SFedU Software Package for Nucleotide Sequence Analysis

Boris J. Steinberg, Jumana M. Abu-Khalil, Mikhail G. Adigeyev, Sergey V. Avdyakov, Andrey A. Bout, Anton V. Kermanov, Evgeny A. Pshenichnyy, Galina V. Ramanchauskayte and Natalia Ponomareva

***Abstract*—** We present a software package for nucleotide sequence analysis. Programs are being under development in the Southern Federal University (SFedU), Russia (available at mmcs.sfedu.ru/bio/). Currently, the package consists of modules for a number of bioinformatics problems including novel post-genomic challenges (like de-novo motif discovery). The most resource-consuming calculations are optimized by means of special data structures, parallel running, etc. For example, the package includes two variants of pairwise alignment algorithms (parallel block and parallel block with optimal memory usage) tailored for parallel run 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 paper describes the current state of the project, some performance evaluation results and plans and approaches for future improvements.

***Keywords*—** Bioinformatics, high performance computing, nucleotide sequence analysis.

This work was supported by project of the Ministry of Education and Science of Russia №6.703.2014/K.

Boris J. Steinberg is with the Southern Federal University, Dept of Math, Mechanics and Comp. Sci., Rostov-on-Don, Russia (e-mail: borsteinb@mail.ru).

Jumana M. Abu-Khalil is with the Southern Federal University, Dept of Math, Me-chanics and Comp. Sci., Rostov-on-Don, Russia (e-mail: juma-na.abukhalil@gmail.com).

Mikhail G. Adigeyev is with the Southern Federal University, Dept of Math, Mechan-ics and Comp. Sci., Rostov-on-Don, Russia, (e-mail: madi@math.sfedu.ru).

Sergey V. Avdyakov is with the Southern Federal University, Dept of Math, Me-chanics and Comp. Sci., Rostov-on-Don, Russia (e-mail: sergeyavdya-kov@gmail.com).

Andrey A. Bout is with the Southern Federal University, Dept of Math, Mechanics and Comp. Sci., Rostov-on-Don, Russia (e-mail: a-bout@yandex.ru).

Anton V. Kermanov is with the Southern Federal University, Dept of Math, Mechan-ics and Comp. Sci., Rostov-on-Don, Russia and Research Institute for Plague Con-trol, Rostov-on-Don, Russia (e-mail: av-kermanov@mail.ru).

Evgeny A. Pshenichnyy is with the Southern Federal University, Research Institute of Biology, Rostov-on-Don, Russia, (corresponding author to provide phone: 007-863-2975070; address: 344090 pr. Stachki 104/1 Rostov-on-Don, Russia; e-mail: pshenichniy.eugene@gmail.com).

Galina V. Ramanchauskayte is with the Southern Federal University, Dept of Math, Mechanics and Comp. Sci., Rostov-on-Don, Russia (e-mail: galinka@lastbit.com).

Natalia Ponomareva is with the Southern Federal University, Research Institute of Biology, Rostov-on-Don, Russia (e-mail: nsponomareva@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) leads to need of effective algorithms for nucleotide sequence data manipulations.

One of the most important bioinformatics problems is nucleotide sequence alignments. 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. Whereas global alignment is used in bioinformatics for evaluation of similarity between two sequences, 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.

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 joined together into regulatory loop.

Practical research in bioinformatics involves not only solving computationally difficult problems but also some additional tasks which are worth of automation. SFedU package provides several utilities for such tasks. Lots of bioinformatical software packages are available for biologists. However, most

of them are either primarily user interface wrappers integrating third party tools, or specialized heuristic procedures [8],[11],[12]. The main focus of our package is the enhanced analyses of DNA sequences. The package is oriented both for long strings (applicable for alignments, as an example) and for substring search (motifs and patterns).

II. Global Alignments

Our package includes several alignment procedures. The main features of these procedures are running alignments in parallel blocks and memory usage optimization. The basic data structure of Needleman-Wunsch and other 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 classic Needleman-Wunsch algorithm, as it does not save the similarity matrix as a whole. The size of each block is a flexible parameter, whose variation can lead to reduction of memory usage. Another our procedure for global alignment is based on the parallel global alignment algorithm with optimal memory usage. It performs alignment faster than Hirschberg's algorithm [3] and uses less memory than Needleman-Wunsch algorithm. This algorithm is based on [13] and the hyperplanes method [14]. 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. 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 parallel block algorithm with optimal memory usage is faster than its base algorithm [13] by 35%. 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.

Table 1. Calculation times for global alignment procedures

Seq. length (bp)	Time (sec)						
	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

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 has been based on Needleman-Wunsch one, so it requires a lot of memory. However, it is more memory effective than Ngila tool. The designed algorithms are more memory intensive than EMBOSS Stretcher.

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

ISBN: 978-1-61804-323-8

III Local Alignments

The parallel block procedure for local alignment is based on the Smith-Waterman dynamic programming 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, version 2.0u4 (Feb. 1996). 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. 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 classic 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 3. Calculation times for local alignment procedures

Seq. length (bp)	Time (sec)				
	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

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, and therefore could not perform alignment of long sequences. 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; in case of long sequences, Matcher requires less space, but this difference is small.

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

IV Suffix Tree-Based Procedures

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 (sec)	Our package (sec)
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 (sec)
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

Motif Discovery. 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.

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

Q_P	Q_N	MERCI (sec)	Our package (sec)
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

The motif discovery tool implemented within our package 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 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).

V Other modules

In addition to the above-mentioned procedures the package includes several modules that are useful for practical bioinformaticians. These modules allow: 1) automatically download sequences from online genomic databases; 2) calculate character and short words statictics; 3) search long sequences for palindromes.

The problem of loading large amounts of nucleotide sequences arises frequently when analyzing bioinformatical data. Most genomic databases can be accessed using public APIs. We have implemented a separate module for batch loading information from such databases. The module allows fetching nucleotide sequences from Ensembl and NCBI databases though the Ensembl API and the Entrez Programming Utilities (eUtils). Obtained sequences and additional meta-information can be stored within any SQL relational database. Database storage allows the data to be accessed simultaneously from several processes, including processes running on different computers.

Character and short words (oligos, oligomers) statistics are important sources for exploring patterns of DNA organization. The SfedU package includes a module for fast calculation of DNA statictics.

A 'palindrome' in bioinformatics is a sequence which is equal to its own reverse complement and flanks it. The palindromes are significant for genetic research because such sites formed in exact parts of nucleic acids could be the reason for stalling polymerase during RNA synthesis in a cell or DNA replication. It occurs due to hairpin-like structure of these elements interferring the enzyme to act appropriately [26]. The SFedU bioinformatics package includes a module for fast searching palindromes in given nucleotide sequences.

VI Plans for Future Development

Degenerate (universal) primers are widely used for PCR (polymerase chain reaction) of variable sequences. Such approach is valid to identify bacteria in metagenomic studies, in particular for pathogen research [27]. Theoretical basis and limitations for in silico primer construction have been discussed in [28]. We have implemented heuristic search algorithm. Other widely used algorithms [28, 29] will be included with parallel programming options and user-friendly interface with lots of parameters which could help in fine-tuning the expected result.

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/1 07-8139 (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).