A Random Walk Down the Genomes: DNA Evolution in Valis

The authors propose a new software system, Valis, that incorporates biological data and domain-specific knowledge and show how biologists can use it to model, analyze, and experiment with genomic evolutionary processes.

Salvatore Paxia
Archisman Rudra
Yi Zhou
Bud Mishra
New York University

While chemistry and physics are the substrate of biology, researchers now believe that a better understanding of biology will come through information-theoretic studies of genomes, providing new insights into DNA's role in governing metabolic and regulatory pathways. Consequently, the mathematical approaches derived from systems sciences—dynamical systems, control theory, game theory, information and decision theory, and mathematical logic—are playing increasingly important roles in biological research.

Understanding the evolutionary processes that act on these "codes of life"—including point mutation, recombination, gene conversion, replication slippage, DNA repair, translocation, imprinting, and horizontal transfer—requires the ability to analyze vast amounts of continually generated genomic data. The challenges, intrigues, and excitement that these genomic sequences have come to symbolize have catapulted the embryonic field of bioinformatics to the forefront.

Bioinformatics currently consists of a set of tools to "contig" genomic sequences and organize, annotate, and search sequence databases and generate computationally or statistically intriguing problems. However, researchers in this emerging discipline require more complex mechanisms to investigate the full ensemble of available biological facts.

VALIS

To meet this challenge, New York University's Bioinformatics Group is creating a computational environment—the vast active living intelligent system—designed to solve the immediate genomic and proteomic problems that the biological community currently faces but flexible enough to adapt to the maturing bioinformatics field. Inspired by Philip K. Dick's 1981 science fiction novel, Valis (http://bioinformatics.cat.nyu.edu/valis/) envisions a modern biology driven by large-scale processing of heterogeneous data from diverse sources as well as sophisticated algorithms to extract meaningful information and suggest new experiments, either to validate old data or resolve ambiguities.

Individual researchers have already written some of these algorithms, but the resulting tools usually depend on many specifics of internal data representation, different assumptions about the nature of the data, and idiosyncratic visualization and manipulation schemes. Current data sources range from GenBank genomic sequences to results of individual microarray experiments. Interfaces to these data sources vary widely as well, with a concomitant increase in complexity. Further, the present trend of ad hoc algorithm development leads to little code sharing.

Valis is a language-independent environment for prototyping bioinformatics applications that provides

- a set of libraries to read input data stored in relational databases or standard file formats,
- efficient implementations of algorithms useful to genomics, and
- numerous visualization tools.

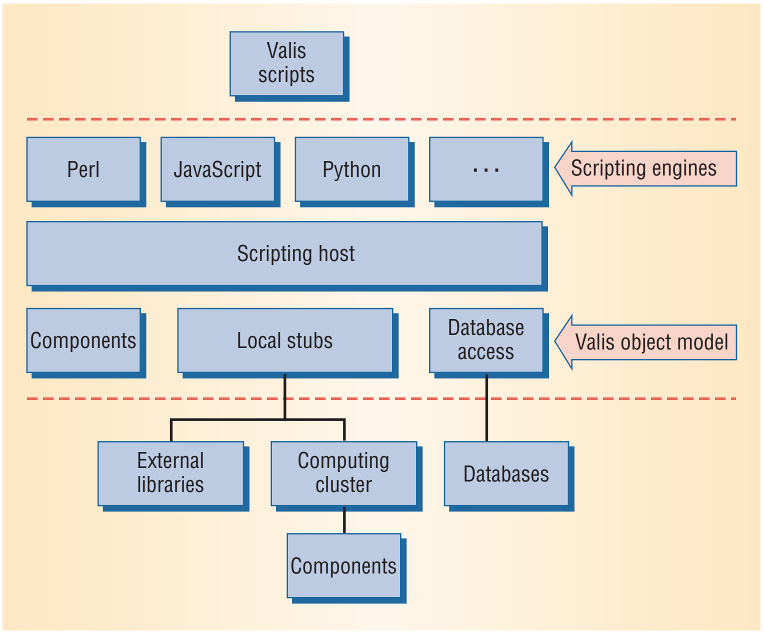

Figure 1. Valis architecture. User-written scripts manipulate objects, which Valis exports to scripting engines from the low-level libraries.

Instead of developing each tool anew, Valis defines low-level building blocks and uniform APIs invoked using high-level scripting languages. This flexibility lets biologists write simple scripts to perform fairly involved bioinformatic processing.

SYSTEM ARCHITECTURE

As Figure 1 shows, Valis consists of three basic layers. The bottom layer consists of data-access routines, libraries that provide efficient algorithms for data-analysis tasks, and visualization components. The libraries are written in multiple languages and can be executed on remote servers, thus requiring a middle layer to wrap the code into reusable components with a standard interface. This middle layer exposes the Valis object model to the supported scripting engines. The top layer, where higher-level scripts are interpreted, provides the Valis user interface and development environment.

Valis currently supports JavaScript, VBScript, Python, Perl, and SETL. The syntax of scripts written in these languages varies, but all scripts use the same class hierarchy. For example, upon learning that a Valis `Sequence` object has a method called `Input` that will read a file's sequence, users can subsequently take this same primitive from all five languages.

To deal with the proliferation of incompatible and proprietary file formats in the genomics community, Valis provides data input objects that can read sequences, maps, tables, annotations, microarray data, and so on from disparate sources. Once the data is loaded, Valis presents it uniformly to the computational layer.

Valis includes extensive computational facilities to process genomic data: numerical algorithms, string-processing routines, alignment tools, sequence and map assembly facilities, and statistical analysis algorithms. Although researchers can prototype a given algorithm in any supported scripting language, Valis provides efficient implementations of the basic building blocks for processing data in quasi-real time. For example, it's possible to extend the system without recompilation as well as dynamically load new native libraries into it.

Although designed for workstations, Valis can run computation-intensive processes on computer clusters. The current implementation runs on a Beowulf cluster, which we plan to enhance periodically. Once processing is completed, it's important to be able to quickly view the results. Valis thus provides numerous visualization tools that let users display sequences, maps, microarray data, tables, graphs, and annotations. Users can customize the widgets from the scripts.

ANALYZING DATA FROM DIVERSE SOURCES

Valis makes it possible to analyze genomic sequences in novel ways to uncover the footprints of evolutionary processes that have acted on DNA sequences. Our approach deviates significantly from the current practice of data mining for "interesting features" by using statistical algorithms built upon biological insights. The "Valis in Action: Analyzing a Human Chromosome" sidebar provides an example of how we use the system to analyze large-scale genomic structures.

Long-range correlation

We base our analysis on *fractional Brownian motion,* a non-stationary stochastic process (but with stationary increments) that displays statistical self-similarity—that is, it's the same as a scaled version of itself. fBm is a generalization of the classical Brownian motion process, characterized by a random variable $S(t)$ parameterized by time t, where $S(t)$ represents the cumulative effect of many small fluctuations over the time interval from 0 to t. One interesting property displayed by fBms is that for any positive scaling parameter r and time instances $t > u$, $[S(rt) - S(ru)] / r^H$ and $S(t) - S(u)$ are statistically indistinguishable. Intuitively, speeding up the time by a factor r has the effect of scaling the fBm process by a power H of the scaling factor—the fBm's Hurst exponent.

These mathematical properties render an fBm particularly useful as a model of genomic sequences. For example, using H we can look for subtle changes in *long-range correlations* (LRCs) in a genomic sequence and thus hypothesize about the effects of various evolutionary processes.

When H is equal to 0.5, an fBm reduces to the

Valis in Action: Analyzing a Human Chromosome

The following example, written in JavaScript, illustrates how we use Valis to analyze large-scale genomic structures—in this case, the DNA sequence of human chromosome 22, the first to be decoded by the international Human Genome Project.

We first load the chromosome's data set into the system from a `fasta` file. Then, we annotate the sequence with data from a GoldenPath mirror site (http://genome.ucsc.edu/). These predefined annotations, commonly accepted by the biological community, provide rudimentary meanings to various genomic regions that Valis will analyze further. Finally, we run a word frequency analysis algorithm to find the probability distribution of all words of length *k* within the chromosome.

The script starts by selecting the language and clearing the output window:

```
#language JSCRIPT
Valis.Clear();
```

Next, we create a Structured Query Language (SQL) data access object, and connect to one of our databases:

```
sql = Valis.CreateObject("Sql");
sql.Connect("DSN = mysql; UID = someuser;
PWD = somepwd");
```

This creates a DNA sequence object—`DNASeq`, a string of *A, T, C,* and *G*—called `seq` and inputs its data from a `fasta` file.

Most complex objects in Valis have a `Display` method. Invoking the `seq.Display()` method results in:

```
seq = Valis.CreateObject("DNASeq");
seq.Input("C:\\GoldenPath\\chr22.fasta");
seq.SelectSequence(1);
seq.Display();
```

We then run an SQL query on the interface object, which returns a Valis table. This is a flexible object, in which each column can have a different type—`string`, `integer`, `double`, and so on. The ExecSQL method automatically generates the table's columns and types and fills it up with the query results. Again, we can display the table with the corresponding `Display` method.

```
table = sql.ExecSQL("select name,strand,
cdsStart,cdsEnd from genscan
 where chrom = 'chr22'");
table.Display();
```

The last step involves creating a scrollable Bander widget. We use this widget to display a sequence along the *x*-axis and numerous bands along the *y*-axis. We then load the sequence in the widget at position zero and create some bands: `b1` and `b2` are Boolean bands, which will be true when the sequence is either *A* or *T*, or *G* or *C*; `b11` is a block band that will contain the results of the SQL query; and `freq` will obtain the word frequency of this particular sequence.

```
a = Valis.CreateObject("Bander");
a.LoadSequence(seq,0);

b1 = a.AddBand(1,"AT");
b2 = a.AddBand(1,"GC");
m=a.AddBand(1,"Masked");
b11 = a.AddBand(5,"GenScan");
freq = a.AddBand(4,"Freq");
```

Now we can perform the necessary computations to change band colors and sizes. `CharBand` will create a band for true values when Valis finds one of the designated characters in a genome sequence—for example, *AT* designates the pattern "*A* or *T*."

```
a.CharBand(b1,"AT");
a.SetColor(b1,RGB(100,0,0));  //Red

a.CharBand(b2,"GC");
a.SetColor(b2,RGB(0,100,0));  //Green

a.CharBand(m,"N");
//Either of the match fails
a.SetColor(b11,RGB(0,200,200));  //Cyan
```

The next step involves loading a block band with rows from a table. The parameters are the table containing the data to be accessed, the destination band, and the columns containing the starting position, ending position, strand, and description. Finally, we run an efficient word frequency analysis algorithm that will fill the frequency band with the occurrences of the substrings of length 15 starting at each position.

```
a.LoadBlocksFromTable(table,b11,2,3,1,0);
a.SetColor(freq,RGB(100,0,100));
a.SetSize(freq,200);
a.FindRepeats(freq,14);

a.Display();
```

As shown by this simple protocol—which lets users visually examine a DNA sequence for biological insights correlating annotated genes, repeats, and conserved regions with computer graphic richness, word frequency, and highly repeated words—Valis enables effortless exploration of long genomic sequences.

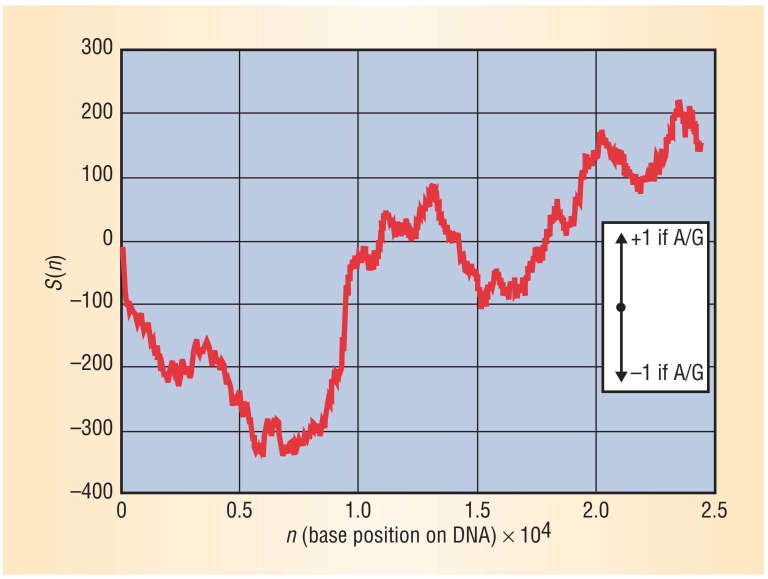

Figure 2. Sample DNA walk landscape. In this example, Valis uses the purine-pyrimidine rule to map a DNA sequence onto a DNA walk.

familiar Brownian motion—an independent random sequence—and there is no long-range correlation. When *H* is not equal to 0.5, the successive increments are Gaussian but because of the correlation are not independent of one another.

When *H* is greater than 0.5, the correlation between increments is positive, and the values tend to cluster together, a phenomenon called *persistence*. Likewise, when *H* is less than 0.5, the process is *antipersistent*—increments are negatively correlated and the values do not cluster together.

For the purpose of modeling a DNA sequence as an fBm, we think of the sequence as a string composed of four letters: A (adenine), T (thymine), C (cytosine), G (guanine). Not all parts of the genomic code are for proteins, however, and it seems likely that multiple processes operate on various DNA regions. Because different characteristic length scales are associated with these processes, we expect LRC levels to vary. Indeed, one group of researchers has found a statistical association between the letters at long distances on the same string, with the LRC level depending on the genomic region.[1]

DNA walks

The class of fBm models that results from these analyses is a *DNA walk*, which is similar but not identical to a random walk model. For example, consider the nucleotide base pairs A/G (purine) and C/T (pyrimidine). Although we could synthesize a DNA sequence by randomly and independently choosing either A or G, or C or T, and incorporating the chosen base at the end of a growing string, such a random walk is not likely to have produced genomic sequences occurring in nature.

As Figure 2 shows, every DNA sequence can form a DNA walk depending on the rule used to binarize the sequence. In this case, using the purine-pyrimidine rule, Valis denotes every encounter with A/G as $X_i = +1$ and every encounter with C/T as $X_i = -1$. The trajectory of the partial sum $S_n = \Sigma_{i=1}^{n} X_i$ along a DNA sequence gives a DNA walk landscape.

The LRC level in various DNA walks can be measured by their Hurst exponents. Valis's modular architecture makes it possible to test these models easily without worrying about database access and data representation details. We implemented many algorithms in Valis to estimate *H*.

We have used Valis to analyze the genomes of bacteria, invertebrates, and vertebrates. All the DNA sequences and annotations are available for downloading from the National Center for Biotechnology Information's GenBank database (http://www.ncbi.nlm.nih.gov/Genbank/).

First, we separated each organism's genomic sequence into two subsequences: one from the coding sequences, the other from the noncoding sequences—introns (within genes) and intergenic (between genes) regions. For each of these separated sequences, we then estimated the Hurst exponent using various methods such as rescaled range analysis[2] and detrended fluctuation analysis.[3] Finally, we compared *H* values in the DNA walks with that of a random sequence generated from Brownian motion.

Table 1 shows the estimated Hurst exponents for the entire genomic sequence of a bacterium (*Escherichia coli K12*), a unicellular eukaryote (*Saccharomyces cerevisiae,* baking yeast), an invertebrate (*Drosophila melanogaster,* fruit fly), and a vertebrate (*Homo sapiens*). Note that *H* values tend to be higher in the noncoding regions than in the coding ones. Further, the estimated *H* values tend to be higher for more complex organisms. The DNA walk down the coding region sequences thus behaves more like a Brownian motion, while the noncoding regions exhibit persistent LRC.

IN SILICO EVOLUTION

What accounts for the variation in LRC levels between different regions? A natural explanation is

Table1. Estimated Hurst exponents in different organisms and regions of genomic sequences.

Organism	*Escherichia coli K12*		*Saccharomyces cerevisiae*		*Drosophila melanogaster*	*Homo sapiens*
Region	Coding	Noncoding	Coding	Noncoding	Coding	Coding
***H* value**	0.5199	0.5676	0.5644	0.6522	0.5843	0.6060

that this correlation is attributable to the slow accumulation of random mutations in the genomic sequence. Also, various repair mechanisms likely operate in the coding regions to lessen the mutations' impact. To test how these and other processes influence LRC, we can model their dynamics to determine whether in silico—computer-simulated—genomes have the same statistical structures as in vivo genomes at every scale.

Modeling genomic events

Because this approach requires far more memory than current computer systems can realistically provide, most population evolution studies use compact genomic representations instead of actual sequences. However, calculating quantities such as *H* demands maintaining the sequences in all intermediate populations.

As Figure 3 shows, Valis solves this problem by formalizing the classes of possible transformations that can act on genomes as algebraic datatypes. Each line represents a DNA molecule. Fragments marked by the same number/letter are homologous—for example, 2 and 2′. These representations track the transformations that accumulate in different sequences. This leads naturally to a data structure for sequences that maximizes data sharing while automatically maintaining data consistency.

DNA replication slippage can result in deletions and duplications of the daughter strand (pink lines). During recombination, when the crossing over occurs unequally between two homologous DNA regions (pink 2 → brown 2′), one homologous region may be deleted and another duplicated. Translocations are usually caused by double-breakages in the DNA. If the broken ends religate to each other (no homology is required), as Figure 3 shows, a reciprocal transposition occurs that changes the positions of some DNA fragments. During transposition, a transposon (ϕ) can cut itself out of the DNA sequence and cause a deletion, or it can replicate itself (ϕ′) and insert into another DNA sequence and cause duplication.

Genome grammar

In Valis, direct concatenation of sequences still remains computationally expensive and is implemented via logical operators that "remember" the concatenation of two sequences. Such operators are available to a higher-level garbage-collector-like subsystem that hides the implementation details. This library frees biologists from choosing particular compact representations of sequences suitable for the evolutionary simulation in which they're interested.

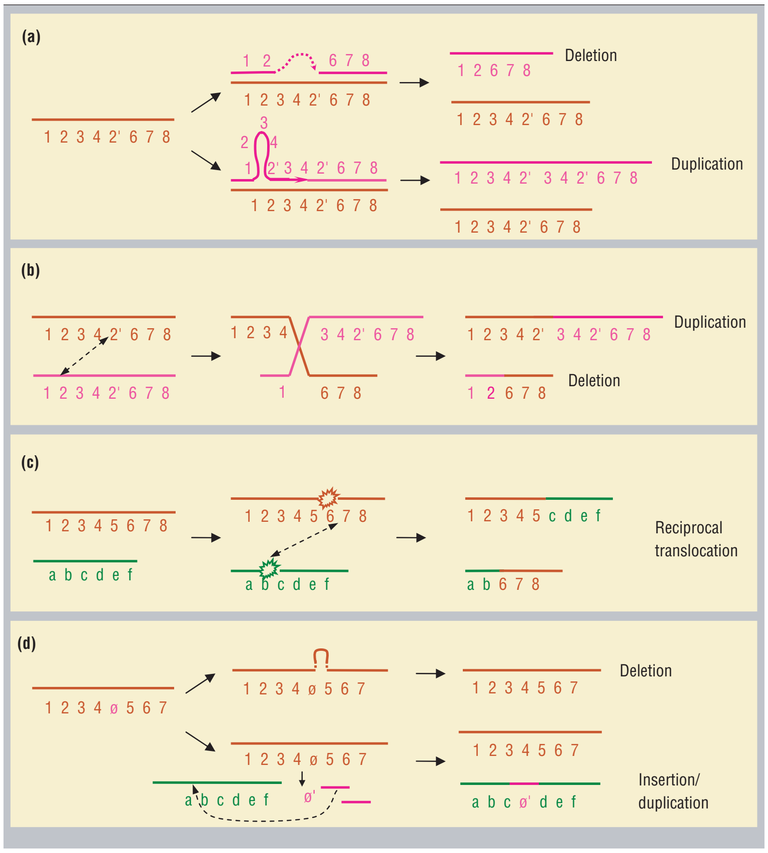

Figure 3. Schematics of selected genomic evolution events: (a) replication slippage, (b) recombination, (c) translocation, and (d) transposition. Each line represents a DNA molecule. Fragments marked by the same number/letter are homologous.

We can extend such a scheme to represent the probabilistic processes of generating a genomic sequence. This leads to tools for synthesizing artificial sequences with statistical properties close to those of natural DNA sequences. These tools are structured around a *genome grammar,* a set of libraries for creating and manipulating stochastic grammars with primitives for many kinds of mathematical probability distributions.

Using stochastic grammars to generate genomic sequences is not new,[4,5] but integrating the grammatical structure and memory manager results in a novel, efficient runtime system that facilitates quick evolutionary experiments in silico.

Generic modeling process

Other Valis modules can model the vast number of cellular mechanisms that an action on the genomic sequence triggers. A process that models biological machinery generically describes entities such as a protein complex or an enzyme, walking along the DNA and modifying it under local scrutiny:

- The machinery looks for a certain initiation recognition site on the DNA sequence—as determined by the DNA's constituent bases, physical or chemical properties, and so on. The initiation site also depends on the properties associated with the machinery itself.
- After recognizing the site, the machinery starts to walk along the DNA sequence, changing its

own state if necessary and the DNA sequence it passes over.

- At some point—typically when the terminating site is encountered—the process halts.

This process can model how a transcription factor associates itself to a regulatory sequence. Depending on the distribution of the initiation sites and the relative distributions of the corresponding termination sites, we can associate an effective radius as well as a probability distribution with each of the events that this machinery can mediate. We have implemented just such a linear automaton as a general-purpose machine.

Valis modules can model the vast number of mechanisms that an action on the genomic sequence can trigger.

Debugging in silico experiments

Biologists must be able to use real organisms to statistically recreate in silico evolution. Various mutant organisms that differ from their related wild type exhibit significant changes in the occurrence probabilities of genomic evolution events. These organisms are natural candidates for "debugging" our in silico experiments. One candidate is pol3-t, a temperature-sensitive mutant in yeast DNA polymerase that, under nonpermissive temperatures, increases the rate of small deletions by up to 100 fold.[6]

Another example is the DNA mismatch repair system in eukaryotes. This system contains two subsystems correcting the mismatched DNA fragments of different sizes—one favors small mistakes (one to two bases), the other favors bigger deletions or insertions (more than five bases).[7] The mutations in each of the two subsystems diminish the cell's ability to correct DNA sequence mismatches and significantly increase the rate of sequence changes.

Researchers have traced more than 10,000 generations of *E. coli* in laboratory evolution.[8] Similar experiments can explore eukaryotic sequence evolution in budding yeast. We can track the changes in these mutants' genomic sequences and observe how they differ from the wild types. Tracing these changes helps to gather information about how these processes shape genomic sequences. In silico evolution will ultimately help us devise the most efficient in vivo experiments, which in turn will validate and further improve our in silico model. ■

Acknowledgments

We thank Mike Wigler, Elizabeth Thomas, Misha Gromov, and Ale Carbone for offering many exciting ideas, useful contributions, and ways to think about genomes. Thomas Anantharaman, Dave Schwartz, Izzy Edelman, Jack Schwartz, Eric Siggia, Chris Wiggins, Gloria Coruzzi, Dan Tranchina, Suse Broyde, Phil Benfey, Harel Weinstein, Mike Seul, Bob Desnick, and many others indirectly contributed to our efforts. Valis is supported by grants from the National Institutes of Health; the NYU University Research Challenge Fund; the US Department of Energy; the New York State Office of Science, Technology, & Academic Research; DARPA; and an HHMI Biomedical Support Research Grant.

References

1. C-K. Peng et al., "Long-Range Correlation in Nucleotide Sequences," *Nature,* 12 Mar. 1992, pp. 168-170.
2. E.E. Peter, *Fractal Market Analysis: Applying Chaos Theory to Investment and Economics,* John Wiley & Sons, New York, 1994.
3. C-K. Peng et al., "Mosaic Organization of DNA Nucleotides," *Physical Rev. E,* Feb. 1994, pp. 1685-1689.
4. G. Myers, "A Dataset Generator for Whole Genome Shotgun Sequencing." *Proc. 7th Int'l Conf. Intelligent Systems for Molecular Biology* (ISMB 99), AAAI Press, Menlo Park, Calif., 1999, pp. 202-210.
5. D.B. Searle, "The Computational Linguistics of Biological Sequences," *Artificial Intelligence and Molecular Biology,* L. Hunter, ed., MIT Press, Cambridge, Mass., 1993.
6. H.T. Tran et al., "Replication Slippage between Distant Short Repeats in *Saccharomyces cerevisiae* Depends on the Direction of Replication and the RAD50 and RAD52 Genes," *Molecular and Cellular Biology,* vol. 15, no. 10, 1995, pp. 5607-5617.
7. E.A. Sia et al., "Microsatellite Instability in Yeast: Dependence on Repeat Unit Size and DNA Mismatch Repair Genes," *Molecular and Cellular Biology,* vol. 17, no. 5, 1997, pp. 2851-2858.
8. R.E. Lenski and M. Travisano, "Dynamics of Adaptation and Diversification: a 10,000-Generation Experiment with Bacterial Populations," *Proc. Nat'l Academy of Sciences of the United States of America,* 19 July 1994, pp. 6808-6814.

***Salvatore Paxia** is a PhD candidate in the Department of Computer Science & Mathematics at New York University's Courant Institute of Mathematical Sciences, a research scientist with the NYU*

Bioinformatics Group and a cofounder of Ondotek, an information technology company in New York City. His research interests include bioinformatics environments, very high level programming languages, free-format databases, autostereo 3D display technology, and hardware support for distributed computing. Contact him at paxia@cs.nyu.edu.

***Archisman Rudra** is a research scientist with the NYU Bioinformatics Group. His research has focused on statistical, algorithmic, and learning theories applied to biological and computer vision problems. He received a PhD in computer science from NYU's Courant Institute of Mathematical Sciences. Contact him at archi@cs.nyu.edu.*

***Yi Zhou** is a PhD candidate in New York University's Department of Biology and a member of the NYU Bioinformatics Group. Her research focuses on modeling DNA evolution events, including sequence analysis, biological hypothesis forming, mathematic modeling, computer simulation, and model testing. She received an MS in biology from NYU. Contact her at joey@cs.nyu.edu.*

***Bud Mishra** is a professor of computer science and mathematics at NYU's Courant Institute of Mathematical Sciences, a professor at Cold Spring Harbor Laboratory's Watson School of Biological Sciences, a codirector of NYU's Center for Comparative Functional Genomics, and a cofounder of OpGen, a biotech company in Wisconsin and New York. His current research focuses on modeling cellular and genomic evolutionary processes; single-molecule-based technology for mapping, sequencing, karyotyping, and haplotyping; microarray-based technology for correspondence mapping; and gene-expression analysis with applications to cancer study. Mishra received a PhD in computer science from Carnegie Mellon University. He is a member of the ACM, the AMS, and the AAAS and a senior member of the IEEE. Contact him at mishra@nyu.edu.*