A BIOINFORMATICS EXPERIENCE COURSE*

Charles Toth and Richard Connelly
Biology and Mathematics/Computer Science Departments
ctoth@providence.edu and rconnell@providence.edu
Providence College
Providence, RI 02918

ABSTRACT

The course is an upper level elective that pairs a biology major with a computer science major. The biology and computer science components are loosely coupled. The former immerses students in an on-going research project while the latter introduces programming to biologists and demonstrates the complexity of sequence analysis. The goal is to comprehend the challenges in the other discipline from peer-to-peer experience.

1. COURSE STRUCTURE AND GOAL

The course is an upper level elective for computer science and biology students. It is designed for an even distribution of biologists and computer scientists. Each student is teamed with one from the other discipline. The goal is to have each student comprehend the challenges in the other discipline from peer-to-peer experience.

The course has loosely coupled biology and computer science components. The biology component immerses the students in an on-going research project that requires both the wet science of molecular biology and internet-based data mining and sequence analysis. The computer science component is a series of labs that introduce programming to the biologists and use established dynamic programming methods to demonstrate the complexity of sequence analysis. Both components use student presentations to evaluate performance.

The interdisciplinary student team concept is borrowed from LeBlanc and Dyer ([17]). Also, experience with a poster project ([30]) clearly showed that the team method would much improve the learning experience.

The course is designed to be stand-alone. It differs from the "infusion" model ([16, 17]) where bioinformatics is incorporated into the cores of each discipline. The course is also not intended to approximate an undergraduate bioinformatics curriculum ([3-5, 9, 20, 26, 27]).

2. BIOLOGY COMPONENT

The students were immersed in an on-going research project that provided experience in how bioinformatics is utilized by scientists engaged in basic research. A project was selected that struck a balance between the wet science of molecular biology and the computer-based work of database mining and DNA sequence analysis. Other undergraduate bioinformatics courses link existing computer science and biology courses to demonstrate bioinformatics, but this course directly demonstrated the major role that bioinformatics plays in genomics research. The decision to use a research project instead of stand alone labs was to directly demonstrate to the students the integration of different scientific disciplines.

In general, the biology portion of the course examined how changes in the growth rate of an organism affect the regulation of genes involved in proliferation. The research project for the course was the study of lichen growth properties during nutrient starvation ([23]). The hypothesis was that lichens growing on a rock substrate would undergo a diauxic shift and alter its gene expression profile. A diauxic shift occurs when an organism switches to a different metabolite. Other Ascomycetes such as *Saccharomyces* show a distinct gene regulatory pattern in which mitochondrial oxidative phosphorylation enzymes are upregulated in response to glucose depletion as the fungus changes its metabolic program ([8]). Thus, the aim of the lichen experiment was to determine if ATP synthase and cytochrome C oxidase genes show the same upregulation in the fungal lichen species. There is minimal genomic information for the fungal genus, *Parmelia*, that grow on rock substrates in New England; thus, this project was a perfect way to introduce the use of genomic databases as a tool to study gene expression in related organisms ([10]).

In order to study lichen gene expression, database mining and DNA analysis was required to identify *Parmelia* gene sequences using homologous sequences from related Ascomycetes organisms. Each student group picked a unique mitochondrial gene to study so that the students were performing the same tasks but doing unique work. The student work groups were quickly immersed in internet-based bioinformatics web sites to obtain the information needed to generate the genomic data. The students were trained in direct hands-on experience on how to manipulate and traverse through the annotated databases to find the correct information and analyze it to generate multiple gene alignments for their projects. The final outcome for the students was the creation of degenerate DNA oligonucleotide sequences that matched conserved regions of their fungal mitochondrial gene of choice.

At this point in the project, the students entered into the molecular biology portion of the biology component. The students performed all of the experimental protocols themselves with direct training and supervision of the biology professor. They were able to generate usable data and learn how to trouble shoot and optimize procedures since the

work was part of an on-going project and not based on lab exercises or educational kits. A drawback of using an on-going research project is that oftentimes the project requires optimization or troubleshooting in order to make sufficient progress. This can be helpful is demonstrating to students the nature of scientific research, but it can reduce the effectiveness of the training. Since the completion of the bioinformatics course, the research project was completed and a manuscript is in preparation. In the future, the course will run smoother by using protocols and reagents that have previously been shown to produce usable results.

3. COMPUTER SCIENCE COMPONENT

The computer science portion was done as a series of labs. The secondary, but initial, objectives were to introduce programming to the biologists, present *Perl* ([25]) to the computer scientists, and use regular expressions in bioinformatics. The primary objective was to show the computational complexity of sequence analysis by learning the dynamic programming methods in the *Needleman-Wunsch (NW)* and *Smith-Waterman (SW)* alignment algorithms (Chap. 2, [14]). These were then used to introduce the methods in *BLAST* ([2]). A Folding@Home ([24]) demonstration further illustrated complexity.

All the computer scientists were experienced tutors. Traditional *Computer Science 1* and *2* labs were rewritten for *Perl* with additional bioinformatics examples. The instructor did a conceptual introduction and then the computer scientists taught programming to the biologists as they learned *Perl*. Regular expressions were new to all, but the same pedagogy was used with new labs. These labs emphasized finding open reading frames (Chap. 1, [14]).

The recursive *Fibonacci* algorithm was used to introduce dynamic programming. Students used traditional call stack and call tree methods to learn how *memoization* ([22]) improves performance. Paper and pencil exercises (based on Chap. 2, [14]) were then used to teach the dynamic programming techniques in the *NW* and *SW* algorithms. *Perl* implementations of the algorithms were available for reference.

Starting from the same examples, each team was asked to present how regular expressions are used to find open reading frames and to present the dynamic programming techniques in the *NW* alignment algorithm. The teams were asked to split the presentation evenly between members.

The student presentations were conceptual because it is unrealistic to expect new programmers (biologists) to develop practitioner skills in recursion and matrices in one half of a semester.

4. PREREQUISTES, PEDAGOGY, EVALUATION, FUTURE

There were no formal prerequisites, but all the computer science students had taken general biology in high school and/or college and one biology student had taken *Computer Science 1*. All the computer science students were tutors and most of the biology students were lab assistants. The course was dually coded so that each student could take it as an elective in his/her major.

There were no required or recommended texts. The primary computer science sources were a general *Perl* text ([18]), two "*Perl* for biologists" texts ([12, 29]), and Chapters 1 and 2 of *Fundamental Concepts of Bioinformatics* ([14]). The latter is a rich source for concepts and practice exercises. Single session computer science labs were used in order to interleave with the slightly unpredictable schedule inherent in research. The biology portion used the research itself for background concepts, but in the future selected chapters from a general biology text and research protocols will be utilized to provide sufficient background for the students.

The students were very motivated and enthusiastic. Their work was exceptional and clearly revealed that each student had spent considerable effort to thoroughly understand the material in the other discipline. Five-sixths of the class completed the course evaluation and the responses were very complimentary and tolerant of "new course" challenges. As might be expected in an "experience course", the responses indicated that students were occasionally "over their heads" in the other discipline. Specific suggestions included requiring an alignment algorithm project (from a biologist) and suggestions for tightening the components and improving delivery.

Course repetition will tighten the components and smooth the delivery. Most students take general biology, so it is a reasonable prerequisite for computer scientists. The schedule makes it difficult for biology students to take *Computer Science 1*, so it is not a reasonable prerequisite. The disciplines intersect at the internet sequence analyzers (*BLAST*, *CLUSTAL*, etc.); consequently, recursion and matrices are the underlying computer concepts. In the future, each team will be asked to extend several labs into a software project, but the biologist will be an "apprentice" rather than using the traditional "group" structure. An *Introduction To Bioinformatics Algorithms* ([13]) has exemplary extensions of the *NW* algorithm (Chap. 6).

An "experience course" is a practical response to multi-disciplinary and emerging technologies. It is consistent with the expectations of today's students and today's marketplace ([19]). It is also consistent with the education of early bioinformatics practitioners ([13]).

BIOLOGY SYLLABUS

Week	Biological Objectives	Computer Objectives
1	Identification of homologous mitochondrial genes from Ascomycetes	Database mining of *Saccharomyces* genome ([21]), *BLAST* ([2]) analysis of *Saccharomyces* genes to identify fungal homologs, *Clustal* ([7]) multiple alignment analysis for DNA lineups
2	Creation of degenerate DNA oligonucleotides for PCR amplification from week 1 *Clustal* results	Oligonucleotide analysis software for determination of melting temperature, hairpin formation, heterodimer formation ([11]).

3	Lichen RNA isolation	
4	cDNA preparation	
5	PCR amplification of lichen cDNA pools, horizontal gel electrophoresis	
6	Optimization of PCR protocols for primer concentration, annealing temperature, Mg2+ concentration	
7	Real-time PCR analysis of lichen gene expression	Use of *Stratagene MX3000p* ([28]) software for real-time PCR cycling protocols
8	Optimization of real-time PCR running conditions	
9	Quantization of lichen gene expression using real-time PCR	Use of *Stratagene* software for comparative gene quantization

COMPUTER SCIENCE SYLLABUS

Topic	**Objective(s)**	**Method(s)**
Perl introduction	Computer Scientists: Learn syntax and scalars. Biologists: Learn syntax, assignment, print, and scalar types from computer scientists.	Multi-part lab with many "what if" exercises (e.g., what if a string scalar is used in arithmetic).
Regular expression introduction	Learn regular expression syntax and substitution and translation techniques.	Use substitution to convert a DNA sequence to RNA and use translation to create the complement sequence.
Conditional statement	Biologists learn from computer scientists.	Traditional *CS1* "what if" lab.
Iteration	Biologists learn from computer scientists.	Traditional CS1 lab using factorial and *Fibonacci* numbers. Bioinformatics lab that calculates the percentage of G's and C's in a DNA sequence.

Regular expression matching techniques	Find a start codon and its stop codon. Find open reading frames (ORFs).	Lab to find start and stop codons. Lab to find the ORFs in a RNA sequence, regardless of starting nucleotide.
Functions[1]	Biologists learn from computer scientists.	Traditional *CS1* lab using a factorial function to calculate a combination.
Recursive functions	Biologists learn from computer scientists.	Traditional CS2 lab using factorial, *Fibonacci* numbers, and combinations. Students trace the call stack using the *Perl* debugger and paper and pencil techniques.
Dynamic programming[1,2]	Use *memoization* ([22]) to introduce dynamic programming.	Use a global array in the recursive *Fibonacci* algorithm to eliminate recalculating a *Fibonacci* number. Trace the call stack to see the improvement.
Needleman Wunsch (NW) alignment	Learn *NW* global and semi-global alignment methods ([14], Chap. 2).	Use paper and pencil to "dynamically program" alignment matrices. Implementations[3] are available for demonstrations.
Smith Waterman (SW) alignment	Learn *SW* local alignment method ([14], Chap. 2).	Use paper and pencil techniques to "dynamically program" alignment matrices. An implementation is available for demonstration.
Advanced alignment	Learn additional match-mismatch scores and gap penalties ([14], Chap. 2). Introduce *BLAST* search methods ([6], Chap. 7).	Use paper and pencil to calculate match-mismatch and gap scores. Use *lalign* ([15]) to show gap penalties and *BLAST* match-mismatch scores.
Bioperl ([1]) Introduction	Demonstrate download, *FASTA* ([6], p. 50), and *BLAST* features.	Scripted demonstration lab.
Folding@Home ([24])	Demonstrate computational scale of protein folding.	Temporarily install and run *Folding@Home*. Show the simulation time of a protein fold and the need for distributed computation. Use the Windows Task Manager to show zero idle cycles.

[1] Labs also introduce arrays in Perl.
[2] Lab also introduces references and reference parameters.
[3] The programs also show how to create a matrix in Perl.

REFERENCES

1. Bioperl. http://bio.perl.org/.
2. BLAST. http://www.ncbi.nlm.nih.gov/BLAST.
3. Buffalo. *University of Buffalo Bioinformatics Curriculum*. http://wings.buffalo.edu/academic/department/fnsm/bio-sci/overview.html.
4. Burhans, D.T. and G.R. Skuse, The Role of Computer Science in Undergraduate Bioinformatics Education. *SIGCSE*, 2004. 36(1): p. 417-421.
5. Canisius. *Canisius College Bioinformatics Curriculum*. http://www.canisius.edu/bif/curriculum.asp.
6. Claverie, J.M. and C. Notredame, *Bioinformatics for Dummies*. 2003: John Wiley.
7. Clustal. http://www.ebi.ac.uk/clustalw/index.html.
8. DeRisi, J.L., V.R. Iyer, and P.O. Brown, Exploring the Metabolic and Genetic Control of Gene Expression on a Genomic Scale, in *Science*. 1997. p. 680-686.
9. Geocities. *Bioinformatics Courses Worldwide*. http://www.geocities.com/bioinformaticsweb/coworld.html?200513.
10. Hinds, J.W. *Lichen Flora of Eastern North America: The Genus Parmelia Sensu Stricto*. Eastern Lichen Network, DOI: ElectronicLichen Flora of Eastern North America: The Genus Parmelia Sensu Stricto. Resource Number
11. IDT. *Integrated DNA Technologies*. http://www.idtdna.com/SciTools/SciTools.aspx.
12. Jamison, D.C., *Perl Programming for Biologists*. 2003: John Wiley & Sons.
13. Jones, N.C. and P.A. Pevzner, An Introduction To Bioinformatics Algorithms. 2004: MIT Press.
14. Krane, D.E. and M.L. Raymer, *Fundamental Concepts of Bioinformatics*. 2003: Benjamin Cummings.
15. lalign. http://www.ch.embnet.org/software/LALIGN_form.html.
16. LeBlanc, M.D. and B.D. Dyer, Bioinformatics and Computing Curricula 2001: Why Computer Science is Well Positioned in a Post-genomic World. SIGCSE, 2004. 36(4): p. 64-68.
17. LeBlanc, M.D. and B.D. Dyer, Teaching Together: A Three-Year Case Study in Genomics. *The Journal of Computing Sciences in Colleges*, 2003. 18(5): p. 85-95.
18. Lerner, R.M., *Core Perl*. 2002: Prentice Hall PTR.

19. Lohr, S., A Techie, Absolutely, And More. *New York Times*, Aug. 23, 2005, p. C1-C2.

20. Morgan, E. *Bioinformatics Courses Around the World*. http://wbiomed.curtin.edu.au/teach/biochem/resources/Bioinformatics.html.

21. NIH. *Saccharomyces Genome Database*. http://www.yeastgenome.org/.

22. NISF. *Dynamic Programming - Memoization*. http://www.nist.gov/dads/HTML/memoize.html.

23. Palmquist, K., Carbon Economy in Lichens. *New Phytol*, 2000. 148: p. 11-36.

24. Pande, V. *Folding@Home*. http://folding.stanford.edu.

25. Perl. http://www.perl.com/.

26. Ramapo. *Ramapo College Bioinformatics Curriculum*. http://www.ramapo.edu/catalog_04_05/academicPrograms/TAS/bioinfo.html.

27. RIT. *Rochester Institute of Technology Bioinformatics Curriculum*. http://www.rit.edu/~932www/ugrad_bulletin/colleges/cos/bioinfo.html.

28. Strategene. http://www.stratagene.com/homepage/.

29. Tisdall, J.D., *Beginning Perl for Bioinformatics*. 2001: O'Reilly & Associates.

30. Wray, K.A., Perl Algorithm to Calculate and Categorize PHI and PSI Angles in a Protein. *The Journal of Computing Sciences in Colleges*, 2005. 20(5): p. 98-99.