Gene Discovery in DNA Sequences

Steven L. Salzberg, The Institute for Genomic Research

GENOMICS OFFERS TREMENDOUS CHALLENGES AND OPPORTUNITIES FOR COMPUTATIONAL SCIENTISTS. ADVANCES IN BIOTECHNOLOGY HAVE PRODUCED ENORMOUS VOLUMES OF DNA-RELATED INFORMATION, BUT THE RATE OF DATA GENERATION IS OUTPACING SCIENTISTS' ABILITY TO ANALYZE THE DATA. THE AUTHOR DESCRIBES TWO COMPUTATIONAL TECHNIQUES TO SOLVE THE GENE-FINDING PROBLEM.

GENOMICS IS THE SCIENCE OF studying the complete genetic content of living organisms. The field has come to prominence in the past few years, as advances in DNA sequencing technology have made decoding whole genomes possible. A genome is the complete DNA complement of an organism, which contains the instructions that control virtually everything about how the organism lives: development, metabolism, aging, sensitivity to infection, and so on.

As recently as 1995, scientists at the Institute for Genomic Research (TIGR) completed and published the very first complete genome sequence of a free-living organism, the bacterium *H. influenzae*.[1] Since the publication of that genome, scientists at TIGR and elsewhere have published 25 more complete genomes and chromosomes, and projects to complete the genomes of much larger organisms have progressed rapidly.

Current estimates indicate that sequencing of the entire human genome, over 3.3×10^9 nucleotides, will be completed within the next two to three years. Bacterial genomes, typically in the 1 to 4 megabase (million base) range, are being sequenced at a rate of about one per month, with approximately 75 genome projects under way worldwide. Sequencing of the genomes of the nematode *C. elegans* and the fruit fly *D. melanogaster*, each over 100 megabases long, is very close to complete and should appear within the next year. Generating a complete sequence is a major scientific project, involving many scientists and technicians and years of coordinated efforts.

Accompanying this explosion of new data has been the rapid growth of the computational biology field, including the area known as computational genomics. With the DNA sequence describing so many complex processes, it is only natural that computational methods have become a vital tool in scientists' efforts to understand how life works.

Expertise in the emerging area of computational biology requires training in two traditionally distinct fields: computer science and molecular biology. Scientists from both disciplines, as well as a handful of scientists from areas such as physics and chemistry, have moved into the field. Although a tutorial on molecular biology and genetics is beyond the scope of this brief article (tutorials are available both online and in print[2]), a few basic definitions are necessary for this discussion (see the "How DNA produces proteins" sidebar).

One of the central discovery challenges for newly sequenced DNA is that of identifying the genes (the portions of the DNA that get transcribed to RNA), the protein products they produce, and the functions of those proteins. Genome projects today use computational techniques almost exclusively to identify

1094-7167/99/$10.00 © 1999 IEEE

genes, because traditional laboratory techniques are very slow and expensive. In many cases, genes identified by a computer will require laboratory verification, but in the world of high-throughput DNA sequencing, scientists rarely have time to wait for verification before publishing. In this article, I describe the two main computational techniques used for discovering new genes: sequence alignment and computational gene finding.

How DNA produces proteins

DNA is a very long molecule made by concatenating four other complex molecules called nucleotides, themselves comprising several smaller units. The nucleotides come in four varieties defined by the *bases* that distinguish them: adenine (a), guanine (g), cytosine (c), and thymine (t). This four-letter code is the basis of life: every living organism contains DNA in the form of one or more long chromosomes. DNA is double-stranded, and the two strands are paired according to the simple rule that a is always paired with t, and c is paired with g. This base-pairing rule means that one strand completely determines the other, and consequently either strand can serve as a template for making a copy of the double-stranded molecule. Whenever a cell divides, the two strands divide and form such templates, which make two complete double-stranded copies of the parent cell's chromosomes.

The process by which DNA governs the workings of a cell is extremely complex, and at present scientists understand only tiny slices of the process in any great detail. One of the fundamental processes is that DNA is transcribed into an intermediate form known as RNA. A *gene* is the segment of DNA that gets transcribed. The RNA does not last long before being degraded; but while it exists, RNA serves as a template for the protein-making machinery.

Proteins, the essential molecules that do most of a cell's work, are strings of amino acids from the 20-letter amino acid alphabet. To make a protein, a molecular machine called the ribosome translates RNA three bases at a time into a protein sequence, using a code that uniquely maps every three-base combination (codon) onto an amino acid. The genetic code maps 61 of the 64 possible codons onto the 20 amino acids; some amino acids are encoded by multiple codons. Three special "stop" codons do not encode anything: when the translation machinery hits one of these, the process halts and the protein is complete. These proteins then fold up into complex 3D structures, some of which migrate to different places in the cell or even leave the cell entirely to do their work.

Sequence alignment

The first task is to compare each new DNA sequence with all known sequences to find other similar sequences. When scientists find a sequence with good similarity, they can reasonably infer that the new sequence has a similar function to the previously known gene. Thus, without any additional laboratory work, new biological knowledge can be made available.

Sequence alignment is simply a special case of string matching, a research field with a long history in computer science. The alignment process determines the edit distance between two strings, where the editing process in this case represents the evolutionary changes that have modified the sequences. In molecular biology, algorithms for alignment date back to 1970,[3] and efficient algorithms began appearing in the early 1980s.[4] An example alignment of two sequences appears in Figure 1 (which uses English text rather than DNA to simplify this discussion).

A sequence alignment shows where two sequences match and finds the best places to insert gaps and bring subsequences in line with one another. As the figure shows, gaps might appear in either sequence. Inserting gaps usually incurs a penalty, and the cost of gaps greatly impacts how the optimal alignment looks. For example, the first alignment in Figure 1 has four gaps and 13 characters that match. The second alignment has only two gaps and 11 matching characters. If a match's value exceeds a gap's cost, then the first alignment produces a higher score.

The key technique in the algorithm for optimally aligning two sequences is dynamic programming, which lets biologists align sequences of length M and N in time $O(NM)$. The algorithm to align two sequences fills a matrix defined by sequence A along the rows (giving M rows) and sequence B along the columns (N columns). Then each entry in the matrix is filled, proceeding left to right and row by row, as follows:

$$D[i,j] = \max\begin{cases} D[i-1,j]+b \\ D[i,j-1]+b \\ D[i-1,j-1]+S[A_i,B_j] \end{cases}$$

where b is the penalty for inserting a gap of one character in either sequence, and $S[A_i,B_j]$ is the score defined by the alignment of the characters at position i in sequence A, and j in sequence B. As the formula shows, only three positions in the matrix are important for filling a given entry: up, left, and diagonally up and left. A diagonal move corresponds to aligning the next pair of characters from A and B; horizontal and vertical moves correspond to adding or extending gaps.

The Smith-Waterman algorithm, one of the most widely used methods for optimal alignment, permits variable-length gaps in each sequence and provides a gap penalty function of the form $a(x-1)+b$, where b is the cost of opening up a gap, a is the cost (usually small) of extending a gap by one position, and x is the gap size. The intuition behind this important function is that the evolutionary forces that change a sequence are as likely to insert (or delete) several characters as they are to insert just one character. Thus, the penalty for creating a gap should be fairly large, but that for extending a gap should be small.

As the amount of sequence data grew throughout the 1980s, researchers needed faster comparison algorithms. William Pearson and David Lipman observed that a hashing approach would run much faster, although it would not guarantee an optimal alignment.[5] To put this in context, consider that the most common use of alignment programs is for aligning a single sequence against a large database of sequences. The hashing approach preprocesses the database by identifying, for all substrings up to a fixed length, those substrings' locations in the database. We can then find, in constant time, the location of each substring appearing in a query, wherever it occurs in the database. Once we have hashed the query to the match-

```
A:  --T----HISISNOTADNA----SEQUENCE
      |      | ||    |        ||||||||
B:  BUTMAYBEITIS---APROTEINSEQUENCE

A:  ------THISISNOTADNA-SEQUENCE
          | ||         ||||||||
B:  BUTMAYBEITISAPROTEINSEQUENCE
```

Figure 1. Two alternative alignments of two short sequences of English text. Vertical bars (|) indicate characters that match; dashes (–) indicate gaps.

Table 1. Performance of Glimmer on six microbial genomes. The additional-genes column reports the number of genes found by the system but not included in published annotation; further investigation is needed to determine if these are false positives or genuine genes.

Organism	Associated disease (if any)	Number of genes	Annotated genes	Additional (false?) genes
H. influenzae	Ear infections and meningitis	1,738	1,720 (99.0%)	242 (14%)
M. genitalium	Urethritis	483	480 (99.4%)	82 (17%)
M. jannaschii		1,727	1,721 (99.7%)	218 (13%)
H. pylori	Chronic ulcers and gastritis	1,590	1,550 (97.5%)	322 (20%)
B. burgdorferi	Lyme disease	849	845 (99.5%)	67 (8%)
T. pallidum	Syphilis	1,039	1,012 (97.4%)	180 (17%)

ing locations in the database, a second process takes over and extends the matches. The bulk of the work is spent here, exploring these potential matches and deciding which ones to report back. Pearson and Lipman first implemented this idea in a program called Fasta.[5] An extension of this idea is to compute, for each substring *s* in the input, all strings within a distance *d* of *s*, where *d* can be defined by any of various distance matrices. A system called Blast uses this approach.[6] These two programs have had tremendous success and are used thousands of times or more per day in labs and companies around the world. (For a more general introduction to string-matching algorithms in computational biology, see the text by Dan Gusfield.[7])

Computational gene finding

The advantage of sequence homology searches is that scientists can very quickly identify the likely function of previously uncharacterized genes. The disadvantage is that many newly sequenced genes have no homologs; that is, no similar gene has ever before been sequenced. In the 20 microbial genomes thus far sequenced, each containing thousands of genes, the number of unknown genes reported has ranged from 30% to upwards of 50%. For these unknown genes, sequence homology searches are of little immediate use. Researchers must instead rely on gene-finding programs to identify the genes initially, and then wait for follow-up research to characterize their functions.

Gene-finding programs are essentially machine-learning programs that must be trained from existing gene databases. They represent one of the more remarkable success stories of computational biology; researchers have used them to discover tens of thousands of new genes. Of these, only a tiny fraction have been explored further, but they provide the basis for an explosion in biology and biotechnology research.

Biological cells fall into two major classes: prokaryotic and eukaryotic. Each requires a different type of gene-finding program. Prokaryotes are single-celled organisms that include all bacteria plus another class of organisms, archaea, which are bacteria-like organisms that often live in extreme environments (for example, extremely high temperatures or high acidity). Prokaryotes have very small genomes, ranging from 0.5 to 10×10^6 bases in length, that typically have a coding density of about 90%—that is, about 90% of the DNA sequence consists of genes, with just a little bit of intergenic DNA connecting the genes together.

Eukaryotes, on the other hand, include humans, mammals, and all multicellular organisms, as well as many single-celled organisms such as yeast and various parasites. They have much larger genomes, ranging from a few tens of millions of bases to well over 10^{10} bases in length. (The human genome is approximately 3.3×10^9 base pairs.) The coding density of eukaryotes is far lower, ranging from a high of 50% for some simple parasites to only 1% to 3% for humans. The reduced gene density makes gene discovery far more difficult.

The computational gene-finding community has developed many approaches to meet this challenge, including hidden Markov models, Markov chains, decision trees, neural networks, and rule-based systems.[2]

Prokaryotic gene finding. As an example, let us consider Glimmer, a gene finder that is widely used in the prokaryotic genomics community. Using a computational technique called interpolated Markov models (IMMs), Glimmer can find approximately 98% of all the genes in a bacterial genome. The system operates *de novo*, requiring no other input than the genome sequence. This ability is crucial because techniques for large-scale genomic sequencing give no hints as to where the genes are. Because bacteria have a high coding density, the system can assume that genes are almost everywhere, and the problem in most cases is deciding the position of the codon (three-base combination) that represents the start of translation. Recall that DNA is translated three bases at a time into proteins; thus, a translation can start in position i, $i + 1$, or $i + 2$, yielding three completely different protein sequences. Translation can, with equal likelihood, use either strand of DNA, giving three additional "reading frames" on the opposite strand going in the other direction. Much of the gene-finding process involves deciding which of these six reading frames is the true coding sequence. Gene finders must also consider that a given region might contain no genes at all.

Glimmer's approach is to build a Markov chain model describing the probabilities of each of the four bases after many different short prefixes, called *contexts*. A zero-order model needs nothing more than the four bases' prior probabilities; order-k models require the computation of 4^{k+1} probabilities. The longer the model, the better it should be at modeling any sequence data, as long as sufficient training data is available. For example, a third-order model should outperform a second-order one, even if the second-order model is correct (for example, if the data was generated by a second-order model). Obviously, if the higher-order model is correct, it will work better. If the lower-order model is correct, then the higher-order model will be identical to the lower one; so even in this case the higher-order model will perform as well.

This is easy to see: suppose $P(a_i \mid b_{i-1}, c_{i-2})$ is the probability of a following the two characters cb, and $P(a_i \mid b_{i-1}, c_{i-2}, d_{i-3})$ is the probability of a following dcb. If the second-order model is correct, then $P(a_i \mid b_{i-1}, c_{i-2}) = P(a_i \mid b_{i-1}, c_{i-2}, d_{i-3})$ because the third previous position is irrelevant. Thus, the two models behave the same. The only time a problem occurs is when there is insufficient data to estimate the higher-order model. In this case, the higher-order model may indeed be inferior, because the probabilities it contains are inaccurate.

With 4^{k+1} probabilities necessary to train a Markov model for DNA sequences, the amount of sequence available critically affects

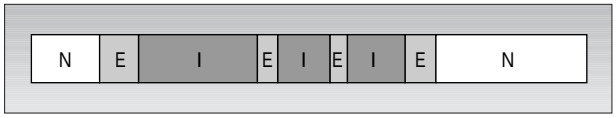

Figure 2. The structure of a typical gene in the human genome. The introns, labeled I and shown in dark gray, are spliced out and discarded during the protein-making process. The exons, labeled E and shown in light grey, are then concatenated and translated three bases at a time into a protein sequence. On either end of the gene is a noncoding (intergenic) sequence, labeled N.

the length of the longest model that can be accurately built. To create training data for a microbial genome, Glimmer uses a simple, highly effective procedure. First, Glimmer scans the genome and extracts long open reading frames (orfs). An orf is simply a run of codons that contains no stop codons. Because three of the 64 codons are stops, any sufficiently long orf has a very high probability of being a gene. Glimmer's training procedure simply extracts long orfs that do not overlap long orfs in other reading frames, and this often identifies as many as half of a genome's genes. This method generates more than enough data to serve as a training set for a bacterial genome, even for relatively small genomes.

A Markov model's optimal length thus depends on the genome's length. To avoid using a different Markov model for each genome, my colleagues and I implemented an approach that builds all models up to a certain length, and then interpolates among them. The advantage of this approach is that, for contexts where sufficient data is available, Glimmer uses the longer Markov model. For contexts where data is sparse (combinations of bases that are rare), Glimmer falls back on a shorter model that has more data behind it. The idea for IMMs originated in the literature on language and speech processing;[8] part of the novelty of Glimmer is its application of this idea to biological sequence data.

Glimmer's central computation is the probability $P(S|M)$ that the model M generated a DNA sequence S. This probability is computed by multiplying IMM(S) over all positions in S. The kth-order interpolated Markov model score, $\mathrm{IMM}_k(S_x)$, is computed recursively as

$$\begin{aligned}&\mathrm{IMM}_k(S_x)\\&= \lambda_k(S_{x-1}) \cdot P_k(S_x)\\&\quad + (1-\lambda_k(S_{x-1})) \cdot \mathrm{IMM}_{k-1}(S_x)\end{aligned}$$

where $\lambda_k(S_{x-1})$ is a numeric weight associated with the k-base context ending at position $x - 1$ in S, and $P_k(S_x)$ is the estimate obtained from the training data of the probability of the base located at x in the kth-order model. Thus the eighth-order IMM score of an oligomer (a subsequence of DNA) is a linear combination of the predictions made by the eighth and lesser-order models down to the zero-order model.[9]

In addition to the IMM, Glimmer contains rules for resolving overlaps (genes usually do not overlap in natural DNA sequences) and otherwise adjusting the predicted genes' boundaries. We have tested the system's performance on numerous bacterial genomes, six of which are shown in Table 1.

The organisms listed in the table include five bacteria and one archaeon (*M. jannaschii*), all of which have been completely sequenced. (Visit www.tigr.org/tdb/mdb/mdb.html for a complete, up-to-date list of genome sequences completed and in progress.) The order given is the order in which sequencing was completed, beginning with *H. influenzae* and *M. genitalium* in 1995. Column 3 gives the total number of genes for each organism. These are the numbers currently annotated in public databases; some of these numbers will likely change in the next few years, as more biological studies on these organisms are conducted.

The fourth column in the table gives the number of genes that Glimmer finds using the simple, fully automated training procedure sketched above. As the table makes apparent, the accuracy is very high, ranging from 97.4% to 99.7%. The final column lists the number of genes found by the system that are not contained in the official annotation. I labeled this column "Additional (false?) genes" intentionally; the current state of knowledge about these organisms does not let us state definitively how many of these extra gene predictions are false and how many might one day be discovered to be real.

Eukaryotic gene finding. The problem of finding genes in the much larger genomes of eukaryotes is far more difficult, for two main reasons. First is these genomes' low coding density, which can be as low as 1% to 3% of the total sequence. Second is the presence of subsequences of RNA called *introns*.

Eukaryotic genes are transcribed and translated, just as those of bacteria are. However, there is an additional step that bacteria do not go through. Eukaryotic cells recognize introns, which they cut out and discard. The remaining pieces (called *exons*) are concatenated, and the translation of this much shorter RNA product comprises the protein.

The reason introns are present in genes is still unknown. In human DNA, introns are not only very common; they are also very long, typically constituting 80% of the original RNA sequence. Figure 2 shows a schematic of what a gene looks like in the genome of humans and other multicelled organisms.

A typical human gene has four or five long introns, and the exons that encode the protein are quite short, usually in the range of 100 to 250 base pairs (bp) in length. Many exons are considerably shorter, some fewer than 10 bp. To date, the experimental evidence describing the biological system that recognizes exons' and introns' boundaries is still incomplete. There are also no foolproof computational methods for recognizing these boundaries. Intergenic regions containing no genes can extend for hundreds of thousands of bases. This makes data modeling much harder: with exons being so short, most statistical methods are inadequate for distinguishing an exon from any other piece of DNA.

Hidden Markov models. Among the most successful gene finders for human DNA are those that use hidden Markov models (HMMs). Several have emerged. One of the best current systems is Genscan,[10] which contains a semi-Markov HMM. Genscan succeeds through sophisticated modeling and the incorporation of many different aspects of biology. One of the most critical elements is its highly accurate method for identifying splice sites, the locations in the DNA where exons and introns meet. Identification of signals such as splice sites lets Genscan compensate for the unavailability of statistics that clearly characterize a short exon. The splice-site recognition algorithm is a probabilistic decision tree that uses correlations between different base positions to split the data, and then scores sites using a Markov chain at the tree's leaf nodes.[2]

Other gene-finding systems have also proven effective for human gene finding. Examples include the neural-net-based Grail system (developed by Edward Uberbacher's group[2]) and the hidden Markov model system HMMgene (developed by Anders Krogh, who has also written a tutorial on the use of HMMs to model DNA[2]) These are just a few of the many examples of gene-finding systems. Each gene finder has its strengths and weaknesses, and some systems seem to perform better on specific organisms.

Benchmark data. A sense of the accuracy (and difficulty) of computational gene finding in eukaryotic data can best be obtained by considering some results on a standard benchmark data set. Moises Burset and Roderic Guigo carefully collected 570 genes from a range of different vertebrate organisms, including humans.[11] Each gene was carefully checked to make sure that solid laboratory evidence supported it and that all exons and introns were present in the data. (Many gene sequences in the public databases are only partial.) The average gene in this set had four to five exons and covered about 5,000 bp of genomic sequence; 20% of these base pairs were exons, and the rest were either introns or intergenic sequence.

Burset and Guigo then ran eight different gene finders on these genes. On average, most gene finders performed well, but their accuracy was much lower than the standard for bacterial gene finding, thus reflecting the greater difficulty of the problem. Genscan had the best performance, correctly identifying 78% of the exons, with a specificity of 81% (that is, 81% of its exon predictions were correct). The second best gene finder, Genie (also an HMM system), had a sensitivity of 69% and a specificity of 70% on the same data.[12] Both systems exhibited far higher accuracy when measured as the percentage of nucleotides correctly recognized as coding (part of exons) versus noncoding (part of introns or intergenic), but this was mainly a consequence of the fact that more than 80% of the nucleotides in the data were noncoding.

Eukaryotic gene finding continues to be an active and important area of research, and with the completion of the human genome expected in the year 2002, the field urgently needs more research into algorithms with greater accuracy. When the human genome is first decoded, much of its annotation will likely be either inaccurate or incomplete, and many years will pass before all human genes can be identified with confidence.

COMPUTATIONAL EFFORTS HAVE attempted to characterize many other patterns that occur in DNA sequences. Such sequence signals are critical to the processes of regulating how much of each gene is produced. As a simple example, consider that different human cell types, such as skin cells and heart cells, have dramatically different appearances and functions, yet both sets of cells contain an identical copy of the DNA sequence and the same genes. This means they encode identical proteins. By producing different mixtures of these proteins, a cell can exhibit an enormous range of functions. The regulatory patterns that govern these mixtures are typically regions of DNA in close proximity to the genes to which certain proteins bind. The cell's ability to recognize specific combinations of bases in DNA sequences provides compelling evidence that the patterns exist, but identifying the patterns computationally is still a challenge.

The 21st century will bring us a revolution in our understanding of human health, genetics, and life itself. This revolution will be fueled by the fundamental genetic instructions contained in the genomes of every living organism. As these genomes are revealed through large-scale sequencing, we will gradually unravel the code for the many complex interactions between protein and DNA that control how a living organism functions. Computational biologists are working to create techniques that will help us understand some of these interactions, but the greatest challenges and discoveries are yet to come. ■

Acknowledgments

Thanks to Owen White for his helpful comments on the manuscript. Also, I was supported in part by NIH Grants K01-HG00022-1 and R01-LM06845-01 and by NSF Grants IRI-9530462 and IIS-9902923.

References

1. R.D. Fleischmann et al., "Whole-Genome Random Sequencing and Assembly of *Haemophilus Influenzae Rd*," *Science*, Vol. 269, Aug. 1995, pp. 496–512.
2. S. Salzberg, D. Searls, and S. Kasif, eds., *Computational Methods in Molecular Biology, New Comprehensive Biochemistry*, Vol. 32, Elsevier Science B.V., Amsterdam, 1998.
3. S. Needleman and C. Wunsch, "A General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins," *J. Molecular Biology*, Vol. 48, 1970, pp. 443–453.
4. T. Smith and M. Waterman, "Identification of Common Molecular Subsequences," *J. Molecular Biology*, Vol. 147, 1981, pp. 195–197.
5. D.J. Lipman and W.R. Pearson, "Rapid and Sensitive Protein Similarity Searches," *Science*, Vol. 227, 1985, pp. 1435–1441.
6. S. Altschul et al., "Basic Local Alignment Search Tool," *J. Molecular Biology*, 1990, Vol. 215, pp. 403–410.
7. D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge Univ. Press, New York, 1997.
8. F. Jelinek and R.L. Mercer, "Interpolated Estimation of Markov Source Parameters from Sparse Data," *Pattern Recognition in Practice*, E.S. Gelsema and L.N. Kanal, eds., Elsevier/North Holland, New York, 1980, pp. 381–397.
9. S. Salzberg et al., "Microbial Gene Identification Using Interpolated Markov Models," *Nucleic Acids Research*, Vol. 26, No. 2, 1998, pp. 544–548.
10. C. Burge and S. Karlin, "Prediction of Complete Gene Structures in Human Genomic DNA," *J. Molecular Biology*, Vol. 268, 1997, pp. 78–94.
11. M. Burset and R. Guigo, "Evaluation of Gene Structure Prediction Programs," *Genomics*, Vol. 34, No. 3, 1996, pp. 353–367.
12. M. Reese et al., "Improved Splice Site Detection in Genie," *J. Computational Biology*, Vol. 4, No. 3, 1997, pp. 311–323.

Steven L. Salzberg is the director of Bioinformatics at the Institute for Genomic Research in Rockville, Maryland, and a research professor in the Computer Science Department at Johns Hopkins University. His research interests include computational biology (with a special emphasis on genomic research) and machine-learning algorithms, including Markov models, decision trees, and memory-based reasoning. He has authored or coauthored over 60 scientific papers and two books, and is cochair of the 1999 Computational Genomics Conference. He received his BA, MS, and MPhil in computer science from Yale University, and his PhD in computer science from Harvard University. Contact him at the Inst. of Genomic Research, 9712 Medical Center Dr., Rockville, MD 20850; salzberg@tigr.org; www.tigr.org/~salzberg.