Educational Module on Genomic Sequence Alignment Using HPC

Angela B. Shiflet
George W. Shiflet
Wofford College
Department of Computer Science
Department of Biology
Spartanburg, S. C. 29303 USA
+01 (864) 909-5396
shifletab@wofford.edu
shifletgw@wofford.edu

Daniel S. Couch
Wofford College Student
Blue Waters Intern
Spartanburg, S. C. 29303 USA
+01 (864) 597-4000
couchds@email.wofford.edu

Pietro Hiram Guzzi
Mario Cannataro
University "Magna Græcia" of Catanzaro
Department of Medical and Surgical Sciences
Catanzaro, Italy
+39 0961-369 4100
hguzzi@unicz.it
cannataro@unicz.it

ABSTRACT

"Aligning Sequences–Sequentially and Concurrently," an educational computational science module by the authors and available online, develops a sequential algorithm to determine the highest similarity score and the alignments that yield this score for two DNA sequences. Moreover, the module considers several approaches to parallelization and speedup. Besides a serial implementation in C, a parallel program in C/MPI is available. This paper describes the module and details experiences using the material in a bioinformatics course at University "Magna Græcia" of Catanzaro, Italy. Besides being appropriate for such a course, the module can provide a meaningful application for a high performance computing or a data structures class.

CCS Concepts

• Social and professional topics~Computing education • Theory of computation~Parallel algorithms • Theory of computation~Dynamic programming • Applied computing~Bioinformatics

Keywords

Computational Science; High-Performance Computing; Educational Modules; Blue Waters; Fulbright.

1. INTRODUCTION

In a Fulbright Specialist visit to University "Magna Græcia" of Catanzaro, Italy, in January 2015, Angela Shiflet and George Shiflet initiated a project with Mario Cannataro and Pietro Guzzi to develop educational modules on high-performance-computing bioinformatics algorithms. Wofford College student Daniel Couch, supported by a one-year internship with the Blue Waters Student Program, implemented the sequential and high-performance computing algorithms associated with the first two of the resulting modules.

DOI: https://doi.org/10.22369/issn.2153-4136/8/1/2

The first product of this collaboration is a module that uses algorithms to determine the highest similarity score and the alignments that yield this score for two DNA sequences. Sequence comparison to determine the similarity or difference of two sequences is a fundamental operation of the interdisciplinary field of bioinformatics. Bioinformatics, which relies on mathematics, statistics, and computer science, provides tools to compile, organize, and analyze the overwhelming volumes of data that are being generated from genomic studies. Because of the enormous quantity of data involved, high performance computing is an essential tool in bioinformatics. Similarities in gene sequences from different organisms, such as human and mouse, can help establish the function of the gene, and through sequence alignment, scientists can help establish the genetic causes of certain diseases. Moreover, sequence alignment is used to study protein functions and as a basis to predict protein structure.

The educational module, "Aligning Sequences–Sequentially and Concurrently," available at [6], develops the sequential Needleman-Wunsch Algorithm for sequence alignment and two parallel versions of the algorithm, the Pipeline Algorithm and the Block-and-Band Version of the Pipeline Algorithm. Also included in the module are the necessary biological background, quick review questions, exercises, and projects. The module was class tested in the course Advanced Techniques for Bioinformatics at University "Magna Græcia" in Spring, 2016, under the direction of Dr. Pietro Guzzi. This paper describes and examines the module and our experiences using it.

2. Module

2.1 Pedagogy

A variety of courses can incorporate the educational module "Aligning Sequences–Sequentially and Concurrently." For instance, a bioinformatics course can encompass the concepts and algorithms and consider or omit the programming components. The module can also be useful in an entry-level programming or data structures course to show an application of two-dimensional arrays. Moreover, a high-performance computing class can cover the module emphasizing the HPC algorithms and concepts.

The module provides the biological background necessary to understand the applications and references for further study. Eighteen (18) multi-part quick review questions throughout the module, with answers at the end of the module, provide immediate feedback. Nine (9) exercises give additional practice to aid understanding of various aspects of the algorithms. The module also provides five (5) project assignments for further exploration using sequential and/or parallel programming. Instructors can obtain implementations of the sequential algorithm in C and the parallel algorithms in C with MPI from [5] or the authors.

2.2 Biological Background

The introduction begins with a story of a woman diagnosed with breast cancer, possibly caused by inherited, mutated genes, and a general discussion of genes. Subsequent background sections are on “Nucleic Acids,” “Proteins,” “Connecting DNA Code to Protein Sequence,” “Mutations and Cancer,” and “Genomics and Bioinformatics.” The latter section indicates the importance of computation to biology by emphasizing that bioinformatics employs mathematics, statistics, and computer science to organize and store in databases vast amounts of data generated from genomic studies and to analyze that data.

Some of the biological material in the module will be familiar to some students but is included for students with minimal science backgrounds. Crucial to the understanding of the material is a basic understanding of deoxyribonucleic acid (DNA). DNA is a long chain of molecules, each containing a nitrogenous base, adenine (A), guanine (G), cytosine (C), or thymine (T).

Among the different bioinformatics algorithms, pairwise sequence alignment was chosen because it is largely used in various fields of biology, e.g. to highlight conserved DNA sequences or protein motifs along evolution, or as a basis of the protein structure prediction algorithms used to predict secondary and tertiary structure of proteins.

In particular, pairwise sequence alignment algorithms arrange the two input sequences (e.g. representing DNA, RNA, or proteins) to discover similar regions that may be due to functional, structural or evolutionary relations among the sequences. On the other hand

Moreover, sequence alignment algorithms use a very simple and intuitive metrics, i.e. the similarity among sequences, as a criterion to evaluate the quality of alignment. Finally, the chosen Needleman-Wunsch algorithm works on tabular data that is a data format very familiar to the students to whom the educational module is addressed.

2.3 Sequential Algorithm

Using bioinformatics, we can align DNA sequences to identify regions that are similar. Such a similarity might indicate that the two regions have the same function or evolve from a common ancestor in a sequence of mutations. In comparing two sequences, such as ATGAC and ACGC, we can employ a metric, called a similarity score, or score, to rate various alignments. For a scoring scheme, the highest possible similarity score indicates the best alignment(s). As the module discusses, an alignment of two DNA sequences has spaces in the sequences so that they are of the same length but so that a space in one sequence is not in the same position as a space in the other sequence. For example, with a dash (-) indicating a space, one alignment of *s* = ATGAC and *t* = ACGC follows:

s:	A	T	G	A	C
t:	A	-	C	G	C

Another possible alignment is as follows:

s:	A	T	G	A	C	-	-
t:	-	-	-	A	C	G	C

Although we can rate the quality of an alignment in many ways, this example in the module defines the score for an alignment as the total of column, or position, scores, where the column scores have the following values: +1 for a match, -1 for a mismatch, and -2 for a space in one of the corresponding positions. Adding all the position scores, the following alignment, with a dash (-) indicating a space, has a score of 1 + (-2) + (-1) + (-1) + 1 = -2:

s:	A	T	G	A	C
t:	A	-	C	G	C
column scores:	1	-2	-1	-1	1

The Needleman-Wunsch Algorithm is a technique to determine the similarity and the alignments that yield this score [4]. The algorithm employs dynamic programming, which divides a problem into a collection of smaller problems and uses the solutions to these smaller problems to solve the larger problem. The Needleman-Wunsch Algorithm makes the best decision for prefixes, or subsequences from the start of the sequences (the smaller problems), as it iterates over the length of those prefixes. The module used the notation *s*[*i..j*] to indicate the subsequence from position *i* to position *j*, where the first position number is 0. For example, in *s* = ATGAC, *s*[1..2] is TG.

We write the developing intermediate similarity scores in a two-dimensional array, or matrix, *a*. A blank (dash) and the bases of one sequence, such as *s*, are row labels, while a blank and the bases of the other sequence, such as *t*, label the columns. As indicated in Figure 1, in row 0 and column 0, we write the on-going scores for matching all spaces with prefixes of sequence *s* and *t*, respectively.

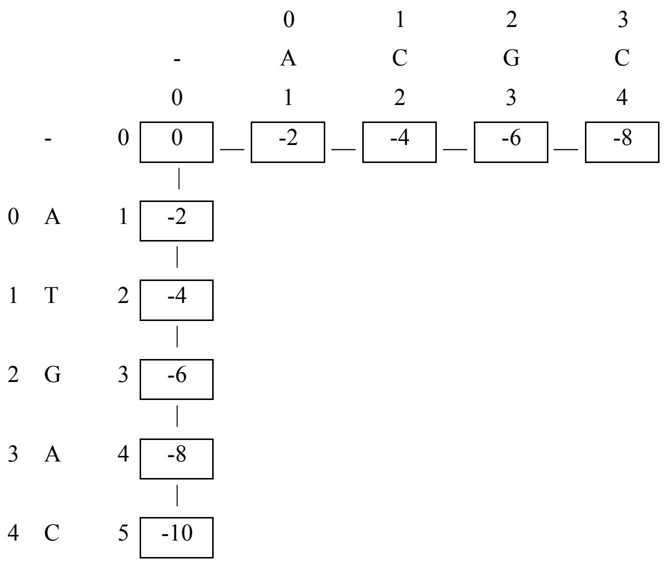

Figure 1. Initial values in similarity matrix

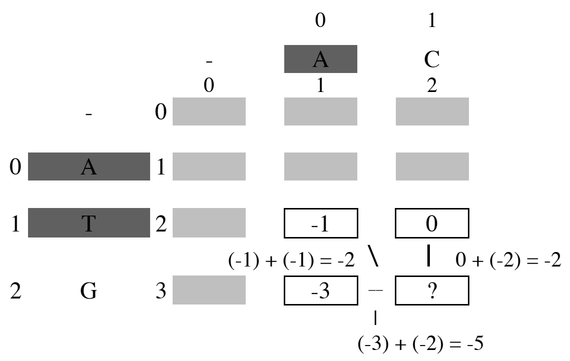

Figure 2. Determine *a*[3][2] from *a*[2][2], *a*[3][1], and *a*[2][1]

To determine the matrix scoring values, we proceed row by row, from left to right, calculating elements. Figure 3 contains the entire similarity matrix, with line segments marking the paths from the maximum element(s). The value in the bottom, right corner, 0, is the similarity of ATGAC and ACGC. Following line segments from that corner backward to *a*[0][0], we obtain a corresponding alignment for the sequences, such as the following optimal alignment:

s:	A	T	G	A	C
t:	A	C	G	-	C

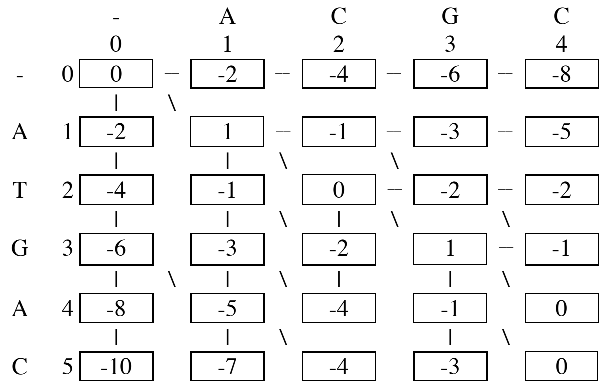

Figure 3. Array of similarity values for ATGAC and ACGC

2.4 HPC in Module

After covering this material, the module shows that employing a two-dimensional array, the complexity of the sequential algorithm is on the order of n^2, $O(n^2)$, where n is the length of a sequence. To illustrate the problem, the module displays a graph of timings using a C implementation of the Needleman-Wunsch Algorithm with an increasing number of nucleotides (Figure 4). When we are trying to match a sequence to multiple sequences in a database, the timing challenge of employing an $O(n^2)$ algorithm so often becomes evident.

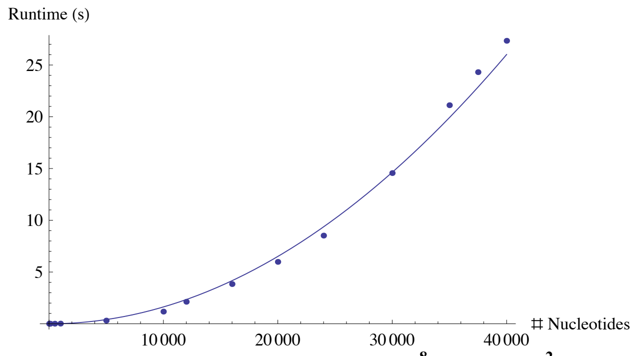

Figure 4 $\textit{runtime} = 1.62585 \times 10^{-8}\ \textit{nucleotides}^2$

Thus, discussion of complexity motivates the need for parallel processing. We can have different processes aligning the search sequence to different database sequences in a way called "embarrassingly parallel," and/or we can have a parallel alignment algorithm, such as the Pipeline Algorithm, to operate on each pair of sequences.

In the Pipeline Algorithm, for simplicity, the module assumes that the number of processes equals the number of rows, n, in the similarity matrix and that Process j is responsible for making the calculations on row j. The processes can simultaneously compute their first column elements without communication with the other processes using the formula j * *spacePenalty* for $a[0][j]$. Then, Process j, for $j = 0, 1, \ldots, n - 2$, can send $a[0][j]$ to Process $(j + 1)$. Similarly, Process 0 can compute the ith element in row 0 as i * *spacePenalty*. Immediately after calculation of $a[i][0]$, Process 0 can communicate the value to Process 1. Now, knowing the crucial values on the row above, $a[0][0]$ and $a[0][1]$, and the value to the left, $a[1][0]$, Process 1 can calculate $a[1][1]$. Moreover, while this calculation is occurring, Process 0 can be calculating its next value, $a[0][2]$. Then, Process 0 sends $a[0][2]$ to Process 1, and Process 1 sends $a[1][1]$ to Process 2, so that enough information will be available to occupy the first three processes. With each step, an additional process is drafted to work. Figure 5 depicts the progress of this pipelining system for sequences of length $n = 5$ and $m = 8$. After receiving communication of the value above, a process can start computation of its element in darker outline. Thus, calculation of values on this anti-diagonal can proceed in parallel [2].

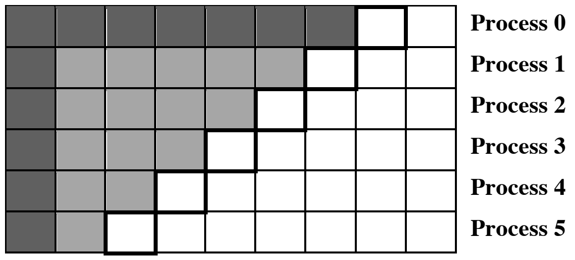

Figure 5 Pipelining the similarity matrix for sequences of length $n = 5$ and $m = 8$

After discussing the algorithm, the module considers the number of steps for n processes to calculate the similarity matrix for sequences of length n, $O(n)$. However, as the module illustrates, one disadvantage is the amount of communication, which is $O(n)$, too. Besides a theoretical discussion in the module, a table presents runtime and speedup results versus number of nucleotides for sequential and pipeline C implementations of the Needleman-Wunsch Algorithm. The table shows that with more than 10,000 nucleotides the pipeline algorithm is faster than the sequential one, but speedup is not linear. Increasing communication between processes dampens the runtime of the HPC version.

The problem of communication motivates consideration of the block-and-band version of the Pipeline Algorithm. To reduce communication, each process calculates a block of several column values. Moreover, as Figure 6 illustrates, we can make a process responsible for a band, or several rows, of elements. After calculating a submatrix, a process sends the block of elements in the last submatrix row to the next process so that the latter can start evaluation of a submatrix. Not only does a process transmit fewer elements, communication can involve a block of elements instead of multiple separate *send* operations, which is slower. One of the module's projects has the students implementing the algorithm and determining optimal block and band sizes. Consideration of the results can lead to a class discussion of scaling and load balancing.

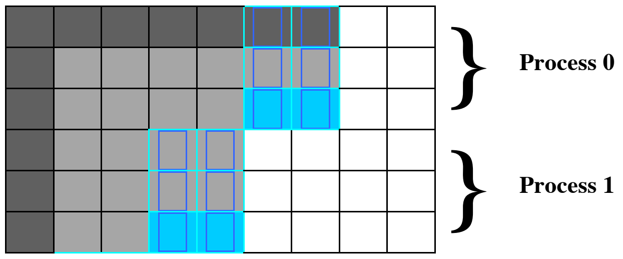

Figure 6 Pipelining a similarity matrix with block size of 2 and band size of 3

2.5 Reinforcement of Material

Eighteen, often multipart, Quick Review Questions throughout the module provide an assessment of the student's comprehension of the material. For example, one question has the students calculating the value of a scoring matrix element by hand. Another 17-part question has the student trace through the scoring algorithm using particular sequences. Answers at the end of the module provide immediate feedback to determine if the student is understanding the concepts.

Nine exercises provide additional reinforcement. For example, three exercises ask the students to develop entire scoring matrices for particular sequences, two involve complexity, and two consider modifications to the sequential algorithm.

Five projects have the students developing various sequential and/or parallel algorithms, performing timings, calculating speedup, and determining advantageous band and block sizes.

3. Class Testing

3.1 Class

In Spring, 2016, the course Advanced Techniques for Bioinformatics at the University "Magna Græcia" of Catanzaro in Italy covered the module. The course is a requirement during the last semester of their Master's Degree in Biomedical Engineering. The thirty (30) students in the class had BS degrees in biomedical engineering and, thus, had more skills on the biological side than in computer science. The students covered the module in one week with four class contact hours. A final assignment, done individually, was to implement the sequential version in Python and to compare their times to those in the module for the sequential and parallel implementations.

3.2 Results

After coverage of the material, seventeen (17) students completed a survey about the module. Table 1 gives the list of survey questions, eliciting a response from 1 for "strongly disagree" to 5 for "strongly agree," with the average scores. The responses were mostly very favorable but indicated some challenges with the parallel algorithms and programming.

Score	Statement
4.47	I understood the science in the module.
4.29	I understood the sequential algorithms in the module.
3.76	I understood the parallel algorithms in the module.
4.53	I understood the importance of using high performance computing.
4.47	The module was readable.
4.59	The Quick Review Questions helped me understand the material.
4.25	The exercises helped me understand the material.
3.13	The project helped me understand the material.

Table 1 Student survey averages (1 – strongly agree and 5 – strongly agree)

The survey also included the following questions that required free responses:

- Please elaborate about the above scores, particularly those below 4.
- What did you like best about the module?
- What did you find most difficult in the module?
- Please give corrections and suggestions for improvement.
- Please make further comments.

These responses indicated a desire by many students for the module to have more examples and explanation of the parallel algorithms. In response, the authors revised the sections on "Pipeline Algorithm" and "Block-and-Band Version of the Pipeline Algorithm" to include three specific, detailed examples and three additional multipart quick review questions with answers on the parallel versions (pipeline and block-and-band with bands of size 1 and greater than 1) of the algorithm.

The students' free responses contained numerous compliments. Students indicated that they liked the linkage between genomics and bioinformatics, the correlation between genetic mutations and cancer, and the discussion of bioinformatics. Several stated that the quick review questions, figures, tables, and pseudocode for the Needleman-Wunsch Algorithm were helpful. One student commented, "The way in which alignment algorithms were explained in the module is better than other articles I read, I really like it," and another stated, "The article was interesting, from which we learned new concepts." About one-third of the students volunteered that the module was "interesting" or "very interesting."

4. CONCLUSION

Based on the survey responses and the students' performance, the module, "Aligning Sequences–Sequentially and Concurrently," accomplishes conveying some of the basic principles of bioinformatics, the Needleman-Wunsch Algorithm, HPC versions of the algorithm (pipeline and block-and-band), and the utility of high performance computing. Moreover, the students understood and appreciated the material and successfully completed the assignment. As a result of their suggestions, the authors improved the module, making it an even better educational module incorporating HPC topics in the context of other applications learning.

5. ACKNOWLEDGMENTS

Our thanks go to the Fulbright Specialist Program, University "Magna Græcia" of Catanzaro, and Wofford College for funding the Shiflets' visit to the university and to the National Computational Science Institute Blue Waters Student Internship Program for funding Daniel Couch's internship.

6. REFERENCES

[1] Cannataro, M., and Guzzi, P. 2011. *Data Management of Protein Interaction Networks*, Wiley.

[2] Chen, Y., Yu, S., and Le, M.. 2006. "Parallel Sequence Alignment Algorithm for Clustering System" in *International Federation for Information Processing (IFIP)*, Volume 207, *Knowledge Enterprise: Intelligent Strategies In Product Design, Manufacturing, and Management*, eds. K. Wang, Kovacs G., Wozny M., Fang M., (Boston: Springer), pp. 311-321.

[3] National Computational Science Institute Blue Waters Student Internship Program. 2016. `http://computationalscience.org/bwsip/` Accessed April 6, 2016.

[4] Needleman, S. and Wunsch, C. 1970. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *J. Molecular Biology*, vol. 48, pp. 443-453.

[5] Website associated with Shiflet, A. and Shiflet, G. 2014. *Introduction to Computational Science: Modeling and Simulation for the Sciences, 2nd ed.*, Princeton University Press `https://ics.wofford-ecs.org/` Accessed April 6, 2016.

[6] Shiflet, A., Shiflet, G., Couch, D., Guzzi, P., and Cannataro, M. 2016. "Aligning Sequences–Sequentially and Concurrently" `https://wofford-ecs.org/` Accessed April 6, 2016.