Computers Are from Mars, Organisms Are from Venus

Junhyong Kim
Yale University

Combining cold silicon and hot protoplasm may constitute a marriage of opposites, but this union could produce genetics research prodigies.

In recent years, biology and computer science have undergone an increasingly intense interdisciplinary merger, one that has attracted media attention and eager investors. The resulting media buzz arises from the strenuous struggle as these two disciplines come together. Is the sound we hear the agonizing grind of unmatched gears, or the smooth hum of an accelerating machine?

I suspect the latter. Biology and computer science share a natural affinity. Erwin Schrödinger envisioned life as an aperiodic crystal, observing that the organizing structure of life is neither completely regular, like a pure crystal, nor completely chaotic and without structure, like dust in the wind. Perhaps this is why biological information has never satisfactorily yielded to classical mathematical analysis. A simple look out the window, however, shows the great abundance of structure in biological objects, from the fractals found in the branches of an oak tree to the symmetries of DNA's double helix.

Machine computations combine elegant algorithms with brute-force calculations—which seems a reasonable approach to this aperiodic structure. Likewise, computing seeks to create a machine that can flexibly solve diverse problems. In nature, such plastic problem solving resides uniquely in the domain of organic matter. Whether through historical evolution or individual behavior, organisms always adaptively solve the problems their environment poses. Thus, examining how organisms solve problems can lead to new computation and algorithm-development approaches.

BIOLOGY MEETS COMPUTER SCIENCE

Biology is the youngest of the natural sciences. When its collected information reaches a critical density, a natural science progresses from information gathering to information processing. This latter activity has become the dominant one in more mature sciences like physics, in which theoretical abstractions and predictions play a primary role because new information is scarce.

Until recently, the major activity in biology has been gathering new information in the lab and field. The growth of biological information in the past five years, especially at the molecular level, has been astonishing. The growth curve of the information stored in GenBank (http://www.ncbi.nlm.nih.gov/), the primary molecular-biology information database, follows an exponen-

0018-9162/02/$17.00 © 2002 IEEE

Currently, the two most successful uses of computers in biology are comparative sequence analysis and in silico cloning.

tial curve that closely mimics Moore's law—doubling every 18 months or so.

When I started doing laboratory work about 10 years ago, it took us around five days to obtain 200 base pairs of DNA sequence data. Two years ago, biotechnology corporation Celera produced the fruit fly genome's 170 million or so bases in a few months—a feat that researchers apparently can now perform in a few weeks. Current estimates indicate that the public-sector sequencing capacity devoted to just the Human Genome Project has reached about 28 million bases a month, with private capacity climbing to several times that figure.

This volume of information overwhelms us. No single person, or even a large group, can ever hope to make sense of this information, especially at the rate researchers produce it. By helping researchers process this vast collection of data, computer science can assist in dispersing this information storm.

Currently, the two most successful uses of computers in biology are comparative sequence analysis and *in silico cloning*—the process of using a computer search of existing databases to clone a gene.

Comparative sequence analysis

When researchers isolate new molecular sequence data in the laboratory, they want to know everything about that sequence. A first step is to see if other researchers have already studied any molecular sequences similar to this sequence. Two mechanisms generate similarity of sequences. Like human relatives, biomolecular sequences related by evolutionary descent share sequence similarities. The structural requirement of performing a particular function also constrains two molecules with similar function to resemble each other. Therefore, researchers can learn a great deal about a biomolecular sequence by comparing extrapolated information to similar, already well-studied sequences.

Probably the most widely used computational tool in biology, BLAST—Basic Local Alignment Search Tool—searches databases like GenBank for all sequences similar to a target sequence. These days, when researchers isolate a new molecular sequence, the first thing they usually do is run a BLAST search against existing databases.

In silico cloning

Finding the information to perform successful cloning—an important activity in biology—is comparable to searching for a particular sentence in a book that resides somewhere in a huge library. Finding a sentence might encompass searching by semantics, such as "find a sentence that expresses the angst of a young prince"; or by syntax, such as "find a sentence that starts with a preposition, present subjective verb," and so forth; or by a pattern, such as "find a sentence that starts with 'To be or not to be.'" Likewise, in an experimental setting, we may want to clone a gene by its phenotype, say, olfaction; by its structure, say, a G protein-coupled receptor; or by a pattern fragment, say, the DNA pattern "ACCAGTC."

Performing these searches in the laboratory resembles physically going to the library to look for a book that contains the information we seek: Both activities present many logistical problems. Genome projects are somewhat like putting all the library's books on a CD-ROM: Although doing so alleviates the physical access problems, other problems remain. Storing the genome data is like having all the books on a CD without help guides such as titles, annotations, and cross-references. Fulfilling the request to "find a sentence that expresses the angst of a young prince" would require much reading and interpretation. However, conducting a pattern search would simplify the task by helping to sift through the information rapidly. Similarly, using in silico cloning to search existing databases is a significant benefit of genome projects. Using biological semantics to search genome databases would be a major achievement.

Computational approaches that use the computer to guide the more expensive and time-consuming wet-lab experiments can help solve long-standing laboratory problems. In a collaborative project with Drosophila biologist John Carlson, my colleagues and I successfully used a quasiperiodic feature classifier (QFC), a new computer algorithm, to solve a 15-year-old problem of isolating the fruit fly's olfactory genes.[1] Through identifying G protein-coupled receptors in the Drosophila genome, the computer program narrowed the possible candidate genes sufficiently to make the experimental work manageable.

Synthesizing information into knowledge

These days, biologists use computers routinely to assist with many activities, including

- biomolecular sequence alignment,
- assembly of DNA pieces,
- multivariate analysis of large-scale gene expressions, and
- metabolic pathway analysis.

Just as conducting normal business activities has become nearly impossible without informatics sup-

port, so too has biological research come increasingly to depend on informatics. What other important challenges confront computational biology?

Historically, and somewhat inherently, biology is a diverse field that collects information from many distributed sources. For example, 50 different laboratories around the world might study a particular gene. Given these heterogeneous information sources, collecting and integrating them into a coherent information set presents a major problem.

Given the pace of data production, the fundamental task of curating and integrating distributed databases is critically important. The problems encountered range from relatively simple to almost philosophical. Simple problems include different labs giving the same object 10 different names. Difficult problems include the seemingly trivial challenge of how to define a gene—a practical rather than philosophical problem for a database specialist. Currently, most major databases, such as GenBank and Swiss-PROT (http://www.ebi.ac.uk/swissprot/index.html), operate partially by human curation and partially by various automated data-sharing schemes. Issues of data quality, interoperation, and integration of dispersed information sources remain problematic.

Computational biology researchers use existing information to extrapolate knowledge about novel biomolecules. Because genome projects generate raw data without giving them biological meaning, annotating this raw data with all the pieces of information that might be biologically relevant becomes important. Useful information includes whether a stretch of DNA contains an amino acid coding sequence, transposons, or a regulatory sequence and, if an amino acid is coded, what its putative function is, and so on.

One detailed annotation addressed the 3,000,000 bases that surround the *Drosophila melanogaster* ADH sequence (http://www.flybase.org/). When researchers completed the Drosophila sequence, they held an annotation jamboree to provide as much biological context as possible for the raw sequences.[2] The researchers used a variety of computer tools combined with human interpretation to carry out much of this annotation. However, given the rate at which researchers now generate DNA sequence information, automatically annotating the raw data presents a computational challenge, and careful human analysis is becoming increasingly difficult. The tools necessary for addressing this problem include gene prediction, gene classification, comparative genomics, and evolutionary modeling.

Although researchers increasingly synthesize biological information into general theories, our knowledge remains scattered and complex. For example, we have a great deal of detailed information about the molecular events governing the early development of Drosophila. However, this extremely complex knowledge takes the form of statements like "gene A and B interact to positively induce gene C, but under the presence of gene D, the positive regulation is modulated by." We usually do not use an a priori theory to gather coordinated data; instead, hundreds of different research groups make independent observations hoping to synthesize that theory. These hundreds of independent observations appear in thousands of articles that use scores of variations in terminology, methodology, and so forth.

Although researchers increasingly synthesize biological information into general theories, our knowledge remains scattered and complex.

Worse, all this data applies to only a tiny segment of the field: the biology of fruit fly development. To cope with the information explosion represented by the tens of thousands of research articles that appear in print each year, we need a system that performs automatic knowledge extraction: a computer program that scans all these articles, classifies them, and produces synthetic new information. While daunting, such a project would be useful not only in the biological context but also in everyday life. No doubt this is why automatic text retrieval and knowledge extraction have become an important research area in computer science (http://www.cs.cmu.edu/cald/research.html), which has just begun to provide new tools for biological research.[3]

COMPUTATIONAL BIOLOGY'S HOLY GRAIL

When asked what the Holy Grail of computational biology is, most researchers would answer that it is either sequence-structure-function prediction or computing the genotype-phenotype map.

Predicting molecular structure

Sequence-structure-function prediction refers to the idea that, given a molecule's sequence identity, we would like to predict its three-dimensional structure and, from that structure, infer its molecular function. In the past, researchers were widely successful in deducing the universal genetic code, which consists of a relational map from DNA sequences to amino acid sequences, revealing the amino acid identity once we know the DNA sequence identity. This code's universality and elegant combinatorial structure are a remarkable fact of nature. In addition, our knowledge of the code has extremely practical consequences.

The relationship between amino acid strings and the 3D structure of proteins is thought of as the second genetic code.

Identifying the amino acid sequence of a protein is difficult and costly, but if we have the genetic code, we only need to identify the corresponding DNA sequence. When going from the amino acid sequence to protein structure, if we could determine a direct relationship between the amino acid sequence and the corresponding protein's three-dimensional structure, we should be able to generate a relational map from the primary amino acid sequences to the folded protein structures. Deducing this map would be another wonderful triumph that would prove tremendously useful. Once we have this second genetic code, obtaining complete information on the corresponding protein structure would only require knowing the DNA sequence.

Several factors make structure prediction from sequence extremely difficult, however. In the case of the genetic code, the possible values consist of the 20 different amino acids. Proteins consist of approximately 1,000 different major structures—called folds—each with tens of thousands of variations. In proteins, the physical forces that govern the interaction of the hundreds to thousands of amino acid residues determine the structure. We do not know the details of these interactions. Even if we knew them, computing the consequences of these forces would be nearly impossible because doing so requires solving a physics problem that involves hundreds to thousands of nonideal bodies.

Still, we have made significant progress in this area, thanks in part to the Critical Assessment of Structure Prediction (CASP) competition (http://www.ncbi.nlm.nih.gov/Structure/RESEARCH/casp3/index.shtml). This worldwide open challenge invites computational biologists to predict protein structures drawn from test cases that have been solved using experimental techniques yet to be publicly released. CASP's contest format has generated a great response, resulting in dramatic improvements in prediction rates.

From genotype to phenotype

Researchers have uncovered reasonable evidence indicating that a protein's structure approximately determines its molecular functions, such as catalysis, DNA binding, and cell component binding. Therefore, some researchers think a relational map between structure and function should be deducible, perhaps as a third genetic code. This idea also drives so-called rational drug design.

If researchers could predict the action of a protein by looking at its three-dimensional structure—much as an engineer looks at an automobile design—they could say, for example, "If we streamline the structure here and reduce the bump here, we will get a better working drug." Further, if researchers solve the sequence-structure problem, they would also know exactly how to make those changes.

Unfortunately, this ideal remains beyond reach, not least because we have at best only a murky idea of what protein function means, both in theory and practice. An object's function often depends on context. For example, a screw that holds a chair together has a markedly different context than the screw on a car jack. Because the parts of an object that confer function also often interrelate, we can't chop off a part and expect the function to remain unchanged.

All of these concerns bring us to the genotype-phenotype map problem. A *genotype* refers to the genetically encoded information in an individual genome. Technical details aside, practically speaking, the genotype consists of the sequence identity for a person's entire DNA. The *phenotype* refers to any measured trait of a particular individual—for example, a person's hair color, body weight, propensity to heart attack, and so on. Genetic encoding does not determine all these measurements, but the idea is that they result from genetic information interacting with a particular environmental context. For every environmental context, there is a map between the genotype and phenotype. Thus, given a person's DNA sequence and environmental context, we should be able to determine most measured traits. That is why many drug companies have begun to ask how the efficacy of a given drug treatment interacts with the recipient's genotype.

Ultimately, we must determine if we can take an engineering approach to biological objects. Some of the most spectacular advances in biology came from the optimistic view that we will achieve a physics-like understanding of this discipline. Perhaps, then, the Holy Grail of biology is a mechanical, physical understanding of living organisms. Is there in fact a grand theory of the organism—a set of laws that govern its form and function? Evolutionary theory has provided such laws and theories for the generative process of populations. Can we now obtain similar theories for an individual generative process? Clearly, the growing body of molecular data and its computation will play a fundamental role in forming the answer to this question.

DNA COMPUTING AND GENETIC ALGORITHMS

Leonard Adleman[4] first suggested the possibility of using DNA for computation. Adleman used sequence-specific hybridization of DNA molecules and polymerase chain reaction to solve the computational problem of finding a Hamiltonian path in a directed graph—a route that starts at a particular vertex of a graph and completely traverses each vertex exactly once before ending at a second designated vertex. A sample Hamiltonian problem would be to find the route, if it exists, from New York to Boston, that goes through each of 10 major Eastern seaboard cities exactly once.

Molecular computing

More specifically, Adleman was motivated by the computational difficulty presented by the class of NP-complete problems—here NP stands for *nondeterministic polynomial time*. Each problem in this class has the property that one can *efficiently verify* whether a proposed solution is indeed an actual solution, but no one knows how to *efficiently find* an actual solution if one exists. For the past 30 years, computer scientists have generally held that no efficient algorithm exists for any of the NP-complete problems. The problem of finding a Hamiltonian path in a directed graph is NP-complete. It is easy to verify whether a proposed sequence of vertices indeed constitutes a Hamiltonian path. However, researchers believe that there is no efficient, deterministic algorithm for finding a Hamiltonian path in a given directed graph—assuming such a path exists.

One useful observation that can be made about any NP-complete problem is that a finite set of potential solutions to propose always exists—— although that set may be of exponential size. For the directed Hamiltonian path problem, we need only consider the set of all possible permutations of the graph's set of vertices. If a Hamiltonian path exists in the graph, it must be represented by one of the permutations. We say that the problem can be solved *efficiently nondeterministically* because if we can guess one of the possible permutations, that permutation can be verified as a solution—or not—efficiently. Hence, we can efficiently solve the problem of finding a Hamiltonian path with an exponential number of processors, each verifying a different permutation. In Adleman's alternate formulation, we use the hybridization of an exponentially vast number of DNA molecules, in parallel, to solve the problem of actually *finding* a permutation that represents a Hamiltonian path.

Since Adleman's original work, additional research has shown that biologists can use DNA to encode a universal computer, the same kind of computer we use on our desktops, and that this property stems purely from DNA's ability to find complementary sequence pairs. Yet considerable skepticism still surrounds the practical use of such computers. The main issues involve concerns about

- encoding the problem and reading the output, which can take days for even simple problems;
- inherent computational errors; and
- the amount of DNA required to solve practical hard problems.

The idea of DNA computers has opened exciting new avenues of research in nanotechnology and computation models.

As an example of this last issue, currently the largest supercomputers perform approximately 10^{19} elementary switch operations per second. Although the molecular interactions of DNA hybridization can be extremely fast, cycling such operations takes approximately 10^2 seconds. It's unclear how many logical-circuit switch operations equal a single DNA hybridization reaction. However, if we assume equivalency, obtaining 10^{19} switch operations would require approximately 10 mmoles of DNA. This would amount to about 100 grams of DNA using, say, 24 base pair sequences—an unbelievably large amount of DNA for a molecular experiment.

Despite these problems, the idea of DNA computers has opened exciting new avenues of research in nanotechnology and computation models. Although real organisms *do* perform complex computations,[5] whether we can harness this ability for our specific use remains an open question.

Genetic algorithms

Researchers such as John Holland laid the foundations for genetic algorithms in the 1960s when they began describing the possibility of machines that can adaptively solve complex problems. In recent years, the term *evolutionary computing* has been applied to the diverse field that flowered from this seminal work because the predominant idea is to solve complex problems—such as a Hamiltonian problem—by emulating the evolutionary adaptive behavior of real organisms.

Strict evolutionary adaptation requires three components. First, the organism should have a property or suite of properties that governs its differential survival. Second, individuals should inherit these properties. Third, a mechanism should generate variations of these properties via mutation.

Evolutionary computing generates a population of computer programs and selects those that are particularly good at solving a posed problem.

Evolutionary computing thus involves generating a population of computer programs and—by tying their survival to their problem-solving ability—selects those that are particularly good at solving a posed problem. The reproduction and inheritance property ensures that this selection process continues through many cycles. The mutation property allows the population to continuously try new variants of the solutions. Many studies show that these algorithm classes may be particularly suitable for hard problems where "the objective function landscape is rough."

We can better understand the meaning of this phrase if we consider that many difficult problems can be visualized as searching for the highest mountain peak in a mountain range. In this metaphor, the height of the mountains represents an optimal criterion, often called the *objective function*. The land these mountains occupy represents the *search space*—the collection of possible solutions whose appropriateness the objective function measures.

For example, in the protein-structure-determination problem, the optimal criterion might be the total free energy of the folded structure, and the search space would consist of all possible ways to fold an amino acid sequence. Together, the search space and the objective function comprise the problem's landscape.

Many computer algorithms use a rule, such as "look around and go up the steepest direction," to traverse such a landscape. But, just as in real life, a rugged landscape often foils these simple rules. Although researchers remain uncertain as to exactly how the theory works, it seems that the principles used in evolutionary algorithms work well in such situations.

Evolutionary computing versus artificial life

Despite its appearance of evolving intelligence and self-direction, evolutionary computation should not be confused with artificial life. Evolutionary computing generally functions as a problem-solving device, whereas artificial life mimics an organism's behavior.

These two ideas do mix, however, in evolutionary programs. Basically, the code from these programs self-replicates and adaptively changes in response to some kind of selection scheme, usually their ability to solve a problem in the shortest amount of time. In genetic algorithms, researchers schematically code into the programs the problem to be solved. Further, this architecture does not change the scheme's parameters—they change adaptively.

In evolutionary programming, on the other hand, the actual execution of the program changes fundamentally. An example involving the protein-structure-determination problem will make this clearer. Given a string of amino acids, we must compute the three-dimensional position of individual atoms that will minimize total free energy. To implement a genetic algorithm, we might start by assigning random three-dimensional coordinates to the atoms to obtain a set of random solutions. Next, we evaluate the solutions' implied free energy and select for the lower free-energy solutions. We then change the three-dimensional coordinates to mutate the solutions.

Under the evolutionary programming approach, on the other hand, we implement a program that takes as input a string of amino acids and produces as output a set of coordinates. Then, the program itself randomly changes, rearranging its own code to, for example, duplicate some parts, delete others, and so on. The environment selects those programs that produce the right solution and make efficient computations.

The idea for a program that replicates and changes its own code first arose among Bell Labs researchers in the 1950s.[6] Tom Ray, a former tropical ecologist, created Tierra, the first truly evolving self-replicating program. An artificial world of simplified program languages that allows programs to mutate, replicate, and evolve, Tierra originally randomly generated small programs, then selected those that learned to self-replicate. Soon, other variants appeared that self-replicated more efficiently. Further, a complete ecology of computer programs evolved, including those that reproduced as a parasite hitching a ride on normal programs.

Many developments have followed Ray's work, including Avida (http://www.krl.caltech.edu/avida/), an Internet program based on concepts similar to Tierra, which lives on the Web and seeks spare computer cycles to examine issues pertaining to evolutionary biology.[7]

INTERDISCIPLINARY RESEARCH

Although many of the most exciting scientific advances have arisen from interdisciplinary work, in practice such collaboration is difficult. Many of the most creative people have trouble finding support and institutional positions. While combining the knowledge of two different fields can be difficult, we can overcome such problems if we work hard and do our homework. There is absolutely no

reason why an expert in biological sciences should not also be deeply knowledgeable in computer science, mathematics, and statistics. Thus, the main obstacles to interdisciplinary science stem from our hubris, which manifests itself in two fallacies: collaboration and expertism.

Collaborative dissonance

The *collaboration fallacy* occurs in several steps. First, an expert in one field needs "help in just this little problem" and nothing else. A biologist for example, might approach a computer scientist and say, "I have this gene-finding problem. Can you run a program for me?" The biologist assumes that

- the computer scientist will not be interested in the biology,
- it's too involved to explain everything, and
- the computer scientist would not understand the explanation anyway.

Running such a program usually requires tedious work on the computer scientist's part. Lacking the scientific context, such as why the gene is interesting and the problem important, the computing expert has no motivation to collaborate.

Second, the desire to revolutionize the other field arises. After having been told the problem's context, the computer scientist may say, "Your whole concept of a gene is flawed—*here's* how we should define a gene." Indeed such fundamental restructuring of an idea in Field A using insights from Field B can be extremely rewarding and revolutionary. However, this kind of attitude does not apply to every interdisciplinary problem. Moreover, every discipline has its own narrative tradition for posing and answering questions. Making progress requires each expert to respect the other field's tradition. Thus, fundamental revisions are a long-term, careful undertaking not to be embarked upon hastily.

Third, the collaborators assume that "an expert in Field A will have nothing useful to say about Field B and vice versa." Specifically, when two experts get together, they expect each other to stay within their own domains and communicate solely through some narrowly prescribed interface.

If a computer scientist comments on the molecular mechanisms of carcinogenesis, or if a biologist makes a remark about Lambda calculus, our first instinct is to wonder if this person knows what he or she is talking about—an extremely unfortunate situation given that such insights from outsiders can lead to important breakthroughs.

Hubris of expertism

Thus does the certainty of our scientific pedigree create the *expert fallacy*. We live in an age of specialization. Experts rule. We talk of "consulting the world's foremost expert in X" and "being taught by an expert in Y."

True, we have amassed more information today than ever before, and no single person can be expected to absorb even a small fraction of that knowledge. When two distinct disciplines such as biology and computer science come together, it may be asking too much for one person to acquire the equivalent knowledge of a specialist in each field. Yet, historically significant advances have frequently been made by such cross-disciplinary collaboration.

By its very nature, the expert concept embodies greed. It is not enough to be expert in a discipline, you must strive to be the foremost expert. If others attempt to contribute to your discipline, they threaten your goal of exceeding all others. Thus, the computer science expert may resist the notion that a biologist might know something about computer science, just as the biology expert may resist the notion of a computer scientist understanding biology.

More importantly, given a collection of experts from different fields, all will strive to position their expertise as being the most important. Thus do we learn to disdain the soft and applied sciences, non-natural sciences, mere engineering, and so on. Worse, this need to differentiate ourselves steadily shrinks the domain of our expertise until we limit ourselves to, say, the Department of the Drosophila's Left Wing.

To make interdisciplinary research successful, we must jettison this idea of the expert. All knowledge is equal. Indeed, if we really knew which knowledge is important and which is not, we could all use it with shared certainty. Growth of knowledge, whether personal or fieldwide, is haphazard and full of windings and intricate turnings. Our best hope lies in keeping our eyes open just in case something interesting happens. We should pursue knowledge vigorously but with agnostic value judgments.

To make interdisciplinary research successful, we must jettison this idea of the expert. All knowledge is equal.

ACCELERATING PROGRESS

Assuming we can overcome the obstacles to interdisciplinary collaboration, what can we expect from the interactivities between biology and computer science? In the ideal world, creativity and insight, not technical processes, would drive science.

I remember the thrill of sitting at a terminal attached to one of the early supercomputers and suddenly realizing I was getting results back as fast

as I could think of things to do. My grand hope for computational biology is that we will be able to provide this responsiveness in the future for *all* biological problems. Not being a card-carrying computer scientist, I cannot say what their dreams would be in relation to biology. But I suspect they wouldn't differ much from the idea of a new computation model that leads to robust, adaptive, flexible problem-solving machines—just like organisms.

The human brain contains about 10^{12} neurons with peak synaptic activity of about 1 kHz, resulting in 10^{15} neuronal operations per second. We obviously do not know how a single synaptic activity translates into a computer chip's elementary switch operations. But, with respect to pure information processing capability, if that number falls between 10,000 to 100,000 such operations, it seems our thinking machines can be made comparable to the human brain. At that point, we only need to determine the software and the computation model to achieve genuine artificial intelligence. This is where the interface between computation and biology becomes important.

We can anticipate a great prognosis for interdisciplinary research in biology and computers. Practically speaking, career opportunities in computational biology are increasing exponentially. Scientifically, computational approaches rapidly devour the problems so easy to approach using a computer, yet so difficult to tackle in the laboratory.

The interface between computation and biology thus offers one of the most exciting growth fields today. Certainly, some of today's claims and speculations will turn out to be noise, and some of the heat will inevitably cool. However, computers and biology form a relationship so natural and compelling that the growth of either field will critically rely on their interaction. Our best hope lies in enthusiastically embracing this relationship without the hubris of prejudice, thus hastening a time when individuals who can discuss quantum computation and post-translational regulation with equal facility initiate a wave of synergistic revolutions. ■

REFERENCES

1. J. Kim et al., "Identification of Multi-Transmembrane Proteins from Genomic Databases Using Quasi-Periodic Structural Properties," *Bioinformatics*, vol. 16, 2000, pp. 767-775.
2. E. Pennisi, "Ideas Fly at Gene-Finding Jamboree," *Science*, vol. 287, 2000, pp. 2182-2184.
3. C.M. Blaschke et al., "Automatic Extraction of Biological Information from Scientific Text: Protein-Protein Interactions," *ISMB*, 1999, pp. 60-67.
4. C.C. Maley, "DNA Computation: Theory, Practice, and Prospects," *Evolutionary Computation*, vol. 6, 1998, pp. 201-229.
5. L.F. Landweber and L. Kari, "The Evolution of Cellular Computing: Nature's Solution to a Computational Problem," *Biosystems*, vol. 52, 1999, pp. 3-13.
6. A.K. Dewndey, "Computer Recreations: In the Game Called Core War, Hostile Programs Engage in a Battle of Bits," *Scientific Am.*, May 1984, pp. 14-22.
7. C. Adami and C.T. Brown, "Evolutionary Learning in the 2D Artificial Life System Avida," *Proc. Artificial Life IV*, MIT Press, Boston, 1994, pp. 377-381.

Junhyong Kim *is an associate professor of ecology and evolutionary biology at Yale University. His research interests include computational biology, statistical theory, and molecular evolution. Kim received a PhD in ecology and evolution from the State University of New York, Stony Brook. He is a member of the Society for Systematic Biology and the Genetics Society of America. Contact him at Junhyong.kim@yale.edu.*