Bioinformatics is an interdisciplinary field where the close collaboration is required between mathematicians, computer scientists and biologists. The rapid growth of data, mostly genomic, due to appearance of high throughput sequence technologies (HST) demands efficient algorithms for fast biosequences processing. Such algorithms are usually based on some sort of smart data structures optimized for processing very long strings. In this paper we describe several software tools based on two popular techniques – dynamic programming and tree-like structures (generalized truncated suffix tree, vp-tree). Some of the described tools were presented in [40] and are available via web interface at mmcs.sfedu.ru/bio/ (Russian version).

II. Sequence Alignments

A. Problem Definition

One of the most important bioinformatics problems is nucleotide sequence alignments [41],[42]. Global alignment is applied for the analysis of conservative parts of sequences, for pinpointing sequence relations and usually is the basic step in molecular phylogenetics inference [1]. The problem of pairwise global alignment can be stated as follows [2]: given a pair of sequences, build a two-row matrix such that the rows contain the characters of the sequences in order, interspersed with some spaces. Each alignment is assigned a numerical characteristic called 'score'. The score reflects the degree of similarity of the sequences. The problem is to build a maximum score alignment. The definition of the pairwise local alignment problem is essentially similar to the definition of global alignment, but the goal is to find a pair of substrings, one in each sequence, that maximizes the score. Local alignment is used for detection of similar fragments within functionally related sequences. A number of dynamic programming algorithms have been designed to find global (Needleman-Wunsch algorithm [3]) or local alignments (Goad and Kanehisa [4], Sellers [5], Smith and Waterman [3], Waterman and Eggert [6], Hall and Myers [7]). There are fast local alignment algorithms, such as BLAST [8] reducing the amount of alignment time at the cost of exactness.

B. Global Alignments

Our package includes several alignment procedures. The basic data structure of dynamic programming algorithms is similarity matrix. A similarity matrix is an (m+1) by (n+1) matrix where m and n are the lengths of the sequences to be aligned. Such model presents serious challenges for efficient parallel execution on present computers.

The basic idea of our algorithm is to divide the similarity matrix into blocks and then apply anti-diagonal approach. This algorithm uses less amount of memory than Needleman-Wunsch algorithm, as it does not save the similarity matrix as a whole. Another our procedure for global alignment is based on [13] and the hyperplanes method [14]. It performs alignment faster than Hirschberg's algorithm [3] and uses less memory than Needleman-Wunsch algorithm. We have performed a quantitative comparison of the two designed algorithms and other dynamic programming algorithms for optimal pairwise global alignment: Needleman-Wunsch, Hirshberg's algorithms and Myers and Miller algorithm [15], as well as other alignment tools: EMBOSS Stretcher [16], based on rapid modification of Myers and Miller algorithm, and Ngila [17], implementing a classic Miller and Myers algorithm. We used an Intel(R) Core(TM) i7 CPU @ 1.6 GHz computer with 4 GB RAM and 4 cores.

Tables 1 and 2 summarize the results of the tests. The columns correspond to algorithms: 'H' for Hirshberg's algorithm, 'NW' for Needleman-Wunsch algorithm, 'ES' for Emboss Strecher and 'N' for Ngila algorithm. The columns 'PAOM', 'PBAOM' and 'PB' correspond to procedures within our package: parallel algorithm with optimal memory usage, parallel block algorithm with optimal memory usage and parallel block algorithm respectively.

Table 1. Calculation times for global alignment procedures

Seq. length (bp)	Time (s)						
	H	NW	ES	N	PAOM	PBAOM	PB
2753 2517	0.4	0.2	0.082	0.27	0.3	0.2	0.1
8376 7488	3.3	1.3	0.53	2.41	1.2	0.8	0.5
268032 239616	2015.7	out of memory	584.9	14109	639.0	362.7	171.1

Table 2. Memory usage for global alignment procedures

Seq. length (bp)	Memory usage (Mb)						
	H	NW	ES	N	PAOM	PBAOM	PB
2753 2517	0.7	27.0	3.95	13.9	1.3	1.4	1.4
8376 7488	0.9	240.0	4.2	121.3	2.9	3.0	5.4
268032 239616	11.2	out of memory	11.6	240.4	66.0	67.0	493.2

The results in Table 1 show that the block algorithm is faster than Needleman-Wunsch algorithm by ~60% and Hirshberg's algorithm by ~80% on tested sets. The table shows that the block algorithm is faster than EMBOSS Stretcher by ~30%. The parallel block algorithm with optimal memory usage aligns long sequences faster than EMBOSS Stretcher by 40% . Ngila tool is considerably slower comparing with designed algorithms.

The memory usage comparison is presented in Table 2. The table shows that the parallel block algorithm with optimal memory usage requires less amount of memory than Needleman-Wunsch algorithm. The parallel block algorithm is more memory efficient than Ngila tool. The designed algorithms are more memory intensive than EMBOSS

Stretcher.

C. Local Alignments

The parallel block procedure for local alignment is based on the Smith-Waterman algorithm combined with block division and anti-diagonal approach. The Parallel Block Local Alignment Algorithm with optimal memory usage is a combination of parallel block local alignment algorithm and the parallel block global alignment algorithm with optimal memory usage, discussed above.

We have compared the running time and memory usage of two presented algorithms and other algorithms for optimal local pairwise alignment, such as Smith-Waterman and Waterman-Eggert algorithms, as well as other alignment tools: EMBOSS [16] Water, which uses the Smith-Waterman algorithm (modified for speed enhancements), and EMBOSS Matcher, based on Bill Pearson's LALIGN application. The experiments were performed on Intel(R) Core(TM) i7 CPU @ 1.6 GHz computer with 4 GB RAM and 4 cores. The results of running time testing are presented in Table 3.

Table 3. Calculation times for local alignment procedures

Seq. length (bp)	Time (s)				
	SW	PBOM	PB	EW	EM
2753 2517	0.28	0.14	0.06	0.42	0.80
8376 7488	0.61	0.60	0.27	5.05	3.19
268032 239616	out of memory	586.29	231.18	out of memory	2205.91

The columns correspond to the algorithms being compared: SW – Smith-Waterman, PBOM - Parallel Block algorithm with optimal memory usage, PB – Parallel Block algorithm, EW – EMBOSS Water and EM – EMBOSS Matcher. The table shows that both designed algorithms are faster than Smith-Waterman algorithm, the parallel block algorithm is faster than EMBOSS Water by ~95% and Matcher by ~90% on testing sets. The parallel block algorithm with optimal memory usage decrease the time of local alignment as compared to EMBOSS tools by 80%.

Table 4 reflects the memory usage of local alignment algorithms under study. The results of experiments show that EMBOSS Water, as Smith-Waterman algorithm, is memory intensive. The EMBOSS Matcher uses less amount of memory than Smith-Waterman and parallel block algorithms. However, Matcher tool is more memory intensive than parallel block local alignment algorithm with optimal memory usage on short sequences.

Table 4. Memory usage for local alignment procedures

Seq. length (bp)	Memory Usage (Mb)				
	SW	PBOM	PB	EQ	EM
2753 2517	53.63	2.09	1.51	56.86	4.19
8376 7488	479.59	3.78	3.92	482.29	4.71
268032 239616	Out of memory	108.15	155.91	Out of memory	21.84

III. Tree-based Procedures

A number of problems in bioinformatics can be formulated in form of seacrhing a genomic sequence for occurences of a short pattern or looking for frequent patterns under some conditions. It is known [3] that many of such problems can be solved efficiently using tree-based data structures. We have implemented several efficient procedures base on suffix trees and vp-trees.

A. Pattern Search

The suffix tree module of presented bioinformatics package is responsible for solving several problems. Exact and inexact pattern search, palindrome search and motif discovery are among them. All of these problems deal with long sequences of characters of a fixed alphabet ({A,C,G,T} for DNA sequences). The huge amounts of data in bioinformatics require special data structures to be used for providing admissible performance. Among such data structures, suffix trees are considered as most appropriate. We have implemented several variants of suffix tree structure. The experiments show that truncated generalized suffix tree provides the best performance improvement for pattern search (both exact and inexact) and motif discovery problem, whereas for the palindrome search it does not give any speedup [18]. The truncation of tree may be useful for searching specific subsequences in a set of longer sequences [19]. There are different approaches for generalized suffix tree implementation [20], but we have designed a new modification that improves search time in a few cases. Our approach is based on branch and bounds method and could considerably reduce space and time requirements for specific problems. We have proposed a modification for Ukkonen suffix tree construction algorithm [21]. Two additional rules were suggested for decreasing both space and build time. These rules handle cases when maximum depth is reached for nodes. Our procedure uses special rules for updating information in leaf nodes as well.

As far as there are a few publicly available implementations, we have compared the results of our implementation with several software tools (MUMMER [22], SUDS [23], ERa [24]). The comparison demonstrates that our procedures either have a speedup or provide more functionality at the same speed. As an example, Table 5 shows that for exact pattern search our procedure has time complexity approximately equal to Mummer suffix tree procedure which has been declared to be the fastest tool for this problem [22]. Our suffix tree procedure has the additional ability of searching occurrences with mismatches. Results in Table 6 demonstrate some time parameters for our inexact search procedure.

Table 5. The times for exact pattern search

Sequence count/ length	Patterns count/ length	Mummer (s)	Our package (s)
1000/5000	100000/10	3.61	4.48
1000/5000	100000/30	6.82	4.21
1000/100000	100000/10	19.57	25.70
1000/100000	100000/30	23.11	20.21

Table 6. Time parameters for inexact pattern search

Sequence count/ length	Patterns count/ length	Number mismatches	Running time (s)
1000/5000	100/10	2	531.5
1000/5000	100/10	5	981.6
1000/100000	10/30	2	255.1
1000/100000	10/30	12	935.1

B. Motif Discovery

Search of regulatory sequences seems to be one of the most important task taking into account the data about non-coding genome regulatory roles obtained due to ENCODE project [9]. This problem could be formulated as a motif discovery [10]. The motif of choice even if it is known, however, could be subjected to changes and variation leading to other task of pattern search (inexact) for relatively huge DNA datasets in order to find genes joint together into regulatory loop.

Motifs are defined as sequence fragments (patterns) whose occurrences in a given set of sequences are statistically significant. The difference between motif discovery and pattern search problems is that in motif discovery problem the motif (pattern) is not given but should be found ('discovered') as the solution.

There are several mathematical models for motif discovery problem, and different algorithms and software tools tend to use different models. We have used three models, implemented in two software tools within the package. One tool searches a set of biosequences for frequent substrings under certain conditions (see also [43],[44]). The second tool is based on a novel model of generalized motifs, which allows the researcher to account for all or some of the motifs' features — similarity (conservativeness), frequency and locality. The third one searches for *dimers* — motifs which consist of two parts (blocks) located at a certain distance from each other.

Frequent motifs

This tool looks for exact (not heuristic) solutions of the following problem: given a positive set of sequences S = {S1, …, Sn} and a negative (or control) set C = {C1, …, Cm}, find all strings (motifs) of length between Lmin and Lmax such that each of the motifs occurs at least in QP positive sequences and in no more than QN negative sequences. The motif discovery module of our package uses truncated suffix tree [37] for speeding up calculations.

We have compared the performance of our module and the software tool MERCI [25] which uses similar motif discovery model. The results are shown in Table 7. All experiments were performed on dataset with 1000 positive and 10 negative sequences of length 1000, the variated parameters were quorums (QP and QN).

Table 7. The times for motif discovery (motif lengths: 10 – 30)

Q_P	Q_N	MERCI (s)	Our package (s)
100	1	130.91	5.92
500	1	36.14	5.85
1000	1	8.46	5.66
100	5	165.21	5.79
500	5	37.57	6.61
1000	5	11.49	5.78
100	10	164.11	6.70

500	10	34.41	5.91
1000	10	14.12	5.64

Generalized motifs.

Genomic motifs are characterized by several feactures, the most significant of them are conservativenes (the degree of similarity of occurrences of the motif), the number of occurrences of motif in the given biosequences, and the localization (occurrences of a motif are often localized in a particular region of DNA) [34]. The models and algorithms used before for motifs discovery focus just on one or two of these characteristics.

The authors have developed [35] a novel model which generalizes all three characteristics of motifs, developed algorithms and software for searching such generalized motifs. The model is based on the *occurrence indicator* of the motive P in the nucleotide sequence S which is defined as follows:

OI(P,S) = F({w(j) * *score*(P, substr(S,j,l)): j=1,…, |S|-l+1}) (1)

Where *l* is the length of the motif P; substr (S, j, l) denotes the substring S starting at position j and having length *l*, and w (j) represents weighting coefficients defining an expected motif's localization. The function *score*(...) quantitatively evaluates the similarity of a motif to a given substring, and the function F(...) takes a set of numbers as its argument. It was shown that the model covers the most commonly used motif models for appropriate selections for score(…) and F(…).

It is also shown that the developed algorithm can be easily modified to find dimers - structural motifs consisting of two parts arranged at a predetermined distance from each other.

For this generalized model for motifs discovery we developed fast algorithms for motif discovery and evaluation (calculating occurrence indicator). These algorithms are independent of the specific choice of model parameters and use Position Specific Scoring Matrix (PSSM) as a motif representation. The motif discovery algorithm is based on the Gibbs sampling method [36] and utilizes truncated generalized suffix trees [37] for speeding up processing. The algorithm for calculation occurrence indicator is outlined below.

1. Given the set of sequences {s_i: i=1,…,k}, „s-i.:i=1,…,k.construct the generalized suffix tree *T* truncated to depth *l*=|*p*| [37].
2. For a vertex *v* in the tree *T* let λ (*v*) denote the *path-label* of *v* [3], i.e. the concatenation of the labels of arcs along the path from the root of the tree to the vertex *v*. For each leaf *v* let L (*v*) denote a list consisting of pairs of the form (*i, j*) where *i* is the sequence's ID (number), *j* is the position of occurrences of λ (*v*) in the *i*-th sequence).
3. For each leaf of *v* in *T*, calculate values $w_i(v)$ as follows:
$$w_i(v) = \sum_{(i,j)\in L(v)} w(j)$$
4. For all *j* calculate $Q_j(p)$ (occurrence indicator of *p* in the *i*-th sequence) applying operation *F* to the set of products *w*(*v*)**score*(*p*, λ (*v*)) for all the leaves.

In the software implementation formula (1) was represented in the form:

$$Q_i = \alpha * S(pattern_l(i), p) + \beta * f(x,\mu) + \gamma * \sum_{i=1}^{k} w_j(v) \quad (2)$$

Here α, β, γ are parameters specified by the user according to the biological problem, *S* is similarity function that implements a basic algorithm of Gibbs sampling, function *f* evaluates a putative motif with respect to its location in the set. The last summand describes the number of occurrences of a pattern in a set of DNA sequences.

Computational experiments have shown that the developed algorithms provide significant acceleration and thus allow discovering the desired motifs with high sensitivity and specificity. Comparison of the efficiency of described method using the truncated suffix tree and the naïve search algorithm following table illustrates.

Table 8. The times for generalized motif discovery

Sequence count/ length	Patterns count/ length	Suffix tree (ms)	Naive (s)
100/100	100/4	451	1.203
100/100	1000/4	2111	61.191
100/100	10000/4	10523	560.040
100/100	100/8	677	1.080
100/100	1000/8	3507	54.149
100/100	10000/8	19936	538.764

Dimers.

Dimer is a motif consisting of two separate parts called 'boxes'. The distance between boxes is usually fixed for a given dimer but can be unknown in advance. Dimers are special cases of a more general notion of structured of structural motifs [38].

A structural model of the motif in general can be described by a pair (m, d) where *m* is a *p-tuple* of single models $(m_1,...,m_p)$ (the *p* boxes); *d* is a (*p*-1)-*tuple* of triplets $((d_{\min_1}, d_{\max_1}, \sigma_1),...,(d_{\min_{p-1}}, d_{\max_{p-1}}, \sigma_{p-1}))$, i.e. (*p*-1)-intervals of distance.

We developed and implemented an algorithm based on Gibbs sampling which discoveries dimers. The formal statement of the problem is as follows.

Given a set of sequences $\{s_j,\ j=1..k\}$, each consisting of *n* nucleotides, we seek for dimers such that the lengths of boxes (m_1 and m_2) are equal to L_1 and L_2 respectively and the distance between the boxes is within the given limits $d_{\min}$ and $d_{\max}$.

Let us rewrite the formula (2) for each box of the dimer (t = 1,2) and fixed line z, selected at random during the execution of the algorithm. Add an additional term, which will be responsible for assessing the distance between the boxes:

$$Q_i^t = \alpha * S(pattern_{L_t}(i), p) + \beta * f(x,\mu) + \gamma * \sum_{i=1}^{k} w_j(v) + g_t(i),$$

where $\lambda(v) = pattern_{L_t}(i)$, $i = 0..n - L_t + 1$, $\{a_j, j = 1..k\}$ and $\{b_j, j = 1..k\}$ are positions of boxes in each sequence at the current iteration of the algorithm.

One possible choice for the function responsible for the evaluation of the distance between the boxes is as follows:

$$g_1(i) = \begin{cases} \dfrac{1}{d_{\max} - d_{\min} + 1} & i \in [b_z - L_1 - d_{\max}; b_z - L_1 - d_{\min}] \\ 0 & i \notin [b_z - L_1 - d_{\max}; b_z - L_1 - d_{\min}] \end{cases},$$

$$g_2(i) = \begin{cases} \dfrac{1}{d_{\max} - d_{\min} + 1} & i \in [a_z + L_1 + d_{\min}; a_z + L_1 - d_{\max}] \\ 0 & i \notin [a_z + L_1 + d_{\min}; a_z + L_1 - d_{\max}] \end{cases},$$

Thus, the basic algorithm of Gibbs sampling was modified for structured motif (dimer) discovery in the set of DNA sequences by adding the second matrix PSSM Q_2, corresponding to a second box in the structure of the motif. This model can be generalized to discover structured motifs with arbitrary number of boxes.

The test results of the algorithm for the model $(d_{\min_1} = 2, d_{\max_1} = 5)$ are in the following table.

Table 9. The times for dimer motif discovery

Sequence count/ length	Length m_1 / length m_2	Count of iterations	Time(s)
100/100	4/4	1	0.015
100/100	4/4	1000	12.422
100/100	8/8	1	0.027
100/100	8/8	1000	24.268
100/100	16/16	1	0.042
100/100	16/16	1000	41.950
100/100	4/16	1000	36.376
100/100	8/16	1000	30.771
100/100	4/8	1000	29.379

C. Locating Origin of Replication

Search for origin of replication in genome of enterobacteria is a topical but still unsolved bioinformatics problem [32]. The problem is to find a region (the origin of replication) within the specified area of the genome. A putative origine of replication should contain other sub-sequences (DNA boxes) repeated a number of times. In addition, an origin of replication should satisfy the following conditions:

1. Replication origin is located in an AT-rich (GC-poor) area.
2. Length of replication origin of bacteria ranging from 100 to 1000 bp.
3. The amount of DNA boxes varies from 2 to 5 repeats on one helix (4 to 10 in two, respectively).
4. The length of DNA boxes varies from 6 to 16 bp.
5. Occurrences of a DNA-box can have up to two mismatches. For example: ATTGCA, AATGGA are two occurrences of the same DNA box with 2 mismatches.

We have developed a program for locating putative origins of replication. This program takes a sequence (genome) and expected length of DNA boxes, and returns the position of putative replication origin and the list of putative DNA boxes.

The method for locating origin of replication is based on locating AT-rich area and then searching for the minimum at the *GC-skew diagram* for this area. For a candidate region we then should determine putative DNA boxes. Since these boxes are not known in advance, this is the most time-consuming stage of the algorithm. A naïve algorithm uses brute force to check all possible substrings of length *l*. This algorithm has time complexity O($l \cdot N^2$). To speed up this operation, we implemented a procedure based on special data structure VP-tree [39]. This structure is a special case of metric trees and allows for fast processing neighbors in a metric space. The VP-tree based procedure has time complexity O($l \cdot N \cdot \log N$).

In Table 11 and Table 12 we have compared performances of our algorithm and the naïve exhaustive search. In Table 12 we present the results of the program in some genomes.

Table 10. Comparing the performance of algorithms relative to the length of box

Length of DNA box (symbols)	Exhaustive search (s)	VP-tree (s)
6	0.089	0.106
7	0.106	0.123
8	0.127	0.151
9	0.153	0.166
10	0.173	0.173
11	0.196	0.181
12	0.216	0.194
13	0.232	0.198
14	0.247	0.206
15	0.262	0.221
16	0.439	0.272

Table 11. Comparing the performance of algorithms relative to the length of the input sequence

Length of sequence *N* (symbols)	Exhaustive search (s)	VP-tree (s)
1000	0.153	0.166
2000	0.634	0.505
5000	3.936	2.835
10000	15.689	8.767
20000	62.502	29.283
50000	391.661	143.741

Table 12. The results of the program for the various datasets

Species	**Found DNA boxes**	**Known DNA boxes**
E.Coli	TTATCCACA	TTATCCACA
	TGTGGATAA	TGTGGATAA
Vibrio cholerae	CTTCATGAT	CTTGATCAT
	ATCATGAAG	ATGATCAAG
Salmonella enterica	GGATCCTGG	-
	CCAGGATCC	

We have compared our program to similar programs: Oriloc, GC-software, Z-curve, GraphDNA, OriFinder [33]. Almost all analogs only make assumptions based on various diagrams, including GC-skew. In Table 13 we compare the functionality of the programs.

Table 13. Comparing the programs' functionality

Program	DNA box	Using diagram
Oriloc	-	GC-skew TA-skew CDS-skew
GC-software	-	GC-skew TA-skew DNA-walk
Z-curve	-	(A+G)-(C+T)-skew (A+C)-(G+T)-skew (A+T)-(C+G)-skew
GraphDNA	-	Z-curve GC-skew AT-skew AC-skew
OriFinder*	+	(A+G)-(C+T)-skew (A+C)-(G+T)-skew GC-skew
Our program	+	TA-skew RY-skew MK-skew GC-skew

* To search for DNA-box user needs to make a guess about its form.

Source code of program for searching for an origin of replication can be obtained for free from the github: https://github.com/zeratul47/searching-an-origin-of-replication-

IV. Degenerate Primer Design

Degenerate primers are an important component of the polymerase chain reaction (PCR). It is widely used for various biotechnological applications. Such approach is useful for identifying bacteria in metagenomic studies, in particular for pathogen research [27]. Also, polymerase chain reaction allows solving various problems in molecular phylogeny (within defined taxa), for example, search for portions of the variable domains of proteins from related organisms. Moreover, PCR can be used for the discovery of new genes and even of new genomes from samples [45, 46]. Also, for a successful reaction and getting high quality product we must take into account a number of biological parameters [47, 48]. Wrong selection of these parameters can negatively affect the reaction and its results.

We have developed a program for searching pairs of degenerate primers. The importance and value of the parameters may vary depending on the task. So, the user is given the ability to change them and control the search process at all stages. For example, he/she can choose the algorithm for minimization of the degeneracy of primers under condition of covering all input sequences. Another option is searching for primers covering maximum number of input sequences under condition that the degeneracy does not exceed the specified value [28]. To improve performance of each algorithm we have introduced some heuristics caused by biological aspects or revealed as a result of testing. The output of the program is a few primer pairs. This allows the user to choose the most suitable pair of primers.

V. Plans for Future Development

Sequences of one species could be searched for SNPs (single nucleotide polymorphisms) or other type of mutations allowing creating mutational profile distinguishing species from another one. It could help in solving problems of biological classification, for instance, of pathogenic bacteria [30]. We are in progress with development of a program making a mutational profile for species or other taxonomic groups using a database with specific sequence marker defined computationally or by user. The other program application is the convenient graphical interface with sequence specific mutations pointed out against reference.

Suffix tree based procedures suffer from excessive memory requirements. Thus, reducing memory complexity while preserving the option of inexact pattern search is an urgent issue in our plans. There a several approaches addressing this problem, and ‘compressed suffix tree’ [31] looks promising enough. In addition, in the next version of the package we plan to implement switching between truncated tree and sparse suffix tree according to patterns size. It seems to improve suffix tree pattern search, both for exact and inexact cases.

References

[1] Thompson, C. C., Chimetto, L., Edwards, R. A., Swings, J., Stackebrandt, E., Thompson, F. L.: Microbial Genomic Taxonomy. BMC genomics, 14(1), 913 (2013)

[2] Pevzner, P.: Computational Molecular Biology: an algorithmic approach. Cambridge, Mass. MIT Press (2000).

[3] Gusfield, D.: Algorithms on Strings, Trees and Sequences. Cambrige University Press, 556 p. (1997)

[4] Goad, W. B., Kanehisa, M. I.: Pattern Recognition in Nucleic Acid Sequences I: A General Method for Finding Local Homologies and Symmetries. Nucl. Acids Res. 10, 247-263 (1982)

[5] Sellers, P. H.: Pattern Recognition in Genetic Sequences by Mismatch Density. Bull. Math.Biol. 46, 501-514 (1984)

[6] Waterman, M. S., Eggert, M.: A New Algorithm for Best Subsequence Alignments with Application to tRNA-rRNA Comparisons. J. Mol. Biol. 197, 723-728 (1987)

[7] Hall, J. D., Myers, E. W.: A Software Tool for Finding Locally Optimal Alignments in Protein and Nucleic Acid Sequence. CABIOS 4, 35-40 (1988)

[8] Basic Local Alignment Search Tool, http://blast.ncbi.nlm.nih.gov/

[9] ENCODE Project, http://www.genome.gov/10005107

[10] McGuire, A. M., Hughes, J. D., Church, G. M.: Conservation of DNA Regulatory Motifs and Discovery of New Motifs in Microbial Genomes. Genome Research, 10(6), 744-757.1 (2000)

[11] Okonechnikov, K.; Golosova, O.; Fursov, M.; the UGENE team: Unipro UGENE: A Unified Bioinformatics Toolkit. Bioinformatics, 28, 1166-1167 (2012)

[12] Molecular Evolutionary Genetics Analysis, http://www.megasoftware.net/

[13] Abu-Khalil, Z. M., Morylev, R. I., Steinberg, B. Y.: Parallel Global Alignment Algorithm with the Optimal Use of Memory. Modern Problems of Science and Education, 1 (2013) http://www.science-education.ru/en/107-8139 (in Russian)

[14] Lamport, L.: The Parallel Execution of DO Loops. Commun. ACM, 83–93 (1974)

[15] Myers, E.W., Miller, W.: Optimal Alignments in Linear Space. Computer Applications in the Biosciences, 4, 11-17 (1988)

[16] Rice, P., Longden. I., Bleasby A.: EMBOSS: The European Molecular Biology Open Software Suite. Trends in Genetics, 16, (6), 276-277 (2000).

[17] Cartwright, R.A.: Ngila: Global Pairwise Alignments with Logarithmic and Affine Gap Costs. Bioinformatics. 23(11),1427-1428 (2007)

[18] Adigeyev, M.G., Bout, A.A.: Efficiency Analysis of Applying Suffix Trees for Solving Some Bioinformatics Problems. Modern Problems of Science and Education. 6, http://www.science-education.ru/en/106-7418 (2012) (in Russian)

[19] Schulz, M.H., Bauer, S., Robinson, P.N.: The Generalised k-Truncated Suffix Tree for Time- and Space-Efficient Searches in Multiple DNA or Protein Sequences. In: Int. J. Bioinformatics Research and Applications, 81-95. Inderscience Publishers (2008)

[20] Bieganski, P., Ned1, J., Cadis, J.V., Retzel, E.E.: Generalized Suffix Trees for Biological Sequence Data: Applications and Implementation. In: System Sciences. Proc. of the Twenty-Seventh Hawaii International Conference, vol. 5 (1994).

[21] Ukkonen, E.: On-line Construction of Suffix Trees. Algorithmica, 14, 249–260 (1995)

[22] MUMmer, http://mummer.sourceforge.net/

[23] SuDS project, http://www.cs.helsinki.fi/group/suds/cst/

[24] Mansour, E., Allam, A., Skiadopoulos, S., Kalnis, P.: ERA: Efficient Serial and Parallel Suffix Tree Construction for Very Long Strings. PVLDB, 5(1), 49-60 (2011)

[25] Vens, C., Rosso, M.N., Danchin, E.G.: Identifying Discriminative Classification-Based Motifs in Biological Sequences. Bioinformatics, 27, 1231–1238 (2011)

[26] Lewis, S. M., Coté, A.G.: Palindromes and Genomic Stress Fractures: Bracing and Repairing the Damage. DNA repair 5.9, 1146-1160 (2006)

[27] Miller, R. R., Montoya, V., Gardy, J. L., Patrick, D. M., Tang, P.: Metagenomics for Pathogen Detection in Public Health. Genome medicine, 5(9), 81 (2013)

[28] Linhart, C., Shamir, R.: The Degenerate Primer Design Problem. Bioinformatics 18.suppl 1, S172-S181 (2002)

[29] Rose, T. M., Henikoff, J.G., Henikoff, S.: CODEHOP (COnsensus-DEgenerate hybrid oligonucleotide primer) PCR Primer Design. Nucleic Acids Research 31.13, 3763-3766 (2003)

[30] Nhung, P. H., Shah, M. M., Ohkusu, K., Noda, M., Hata, H., Sun, X. S.,Ezaki, T.: The dnaJ Gene as a Novel Phylogenetic Marker for Identification of Vibrio Species. Systematic and applied microbiology, 30(4), 309-315 (2007)

[31] Sadakane, K.. Compressed Suffix Trees with Full Functionality. Theo. Comp. Sys., 41(4), 589-607 (2007).

[32] Pevzner P. Bioinformatics Algorithms: an Active Learning Approach, USA: Active Learning Publishers, 2014. P.1-46.

[33] Sernova N. V., Gelfand M. S. Identification of replication origins in prokaryotic genomes // Briefings in Bioinformatics. vol 9. NO 5. 376-391, 2008.

[34] Sandve G.K. A survey of motif discovery methods in an integrated framework / G.K. Sandve, F. Drabløs // Biology Direct. – 2006. – URL http://www.biomedcentral.com/pubmed/16600018.

[35] Technical report on project 'Creating a bio-information technology for research of related organizational scenarios of non-coding DNA and protein coding DNA in the animal and human genomes' 14.740.11.0006 (III). Available: http://niib.sfedu.ru/themesnir/gosudarstvennyie-kontraktyi/14740110006

[36] Charles E.Lawrence, Stephen F.Altschul, Mark S.Boguski, Jun S.Liu, Andrew F.Neuwald, Jon C.Wootton Detecting Subtle Sequence Signals: a Gibbs Sampling Strategy for Multiple Alignment// Science. –New Series. – 1993. –Volume 5131. – Page(s):208-214.

[37] Adigeyev, M.G., Bout, A.A.: Efficiency Analysis of Applying Suffix Trees for Solving Some Bioinformatics Problems. Modern Problems of Science and Education. 6, http://www.science-education.ru/en/106-7418 (2012) (in Russian)

[38] Laurent Marsan, Marie – France Sagot Algorithms for Extracting Structured Motifs Using a Suffix Tree with an Application to Promoter and Regulatory Site Consensus Identification // Journal of computational biology. – 2000. – Volume 7. –Numbers 3/4. –Page(s):345-362.

[39] Yianilos P.N. Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces. // SODA '93 Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, P. 311-321, 1993.

[40] B. J. Steinberg, J. M. Abu-Khalil, M. G. Adigeyev, S. V. Avdyakov, A. A. Bout, A. V. Kermanov, E. A. Pshenichnyy, G. V. Ramanchauskayte, and N. Ponomareva, "SFedU Software Package for Nucleotide Sequence Analysis." *INASE and CSCC Conferences*, Zakynthos Island, Greece, July 16-20, 2015. Available : http://www.inase.org/library/2015/zakynthos/bypaper/MATH/MATH-12.pdf

[41] G. Abdel-Azim, M. Ben Othman, and Z. Abo-Eleneen, "Modified Progressive Strategy for Multiple Proteins Sequence Alignment", *International Journal of Computers*, vol. 5, pp. 270-280. 2011.

[42] L. Chaabane, and M. Abdelouahab, "A Hybrid Method Applied to Multiple Sequence Alignment Problem", *WSEAS Trans. on Computers*, vol. 14, pp. 465-473. 2015.

[43] Z. S. Zubi, and M. A. Emsaed "Using Sequence DNA Chips Data to Mining and Diagnosing Cancer Patients", *International Journal of Computers*, vol. 4, pp. 201-214. 2010.

[44] N. Kerdprasop, and K. Kerdprasop, "Recognizing DNA Splice Sites with the Frequent Pattern Mining Technique", *International Journal of Mathematical Models and Methods*, vol. 5, pp. 87-94. 2011.

[45] Z. Ma, and T. J. Michailides, "Approaches for eliminating PCR inhibitors and designing PCR primers for the detection of phytopathogenic fungi", *Crop Protection* Volume 26, Issue 2, February 2007, pp.145-161.

[46] M. Kotik "Novel genes retrieved from environmental DNA by polymerase chain reaction: Current genome-walking techniques for future metagenome applications" *Journal of Biotechnology* Volume 144, Issue 2, 26 October 2009, pp.75-82

[47] N. von Ahsen, C. T. Wittwer, and E. Schütz, "Oligonucleotide melting temperatures under pcr conditions: nearest-neighbor corrections for Mg2+, deoxynucleotide triphosphate, and

dimethyl sulfoxide concentrations with comparison to alternative empirical formulas”. *Clinical Chemistry 47*, vol.11: 1956-1961. 2001

[48] R. Wojciech “Selection of primers for polymerase chain reaction” *Molecular Biotechnology*, Volume 3, Issue 2, April 1995, pp. 129-134