Computing for Biologists: lessons from some successful case studies *

Dennis Shasha
Courant Institute of Mathematical Sciences
New York University
New York, NY 10012
shasha@cs.nyu.edu

ABSTRACT

My presentation will be online at the address http://cs.nyu.edu/cs/faculty/shasha/papers/sigmodtut05.ppt in addition to at the SIGMOD site. The presentation discusses computational techniques that have helped biologists, including combinatorial design to support a disciplined experimental design, visualization techniques to display the interaction among multiple inputs, and the discovery of gene function through the search through related species, and others.

In this writeup, I confine myself to informal remarks describing both social and technical lessons I have learned while working with biologists. I intersperse these comments with references to relevant papers when appropriate.

The tutorial is meant to appeal to researchers and practitioners in databases, data mining, and combinatorial algorithms as well as to natural scientists, especially biologists.

Lesson 0: Biology is more interesting than you remember it

If you disliked dissecting frogs in high school, have no fear. Modern biology is mostly about genes and proteins and the inference of function. There will be no anatomy or terminology tests. Biologists need us because their data no longer fits into lab notebooks. They have become avid users of database, information retrieval, and algorithmic tools.

Be warned: there is a culture shock. You'll find the data noisy and the theories a little fuzzy, but the great thing about biology is that the discovery of a qualitative tendency (e.g., gene X is critical for cell repair) can have enormous impact even when the quantitative results lack precision. Once the qualitative phenomenon is discovered, quantitative modeling can follow. That modeling is often discrete, because the data is usually too noisy for differential models.

*Work supported in part by U.S. NSF grants IIS-9988636, N2010-0115586, as well as MCB-0209754.

SIGMOD 2005 June 14-16, 2005, Baltimore, Maryland, USA.

Lesson 1: Start with a real biologist

Too many of my computer science colleagues think they can imagine a problem and then take it to the biologists who will adopt it with gratitude and joy. This working method can work, but rarely. It is simply far more productive to start working with a team of biologists by sitting in on their meetings. Note that I said *their* meetings, because that is when you hear their real problems rather than the problems that they distill for you when they meet you in an interdisciplinary symposium.

During those meetings, you will find out quickly how you can make yourself useful and thereby gain credibility. For example, one of the topics discussed in my tutorial (the use of adaptive combinatorial design[4]) came about because my colleagues were wondering how to explore a large experimental space without spending too much money or consuming too much time. I proposed combinatorial design as an intelligent sampling approach and the use of feedback from early experiments to refine the sample. I implemented a rough version of the algorithm within a few days, and, yes, they adopted it with joy.

Once you gain credibility, then of course you can bring to the biologists' attention tools that have been developed for general applications. I've done this for example with the choice of clustering algorithm, as well as graph and tree discovery software [7, 3, 6] (see also http://cs.nyu.edu/cs/faculty/shasha/papers/GraphClust.html). The biologists want to hear your advice, but only after they are convinced you listened to their problems.

Lesson 2: Caring about data

To work well with biologists, you must appreciate that they care about data; computer scientists by and large don't. We computer scientists would like to invent a tool that many people will use – many *other* people that is.[1] We don't usually care for which purpose. This sometimes results in bizarre conversations.

For example, in one project, I had written a program to find pairings between proteins called transcription factors and binding sites [1] Since the program was under development, I would normally run it and then send the result data back. Naturally (for a computer scientist), I wouldn't even glance at the data first. At a meeting a few weeks into this

[1]If you don't believe this, ask your colleagues from IBM, Oracle, and Microsoft how often they actually use their database management systems.

project, my biological colleauges were very excited. "Look at these transcription factor-binding pairs," they said. "They agree with the literature and make interesting predictions."

"That's pretty cool," I replied. "Where did you get that stuff?"

"Your program of course," they said, looking at me with eyebrows high on their foreheads.

Lesson 3: Be interdisciplinary in every way

Biologists' interest in data suggests a natural opportunity for database people: collaborate by consulting on data management issues or house the databases. But don't stop there. To be sure there may be a need for better query languages for biology, but the big new problems fall outside pure database concerns.

You have to be interdisciplinary within computer science too. There are challenges such as experimental design, information visualization, statistical analysis, and machine learning[2]. Many of the machine learning issues, by the way, have to do with the inference of graphs (e.g. networks of protein-protein interactions) or analog circuits (e.g. transcription factor causality directed acyclic graphs). So, talk to your local biologist if you're good at such things.

Lesson 4: Save the best problems for your graduate students

A lot of the problems you will have to deal with will not be considered to be research issues by computer scientists. You will be called upon to help data parsing problems, data schema design issues, decisions about server architectures, and so on. Every once in a while, a jewel of a problem falls out. Consider reserving the jewels for your graduate students even if you end up doing some utility programming yourself.

A contrary approach is to lean on the programmers hired by the biologists to do low-level bioinformatics, but this removes you too much from the real work. When you do something directly useful, you gain your colleagues' trust and respect even when it is easy.

If you decide to follow this lesson, know that you have to be responsive. I try to turn around small requests in a day or less. If the best answer takes a month, but a very good one takes a day, go for the good one. This leads to the next lesson.

Lesson 5: People time is everything

Biologists are pragmatic people. When they solve problems, they think in terms of data gathering times which are days, weeks, or months. They do not care if an algorithm can be improved from 20 minutes to 10 minutes. They will care if an exponential algorithm can be made polynomial, but only if this will save them time. They will care even more if you can save them time and/or money perhaps by using machine learning to suggest the wet lab experiments likely to yield the most information. In one application, a group of us inferred the functionality of bacterial genes by looking at similar genes across species[5]. This led to the identification of a gene that had a good chance of being involved in a certain bacteria's ability to swim. Knocking that gene out did have the desired effect – the bacteria didn't move much. The biologists doing the knockout would not have wanted to do the work without the computationally generated hypothesis. On the other hand, nobody would have trusted the computational prediction alone.

Lesson 6: Meet regularly and informally

A good interdisciplinary research relationship requires a certain level of comfort. Many biologists have a certain fear of mathematics. You probably never liked chemistry that much. You have much to learn from one another. In my collaborations, we meet every week for two hours in a small conference room with cookies from one of the local Italian bakeries. We laugh, we argue, and we ask stupid questions. Sometimes we come up with good ideas.

Acknowledgments

Warm thanks to my wonderful colleagues in the established biology labs of Benfey, Coruzzi, and Piano, the up-and-coming labs of Birnbaum, Lejay, Palenchar and Gutierrez, and the biomedical labs of Shashoua and Greenes. Biological work is long and fraught with hard work lost. My co-authors always face such problems with equanimity and humor.

1. REFERENCES

[1] Kenneth Birnbaum, Philip N. Benfey, and Dennis Shasha. cis element/transcription factor analysis (cis/tf): A method for discovering transcription factor/cis element relationships. *Genome Research*, 11:1567–1573, 2001.

[2] Kenneth Birnbaum, Dennis E. Shasha, Jean Y. Wang, Jee W. Jung, Georgina M. Lambert, David W. Galbraith, and Philip N. Benfey. A gene expression map of the arabidopsis root. *Science*, pages 1956–1960, 2003.

[3] D. J. Cook and L. B. Holder. Substructure discovery using minimum description length and background knowledge. *Artificial Intelligence Research*, 1:231–255, 1994.

[4] Laurence V. Lejay, Dennis E. Shasha, Peter M. Palenchar, Andrei Y. Kouranov, Alexis A. Cruikshank, Michael F. Chou, and Gloria M. Coruzzi. Adaptive combinatorial design to explore large experimental spaces: approach and validation. *Systems Biology*, 1.2:206–212, 2004.

[5] Mitchell Levesque, Dennis Shasha, Wook Kim, Michael G. Surette, and Philip N. Benfey. Trait-to-gene: A computational method for predicting the function of uncharacterized genes. *Current Biology*, 13:129–133, 2003.

[6] D. Shasha and R. Giugno. Graphgrep: A fast and universal method for querying graphs. *Proceeding of the International Conference in Pattern recognition (ICPR)*, pages 112–115, 2002.

[7] X. Yan, P. S. Yu, and J. Han. Graph indexing: A frequent structure based approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 335–346, 2004.