Implementation of the Vectorized Position-Specific Iterated Smith-Waterman Algorithm with Heuristic Filtering Algorithm on Cray Architecture

Witold Rudnicki, Rafal Maszkowski,
Lukasz Bolikowski, Maciej Cytowski, Maciej Dobrzynski, Maria Fronczak,
Interdisciplinary Center of Modeling, Warsaw University

May 2005

ABSTRACT: *Smith-Waterman is the algorithm for finding local optimal alignments of two amino acid sequences, but is too slow for use in regular large database scanning. We have implemented SW algorithm on Cray X1 architecture, exploiting vectorization and parallelization. In addition we have proposed heuristic database filtering algorithm, which is particularly well suited for the Cray architecture, exploiting possibilities given by the BMM unit.*

KEYWORDS: Cray X1, BMM chip, bioinformatics, Smith-Waterman, sequence alignment

1 Introduction

The aim of this paper is to briefly summarize some technical aspects of implementing the Position-Specific Iterated Smith-Waterman algorithm for local sequence alignment on Cray architecture.

The project was initiated at the Interdisciplinary Center of Modeling in 2002. At first, our goal was to test the capabilities of the BMM unit[1] by implementing the original Smith-Waterman algoritm on Cray SV1. At that time there was no *BioLib*[2] package, nevertheless an equivalent tool was necessary. It turned out that the Smith-Waterman algoritm itself cannot substantially benefit from the BMM, but a better understanding of the unit allowed us to design certain filters that would limit the number of calls to the S-W routine.

Finally, as Cray X1 appeared at ICM, it was feasible to implement the iterated version of S-W (PSI S-W), in a fashion similar to the PSI BLAST[3]. It was tempting to check whether such approach could be competitive to PSI BLAST in terms of speed, and whether the alignments obtained by PSI S-W could be substantially better than ones obtained by PSI BLAST.

2 Theory

2.1 Sequence alignments

The purpose of Smith-Waterman algorithm is to find a *local alignment* of two *sequences* that is maximal according to the chosen *scoring system*.

The sequences are simply strings of letters: there are 20 letters to choose from in case of amino acids and 4 letters in case of nucleotides. Let us explain what a local alignment by an example. Take two amino acid sequences: `MDRKVTPGSTCAVFGLGGVGLSAIMGFIL` and `MKLNPGSSGHGGMGATMTSAVMGDRNN`. The optimal local alignment for the two is:

```
Query: 4  KVTPGSTCAVFGLGGVG---LSAIMG 26
          K+ PGS+    G GG+G    SA+MG
Sbjct: 2  KLNPGSS----GHGGMGATMTSAVMG 23
```

[1] Bit Matrix Multiply (BMM) is a hardware functional unit available on Cray SV1 and Cray X1 architectures that performs fast multiplication of two 64×64 bit matrices (the adds and multiplies are *modulo* 2)

[2] Cray Bioinformatics Library (BioLib) is a set of routines for nucleotide and amino acid sequence manipulation

[3] Position-Specific Iterated Basic Local Alignment Search Tool (PSI BLAST) is the most popular tool for local sequence alignment. It is a very fast heuristic, while S-W is an exact algorithm

The top row is a fragment of the first sequence, the bottom row is a fragment of the second one. Each letter of a sequence is assigned to a letter in the other sequence, or to a gap between two consecutive letters in the other sequence.

A scoring system for all alignments shall be defined to select the optimal alignment from the set of all possibilities. The system used by the Smith-Waterman algorithm is based on the probability of various mutations, such as replacements, insertions and deletions, in the protein sequences.

The score of an alignment is computed as follows: each aligned pair of letters is given an integer value, and the values are summed up. Then each gap of length l is given a penalty $G + lE$, where G and E are constants named *gap opening penalty* and *gap extension penalty*. Finally, all the penalties are substracted from the sum and the result is the alignment score.

The scoring system is, thus, respresented by a symmetric table T of scores for each pair of letters, and a pair (G, E) of penalties. The table is 20×20 for amino acids and 4×4 for nucleotides. A fragment of BLOSUM62, a popular amino acid scoring table, is shown below:

	A	R	N	D	C	Q	E
A	4	-1	-2	-2	0	-1	-1
R	-1	5	0	-2	-3	1	0
N	-2	0	6	1	-3	0	0
D	-2	-2	1	6	-3	0	2
C	0	-3	-3	-3	9	-3	-4
Q	-1	1	0	0	-3	5	2
E	-1	0	0	2	-4	2	5

Note that the diagonal elements are always positive and greater than any other one in the row. This is natural, since alignment of a sequence A with itself (a perfect alignment) should have score greater than any alignment of A with a different sequence.

2.2 Smith-Waterman algorithm

The Smith-Waterman is a simple dynamic programming algorithm. Let A and B be the two sequences being aligned. It starts by allocating a table S of dimensions equal to the lengths of A and B. Then it fills the table from upper-left to bottom-right (by rows, columns or antidiagonals – any of these orders is good) with value:

$$S_{i,j} = \max \begin{pmatrix} S_{i-1,j-1} + T(A_i, B_j) \\ S_{i,j-1} - Penalty \\ S_{i-1,j} - Penalty \\ 0 \end{pmatrix} \quad (1)$$

Penalty here stands either for G, when a gap is opened, or for E, when it is extended. The intuitive meaning of a value $S_{i,j}$ is the score of the best local alignment ending at A_i, B_j.

At each step, when a maximum value is found, it is recorded from which direction the result was obtained: diagonal, top, left, or none. The table S together with information about the directions provide enough data to find the best local alignment.

To find the best alignment, one has to do as follows. First, find the maximum value within table S. This is the end of the highest scoring local alignment. Then, backtrace from the cell in the recorded direction. A diagonal move from $S_{i-1,j-1}$ to $S_{i,j}$ represents an aligned letter pair A_i, B_j; a horizontal move from $S_{i-1,j}$ to $S_{i,j}$ represents A_i aligned with a gap between B_{j-1} and B_j; a vertical move is analogous to the horizontal one; while max $= 0$ at $S_{i,j}$ means the optimal local alignment starts at $S_{i+1,j+1}$.

2.3 Database searches and PSI S-W

A typical use of a local alignment algorithm is such: one sequence, *a query*, is aligned against *a database* of sequences, one by one. The most significant alignments, and their corresponding sequences are presented to the user.

Position Specific Iterated Smith Waterman (PSI-SW) is a modification of the original SW algorithm analogous to the PSI-BLAST modification of the BLAST algorithm. In this algorithm the single scoring table is replaced by the position specific scoring matrix (*PSSM* or *profile*). A set of unique 20 scores for each amino acid in the query sequence is used. These scores are obtained in the iterative self-consistent procedure. In the first step the scores from the similarity matrix, such as BLOSUM62 are used to find the set of homologous sequences. Then all sequences are aligned with the query and frequencies of the aminoacid appearance at each position is computed and translated into the scores in the position specific scoring matrix. Then a new search for the homologous sequences is performed using new PSSM, presumably leading to finding more homologous sequences than in the search with the original

socring matrix. This procedure is repeated until no more new sequences are found.

3 Implementation

3.1 Core S-W

The core Smith-Waterman algorithm is quite straightforward to implement. One crucial observation is that updating the S table should be done by antidiagonals, since all operations on a antidiagonal are then independent of each other, which enables vectorization.

3.2 Filtering

For the filtering phase, the S table is divided into several overlapping 64×64 subtables (overlapping is required to eliminate some edge effects). Each subtable is initially filled with 0s and 1s: a cell $S_{i,j}$ is set to 1 iff $T(A_i, B_j) > 0$. Such table can be stored in a BMM register and a series of vector bit operations determines whether the region is likely to hold a high-scoring alignment. When such a 64×64 subtable is found, then the filter calls the real Smith-Waterman algorithm for this pair of sequences.

3.2.1 Filling the 64×64 matrices

In order to create the 64×64 tables, the bit matrix multiply operation is used. Each letter of the alphabet is represented as a 64-bit vector (the size of a word): $A = (1, 0, 0, 0, \ldots), B = (0, 1, 0, 0, \ldots)$, etc. The query is divided into 64-letter long segments, each represented by a bit matrix Q. The matrix Q contains the rows of the scoring table corresponding to the letters of the query. It has a very useful property: for any 64-letter long segment of a database sequence D (which is a 64×64 matrix of vector representations of letters), Q bit-multipied by D gives exactly the subtable S for this query and database sequence.

3.2.2 Identifying long runs rich in 1s

Two main observations are used as a base for the filter. First, the high-scoring regions usually contain long diagonal segments containing *mostly* 1s. The job of the vector bit operations is to locate the short series of 0s that are surrounded by long series of 1s and convert the 0s to 1s. Then, it is checked whether there is a run of 1s that is longer that a given threshold. If this is the case, then the 64×64 subtable has passed this test.

The example below shows how to do such a test. For a diagonal `00111001101110001000` a series of right shifts and `OR`s is done:

```
00111001101110001000
 00111001101110001000
  00111001101110001000
----------------------
00111111111111110111000
```

Next, a series of left shifts and `AND`s is done:

```
00111111111111110111000
01111111111111110111000
11111111111111110111000
----------------------
00111111111111110001000
```

This set of operations converted all the runs of at most two 0s into 1s. The last step is to check whether there is a sufficiently long segment of 1s.

3.2.3 Identifying islands of 1s

Another pattern exploited by the filter is the fact that the high-scoring regions also contain islands of 1s uncut by any 0 (these segments are much shorter than segments from the previous case). Here, the job of vector bit operations is simply to check if there is a sufficiently long run of 1s.

Again, a series of shifts and `AND`s can identify such regions. For a diagonal `00111001101110001000` the result is:

```
00111001101110001000
 00111001101110001000
  00111001101110001000
----------------------
00001000000001000000000
```

A non-zero result means that there were islands of 1s on the diagonal.

3.2.4 Filter parameters

The two tests can be combined either by the `OR` or by the `AND` rule. In fact, most parameters and thresholds of the filter are command-line options. Some research was done to identify the sets of filter parameters that are particularly good for database searches. Two default settings were defined: a conservative and an aggressive one.

3.3 PSI S-W

The implementation of PSI S-W algorithm was also straightforward. It required certain changes to the original S-W code, since the iterated version introduces Position-Specific Scoring Matrices (PSSM) instead of the traditional scoring tables.

3.4 Statistics, user interface

The tool output is similar to the reports generated by BLAST for at least two important reasons: the reports are easy to read, and most people involved in bioinformatics are accustomed to them.

An important step in selecting the alignments that are to be reported to the user is to assess their statistical significance. The tool assesses the significance using Karlin-Altschul-Dembo theory, the same that is used by BLAST ([4], [5]).

4 Performance

4.1 Quality of results

A number of queries were run against the SWISS-PROT database using BLAST and various versions of our tool. Settings of the test was similar to the one presented in [2].

Table 1 shows the number of sequences in the database that attain a fixed level of significance, for different queries and algorithms.

4.2 Speed of execution

A query against a database of 10,000 random sequences took, depending on the algorithm version:

Algorithm	Time for 1 SSP
PSI S-W	80 sec.
PSI S-W + cons. filter	31 sec.
PSI S-W + aggr. filter	7 sec.

5 Conclusions

5.1 Summary of the tool

We managed to implement the Smith-Waterman algorithm and its performance is comparable to the one implemented independently in Cray BioLib.

Moreover, we have designed an optional filter that calls S-W only for the sequences that have a good chance of attaining a high alignment score. The filter was designed to efficiently use the BMM unit. It works well for good, significant alignments, but will not speed up the search in less obvious cases.

Finally, Position-Specific Iterated version of the S-W algorithm was implemented.

5.2 Comparison with BLAST

It seems natural to compare the capabilities of our tool to the BLAST and PSI BLAST.

An obvious advantage of Smith-Waterman is that it is an exact algorithm, while BLAST in only a heuristic. On the other hand, even a highly vectorized implementation of S-W is still, by an order of magnitude, slower than BLAST. Therefore, the choice between S-W and BLAST depends on whether the user needs to have the exact results. If so, then Smith-Waterman is required, and Cray platform is an excellent choice due to vectorization.

With filtering, an optional feature, the tool becomes a heuristic too. In some cases it becomes competitive to BLAST in terms of speed, but otherwise BLAST is much faster while giving similar results. It is worth to notice though, that BLAST vectorizes poorly, so if we restrict ourselves to the Cray X1 platform, our tool remains competitive.

Finally, the comparison between PSI BLAST and PSI S-W is similar to the discussion on BLAST and S-W: the choice between the two depends on whether the user needs exact results.

About the authors

The project was led by Dr. Witold Rudnicki, Deputy Director, HPC Division, ICM. His scientific research focuses on bioinformatics and HPC. He can be reached at: ICM Warsaw University, Pawinskiego 5A blok D, 02-106 Warsaw, Poland, e-mail: rudnicki@icm.edu.pl. Lukasz Bolikowski, Maciej Cytowski, Maria Fronczak and Rafal Maszkowski are Software Developers at ICM. Maciej Dobrzynski was a student of Dr. Rudnicki in an early stage of the project.

A large portion of work was done by Witold Rudnicki and Rafal Maszkowski, who initiated the project. Witold was responsible for the theoretical design, while Rafal implemented the core S-W and done some extensive BMM hacking.

References

[1] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman D.J. (1990) Basic local alignment search tool, *J. Mol. Biol.,* **215**, 403-410.

[2] Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J. (1997) Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* **25**, 3389-402.

[3] Henikoff, S. and Henikoff, J.G. (1992) Amino acid substitution matrices from protein blocks, *Proc. Natl. Acad. Sci. USA,* **89**, 10915-10919.

[4] Karlin, S., Altschul, S.F. (1990) Method for assessing the statistical significance of molecular sequence features by using general scoring schemes, *Proceedings of the National Academy of Science, USA* **87**, 2264-2268.

[5] Karlin, S., and Dembo, A. (1992) Limit distributions of maximal segmental score among markov-dependent partial sums, *Advances in Applied Probability* **24**, 113-140.

[6] Pearson, W.R. and Lipman, D.J. (1988) Improved tools for biological sequence comparison, *P. Natl Acad. Sci. USA,* **85**, 2444-2448.

[7] Wootton, J. C. and S. Federhen (1993). Statistics of local complexity in amino acid sequences and sequence databases. *Computers in Chemistry* **17**, 149-163.

[8] Wootton, J. C. and S. Federhen (1996). Analysis of compositionally biased regions in sequence databases. *Methods in Enzymology* **266**, 554-571.

Protein family	Query	BLAST	PSI BLAST	S-W	PSI S-W	S-W w/filter	PSI S-W w/filter
Serine protease inhibitor	P01008	155	161	157	161	121	121
Ras	P01111	500	1001	568	1407	192	200
Globin	P02232	57	722	147	786	48	48
Hemagglutinin	P03435	141	145	142	170	108	108
Interferon α	P05013	76	76	76	76	70	70
Histocompatibility antigen	P10318	289	454	296	392	146	146
Cytochrome P450	P10635	502	717	662	716	312	312
Glutathione transferase	P14942	119	238	127	211	73	84
Alcohol dehydrogenase	P07327	221	303	232	287	128	129

Table 1: Number of SWISS-PROT sequences yielding statistically significant alignements for various queries and algorithms. All the tests used BLOSUM62 matrix with $(G, E) = (11, 1)$ and E-value of 0.01.