# Distributed Protein Sequence Alignment

**J. Michael Meehan**
**Computers Science Department**
**Western Washington University**
**Bellingham, WA 98225**
**meehan@wwu.edu**

**Heidi Young**
**Department of Computer Science**
**Western Washington University**
**Bellingham, WA 98225**
**heidi.young1@comcast.ne**

**James Hearne**
**Computers Science Department**
**Western Washington University**
**Bellingham, WA 98225**
**hearne@wwu.edu**

**Philip A. Nelson**
**Computers Science Department**
**Western Washington University**
**Bellingham, WA 98225**
**meehan@wwu.edu**

## ABSTRACT

Given the explosive growth of biological sequence databases and the computational complexity of aligning large sequences over extremely large databases most researchers have opted for utilizing the BLAST algorithm. While BLAST is completely appropriate for some purposes, the more rigorous and more computationally expensive Smith-Waterman algorithm is preferred for certain purposes. This work presents an implementation of the mathematically optimal Smith-Waterman protein sequence alignment algorithm using a collection of distributed computers. We also present the use of the Unicon programming language as an alternative for writing biological search algorithms and other applications in bioinformatics rather than the most commonly found C or Perl approaches. The system has fault tolerant capabilities and can dynamically add and remove nodes to deal with worker node failure. The system currently operates on up to 87 P4 3.0 GHz machines. The system is called UDPS for **U**nicon **D**istributed **P**rotein **S**earcher

## 1 INTRODUCTION

This work concerns the development of UDPS a system to perform distributed protein database searches. UDPS is written in a combination of C and Unicon.[1] [1] The pieces of the distributed system communicate via standard sockets. The non-redundant Protein Data Bank (PDB[2]) is divided into roughly equal size slices and acquired by the various nodes of a cluster of computers. A web form based front end provides an interface for users to submit search requests. The system performs the search and sends the result back as a web page or through email.

UDPS uses the Smith-Waterman algorithm [2] for aligning protein sequences. The Smith-Waterman algorithm served as a base for developing the BLAST algorithm. Since DNA, protein, and other biological sequences are quite large and the number of entries in the respective databases are large and growing rapidly, alignment techniques need to be highly efficient . It is common practice for researchers to utilize the BLAST local alignment search tool provided by the National Center for Biotechnology Information [3]. One of the drawbacks of the BLAST tool is that it can miss potentially valuable alignments. This "leakage" in BLAST is the result of the fact that BLAST concedes mathematical rigor for speed. However, with the rapid increase computing power and resources that exist today, the mathematically rigorous Smith-Waterman algorithm is a viable alternative. The Smith-Waterman algorithm is mathematically rigorous, meaning that it will guarantee an optimal alignment. BLAST cannot promise this result.

## 2 BACKGROUND

Finding the optimal alignment between two sequences using Smith-Waterman employs dynamic programming.. Dynamic programming is a technique that is commonly utilized in optimization problems. Dynamic programming algorithms capitalize on the property that in order to find a *global* optimum the subsequences found in the solution must themselves be *locally* optimal. Thus, dynamic programming is the process of extending a locally optimal solution incrementally to produce a globally optimal solution. The use of local and global in this context is not the same as their usage below.

### *2.1 Global Alignment vs Local Alignment*

Global alignment in our context refers to the techniques in which two sequences of DNA, proteins, or some other biological structure are compared for similarity across the entire lengths of the sequences. One end-to-end alignment is performed by creating a cumulative alignment score. The manner in which the alignment scores are computed varies depending on the type of

1 One piece of the system was written in C because at the time we began the project there was a feature of the C select statement that was not available to us given the way select was implemented in Unicon. While we were developing the system this feature was added to Unicon. We may go back at some point and rewrite the one C component in Unicon.

2 See http://www.rcsb.org/pdb/

biological sequence being aligned. In the case of protein alignments, the protein is represented by a sequence of letters. One for each naturally occurring amino acid. The positions are then compared using a similarity matrix.

Similarity matrices are used rather than a simple match / doesn't match due to the nature of amino acid sequences in proteins. It is well know that some amino acids can substitute for certain others in closely related proteins. For example serine is often observed to take the place of threonine in related proteins. These permissible substitutions presumably result from various similarities between the pair of amino acid such are whether they are both hydrophobic or hydrophilic or whether they are both polar.

Various substitution matrices have been developed. Two sets of similarity matrices are in common use. The first set is the Point Accepted Mutation (PAM) matrices. [4] The PAM matrices utilize a log-odds system and attempt to remove background randomness from the scores. The score in a PAM matrix is proportional to the natural log of the ratio of the target frequency to the background frequency. The other set of substitution matrices in common use are the BLOSUM matrices. [5] We will not discuss further how these substitution matrices are derived.

The last detail we need concerning the scoring of the alignments deals with gaps. Gaps are allowed to be inserted into the sequences to account for evolutionary insertions and deletions. The standard approach is to charge a gap opening penalty (G) and then a gap extension penalty (E) for each subsequent gap after the opening gap. There is little to no theoretical foundation we can draw upon to determine what the appropriate values for G and E should be. These number are usually parameters the user of a sequence alignment program can specify. The usual approach is to make G around an order of magnitude larger than E. The exact values selected will depend on which scoring matrix is being used.

The Needleman-Wunsch algorithm [6] is a global sequence alignment method from which the Smith-Waterman algorithm was derived. In this method, two sequences are set up along the edges of a two-dimensional matrix. Then, each cell of the matrix is filled in with scoring values. When the entire matrix has been filled, a trace back is done on the entire matrix to determine the overall similarity match of the two sequences. The algorithm is a global alignment in the the score obtained is accumulated from end to end. If the alignment leading up to a given point has achieved a negative score it is propagated.

Local alignment of two sequences computes the best subsequence alignment. The local alignment technique is used widely in the biological/medical community because proteins can be functionally similar due to localized similarity. Thus, an overall alignment is not of primary interest. In general proteins exhibit a modular structure and it is often the identification of this substructure that is of interest. For example, if a compound is know to bond to a certain region of a known protein then if you can find a similar region in another protein, it is like that the compound may bond it as well.

The Smith-Waterman algorithm is a local sequence alignment that is implemented in a manner similar to the Needleman-Wunsch algorithm. Though it employs only simple modifications to the Needleman-Wunsch algorithm, it does have significantly different results. The Smith-Waterman method finds the best matching regions within a pair of sequences, instead of aligning them along their entire lengths. This is achieved by making small changes to the values in the matrix cells, and following a different traceback method. Smith-Waterman does not allow a score to go negative. If the score reaches zero we are effectively beginning anew to find a local maximum.

### *2.2 Smith-Waterman Algorithm*

Smith-Waterman is guaranteed to find the optimal local alignment alignment within two sequences. The following example presents a simple illustration of the Smith-Waterman algorithm using English words instead of biological sequences. The algorithm is a general alignment strategy and can be utilized for various applications. The algorithm does not care whether the letters stand for amino acids or are just letters. The part that makes it specific to protein searching is the nature of the substitution matrix and its scores.

In the initialization phase, the first row and column of the matrix must be filled with zeros. In this example the two sequences to be aligned, PREEXIST and EXISTENCE are placed along the edges of the matrix as shown.

In the next phase of the algorithm, the fill phase, also called the induction phase, each one of the cells of the matrix is filled with scores and pointers. Each score is computed according to the following equation:

$$SW_{i,j} = \max\left\{ SW_{i-1,j-1} + s\left(a_i, b_j\right); SW_{i-k,j} + g_j; SW_{i,j-k} + g_i; 0 \right\}$$

**equation 1.**

| | | P | R | E | E | X | I | S | T |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| E | 0 | | | | | | | | |
| X | 0 | | | | | | | | |
| I | 0 | | | | | | | | |
| S | 0 | | | | | | | | |
| T | 0 | | | | | | | | |
| E | 0 | | | | | | | | |
| N | 0 | | | | | | | | |
| C | 0 | | | | | | | | |
| E | 0 | | | | | | | | |

*Table 1 – Smith-Waterman Alignment Matrix (initialization phase*

where $SW_{i,j}$ is the Smith-Waterman score for the partial alignment of the sequences ending at position ***i*** of sequence ***a*** and position ***j*** of sequence ***b***. The first term corresponds to extending the alignment by one more position, and the scoring function *s()* returns the score for matching the $a_i$ letter from sequence ***a*** and $b_j$ letter from sequence ***b***. The second term describes extending the alignment by including position ***j*** from sequence ***b*** and inserting a gap of ***k*** positions in length. The third term is for inserting a gap into sequence ***b***. Finally, the fourth term, **0**, is used to end an alignment and is the main difference that distinguishes Smith-Waterman from Needleman-Wunsch. The zero term allows an alignment to end if the scoring of the previous values was not significant enough.

For this example, a simple scoring scheme of

```
Match    = +1
mismatch = -1
gap      = -1
```

is used. However, when aligning protein sequences typically either the BLOSUM or PAM matrices discussed previously is used. For more on substitution matrices see [7].

In the trace-back phase of Needleman-Wunsch, the trace began at the bottom right corner of the matrix and the alignment followed along the entire sequences until it reached the top left corner. In the Smith-Waterman algorithm, the trace-back begins with the *highest scoring cell* in the matrix and the trace-back occurs by following the path through the maximum scores back until a zero value is reached. So for each cell, not only must a scoring value be held, but a directional value must be kept as well indicating how we got from one cell to the next in the optimal path. When the value for the cell is chosen from the four values discussed above a directional arrow can be chosen. The first term corresponds to a diagonal arrow back and to the left, the second term corresponds to a vertical arrow, the third term corresponds to a horizontal arrow, and the zero term is directionless since it is the end of an alignment.

Continuing with the example and using the simple scoring scheme shown above, Tables 2 and 3 show an alignment in progress and completed. Note that when the sequences become aligned their similarity scores align along a diagonal, if there was a gap, the values line up vertically or horizontally.

Since 5 is the highest score in the matrix, that is where the trace-back begins. The trace-back continues toward each of the shaded values. The optimal solution was indeed found; the subsequences EXIST are returned as a result.

## 3 SYSTEM DESIGN

The UDPS has been installed on a cluster of 87 3.0 GHz Pentium 4 machines all running the NetBSD 2.0 operating system,. [8] There is a controller node that runs the Apache web server, a cgi script written in Unicon, a control program written in Unicon, and a communications link program written in C. The control program is executed to initiate the system. It performs all system startup/shutdown tasks. The control program is the system administrators window into the status of the system.

The control program also handles requests from the web front end via the cgi program. Users enter a protein sequences and various parameters dealing with which scoring matrix to use etc. via a web form. The control program starts the communications link at system startup. It then reads a configuration file to determine the nodes in the system it should attempt to use. The control program kicks off a worker program written in Unicon at

each of the worker nodes. The workers all report in to the communications link program by establishing a socket connection. It is the responsibility of the communications link program to determine how many nodes in the system are alive at any point. The control program keeps a queue of alignment search requests which have been delivered via the cgi program. When the control program gives a command to the communications link program to pass work out to the workers included in the command is an indication of how many nodes are working and thus which version of the database is to be used to perform this work.

The protein data bank is divided into n pieces where n is the number of worker nodes currently alive in the system. Files containing the *n*-way split of the database are created by the control program and retrieved from the control programs file space by the workers to initiate their search tables. Each worker node reads the entire slice of the database assigned to it into a table *in memory*. There is sufficient RAM on each of the nodes to accomplish this.

The communications link receives messages from the worker nodes providing the results of the sequence alignment search. If a worker node were to die after work has been passed out for it to do, the communications link program detects this and notifies the control program. The control program then arranges to assign this work to one of the remaining nodes when it finishes its assignment. The control program at this point checks to see whether an *n-1* way split of the database has been prepared. If not it creates one. Subsequent assignments will use the degraded *n-1* capability until the failed node comes back online. When this occurs the communication link detects it and notifies the control program. The control program then reverts to assigning work against the *n*-way split of the database.

An important feature of the system is how each piece of the database is used when searching. In order to cut down on i/o time dramatically, before requests are made the database on each node is read directly into a data structure in memory to avoid having to read from the file over and over during the alignment. If the control program later requests that the worker use a different split of the database than the one currently held in RAM, the worker deletes the old table and reloads from the new file.

| | | *P* | *R* | *E* | *E* | *X* | *I* | *S* | *T* |
|---|---|---|---|---|---|---|---|---|---|
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *E* | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| *X* | 0 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 |
| *I* | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 2 | 1 |
| *S* | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 4 | 3 |
| *T* | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 | 5 |
| *E* | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 2 | 4 |
| *N* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 3 |
| *C* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 2 |
| *E* | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
| | | | | | E | X | I | S | T |
| | | | | | E | X | I | S | T |

**Table 2 – Smith-Waterman Alignment Matrix (final results)**

Users fill in the form provided through the web interface [5] to enter their protein sequences and request to have them aligned with either another protein sequence, or with the the "nr" protein database. The nr database is a combination of the GenBank, EMB, DDB, and PDB sequences. Though "nr" used to stand for non-redundant, this is no longer true for this collaborative database.

## 4 FUTURE WORK

It was noted earlier that the Smith-Waterman algorithm is mathematically rigorous, in that it will return an optimal result for aligning sequences. However, this does not mean that it returns **all** of the optimal results. It is mathematically possible for there to be two or more subsequences that have the same optimal similarity score within the sequences or possibly partially overlapping. In Smith-Waterman a decision is made upon encountering equal score as to which path to retain. This decision is left entirely up to the person implementing the algorithm. If there is a match in scores and a maximum value must be chosen (as in eq. (1)), only one value is retained. Thus, there could be many alternative paths through the matrix that would produce the same score. We intend to modify Smith-Waterman to retain *all* the alternate path information. The user will then have the an

option requesting all alignments which produced the same optimal score. Although the modification of the code required can be accomplished rather easily in ***Unicon*** it opens up the possibility that the amount of memory required to execute the modified algorithm could in theory grow exponentially. We do not anticipate that this will be the case in actuality. This will be an interesting value to track in the program in testing mode.

Altschul and Ericson [9] as well as Waterman and Eggert [10] have noted that searching for local alignments may produce several significant results. It is therefore a mistake to concentrate solely on the optimal alignment. Enhancements to Smith-Waterman to produce the *n* best non-overlapping alignments and as well as other refinements were incorporated into the SIM algorithm presented by Huang. [11] The LALIGN program, which is part of the FASTA package, implements the SIM algorithm. [12] It is our intention to modify our current Smith-Waterman search to offer these enhancements as options on the search form.

The next addition planed for the system is to strengthen the robustness of the system by removing the current *single point failure* at the controller node. The workers will be augmented to detect a failure in the controller node. They will then initiate an election algorithm[3] to select a new coordinator. The worker process at the selected node will fork a new temporary controller task. This task will then control the system until it fails or the original controller node is brought back online. The original controller will notice it did not exit cleanly and will initiate a "rigged" election to once again become the controller.

The web server running the front end for the system to receive work requests resides on the same node as the controller program. Thus, in addition to regenerating a controller the newly elected control node will need to start the web server and use dynamic DNS update to change the IP pointed to by the external URL. These additions should provide a very robust system capable of surviving all but total node failure.

Currently, revised versions of the nr PDB must be downloaded manually. A desirable addition would be to have the control program be used to set the frequency to check for updated versions of the database and to download and split them during idle times. The control program could also be expanded to retain old copies of the database for n days after they have been replaced and then automatically purge them from the system.

**5 REFERENCES**


[1] http://unicon.sourceforge.net/

[2] Smith, T.F. and Waterman, M.S. *Identification of Common Molecular Subsequences.* (1981). J. Mol. Biology. 147, 195-197

[3] *"NCBI BLAST."* National Center for Biotechnology Information. <http://www.ncbi.nlm.nih.gov/BLAST/>

[4] Dayhoff, M. O., Schwartz, R. M. and Orcutt, B. C. (1978) A model of evolutionary change in proteins. In Atlas of Protein Sequence and Structure, M.O. Dayhoff, ed. (Washinton D.C. : National Biomedical Research Foundation), pp. 345-352.

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

[6] Needleman, S.B. and Wunsch, C.D. *A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins.* (1970). J. Mol. Biology. 48. 443-453

[7] *"Substitution Matrices."* National Center for Biotechnology Information. <http://www.ncbi.nlm.nih.gov/Education/BLASTinfo/Scoring2.html>

[8] http://www.netbsd.org/

[9] Altschul, S. F. and Erickson, B. W. (1986). Locally optimal subalignments using nonlinear similarity functions. Bull. Math. Biol. 48, 633-660.

[10] Waterman, M. S., and Eggert M. (1987). A new alogorithm for best subsequence alignments with application to tRNA-rRNA comparisons. J. Mol. Biol. 197, 723-728.

[11] Huang, X. Hardison, R. C. , and Miller, W. (1990). A space-efficient algorithm for local similarities. Comput. Appli. Biosci. 6, 373-381.

[12] Pearson, W. R. (1996). Effective protein sequence comparison. Methods Enzymol. 266, 227-258.


3 We are currently adding the bully algorithm to the system.