Figure 4. The Genetic Contig Assembly Algorithm (GCAA)

Input to CGAA

Fingerprint data and other experimental data on clone overlaps

↓

Calculate map objectives from data using computer programs

↓

Map-Objective Database

The three classes of objectives currently used:
1. Likelihood of overlap between pairs of clones.
2. Extent of overlap.*
3. Clone lengths and length uncertainties.

↓

***A Priori* Contigs**

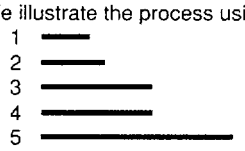
The clones are divided into *a priori* contigs—groups in which each clone is connected to each other by a chain of overlaps all having likelihoods greater than 0.5.

The map-objective data for the clones in each *a priori* contig are retrieved and stored together for input to GCAA.

*The extent of overlap between two clones is currently computed by adding the lengths of the restriction fragments that the two clones appear to share. We hope to develop a version of GCAA that optimizes locations of restriction fragments directly, rather than using only summary overlap information. This method would help ensure the consistency of overlaps between different pairs of clones.

Operation of GCAA on an *a Priori* Contig

We illustrate the process using the following contig of five clones:

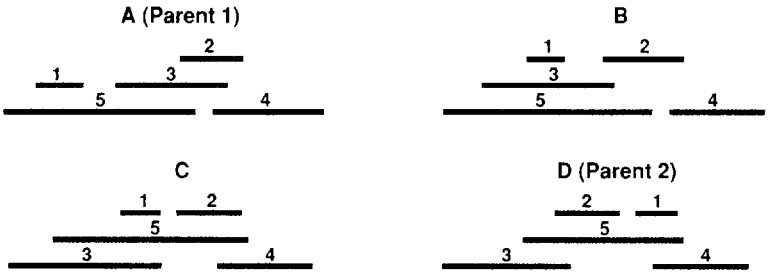


Set-up. Estimate the length of the selected contig from estimated clone lengths and overlap extents in the map-objective database. Then randomly arrange the five clones to make a population of hundreds of GCAA-chromosomes, subject to two constraints:

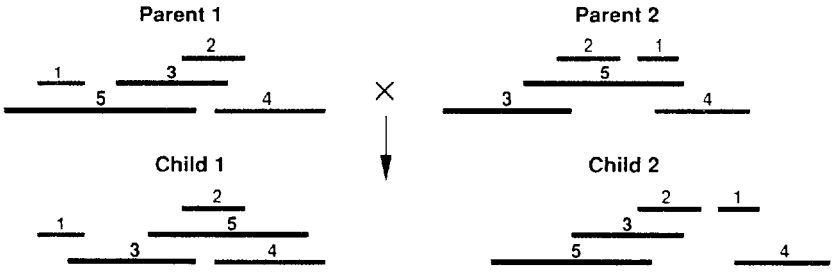
1. Clone lengths must be within a given margin, usually 10 percent, of the estimated clones lengths.
2. CGAA-chromosome lengths must be no greater than 110 percent of the estimated contig length.

↓

Tournament. Randomly select a tournament of four GCAA-chromosomes, as shown below. (In these depictions, vertical postion is irrelevant.) Select the two most disparate GCAA-chromosomes in the tournament for mating.



Mating Procedure. Arbitrarily divide the clones in the parental GCAA-chromosomes into two subsets (shown as black and red). Generate "recombinant" children from the parental chromosomes by exchange of, say, the two subsets of black clones. Note that children can have very different patterns of overlap from those of the parents.



Evolution. Evaluate the fitness of each tournament member and each child, based on the map objectives. Eliminate the two least fit, and place the survivors in the population. In the example, chromosome C and Child 2 are eliminated; chromosomes A, B, D, and Child 1 survive.

↓

Select a new tournament at random and repeat. When the fitness of a child exceeds a simple estimate of the expected fitness, or the number of iterations reaches 5,000 times the number of clones in the contig, save the fittest GCAA-chromosome for display and editing.

The mating scheme is described in more detail in Figure 4.

After carrying out the mating, GCAA evaluates the fitnesses of the two child chromosomes. If some child's overlap and overlap-extent scores are both greater than or equal to those of any of the original four members of the tournament, then that child replaces the least fit original member. (If both scores of the two chromosomes are equal, the clone-length score breaks the tie—its only function in the whole procedure.) The remaining four chromosomes are returned to the pool, and a new tournament is selected. The process is repeated thousands of times, after which the fittest GCAA-chromosomes, for most *a priori* contigs, agree quite well with the data. In practice, experienced users can often improve the output of the genetic algorithm by making small changes. However, few if any people could start from scratch with a sizable number of objectives and produce a result that needed only minor changes.

GCAA has been successfully used at Los Alamos to construct or improve large portions of the chromosome-16 physical map. At the current time, the strategy for completing the map is based on extending contigs in a highly directed way by "walking off the ends" (see "The Mapping of Chromosome 16"). Thus it is particularly crucial right now that we have a computational method to deduce as well as possible the correct arrangement of the clones in all contigs.

New data on the chromosome-16 map are, of course, accumulating daily. So it is essential to be able to apply GCAA in real time. At Los Alamos, H. Brown has built a graphical interface called map_ed for the GCAA algorithm which allows a user to retrieve map objectives from the database, run GCAA, and display or print the resulting map. Thus as new information accumulates, it is always possible to see its effect on the emerging map. Map_ed is being replaced by a more versatile system called SIGMA (discussed below).

Integration: one map is better than many maps. In everyday life one occasionally needs to use several maps of a region at one time, for example a state highway map, a map of a national forest, and a contour map of part of the forest. Each map is at a different scale and has different information, so all are needed.

The same situation obtains with genome maps, but while handling separate geographic maps is only inconvenient, with genome maps the map-construction process (which of course is foundational) is made much less accurate by having the data collected and arranged piecemeal. In fact, separate maps are all intertwined and all incomplete, and any one can be better assembled with information from all the others.

Many current map-management systems have no graphical component at all. Of course this considerably lessens usability. The two systems that do have a graphical interface (Encyclopedia of the Mouse, developed under the leader-

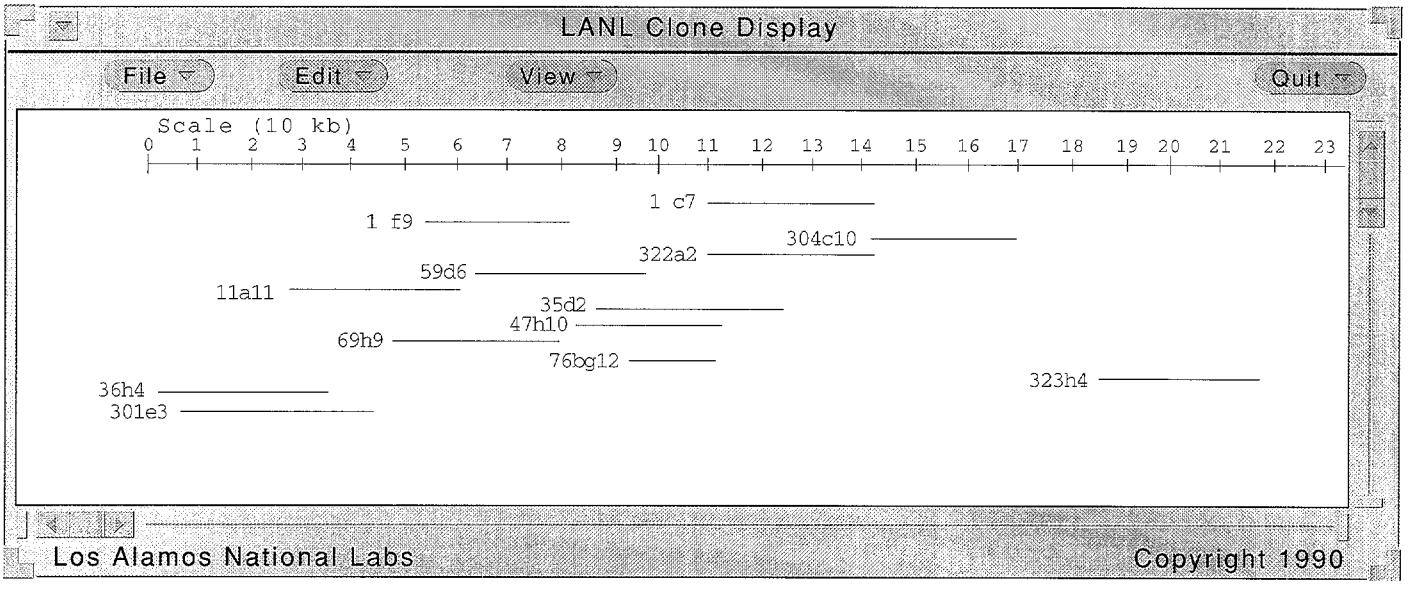


Figure 5. A Screen in Map_ed

Given the number of a starting clone, map_ed deduces the *a priori* contig containing that clone, retrieves the corresponding objectives from the database, and computes the map of the contig using GCAA. The map is displayed and may be edited, printed, and saved.

ship of J. Nadeau of Jackson Laboratory in Maine, and ACEDB, developed by J. Thierry-Mieg of the Centre National de la Recherche Scientifique in France and R. Durbin of the Molecular Research Council in Britain) manage a set of related maps in a graphical electronic "book," but do not integrate all the data for one chromosome into a single map.

We have developed a map-management system (described in "SIGMA: System for Integrated Genome Map Assembly") that gives a more integrated approach in two senses. First, map fragments given in different units are all stored as part of a single map structure, with a screen display that can be switched from one unit to another. And second, the experimental results, as summarized in the map objectives, are stored along with the map, so that the map can be evaluated or revised on the basis of the original data at any time. SIGMA has a graphical interface in the spirit of Computer-Aided Design/Manufacturing systems, in that it represents and allows manipulation of the data in a way that is close to the human conceptual model.

In the stage of widespread application, Electronic Data Publishing makes communication efficient. Getting information from producer to consumer can easily cost more, in time, energy, and money, than generating the information in the first place. This is partly due to the massive amounts of information in the modern world. But it is also due to an increase in the number of places one must look: the number of possible pairwise interactions among N people is proportional to N^2. The modern information explosion creates a widespread need for a network infrastructure which makes saving, finding, and retrieving data as cheap as possible.

A case in point is GenBank, an international collection of nucleotide sequence data managed at LANL for the last decade. The exponential growth of GenBank, described in "Decades of Nonlinearity: The Growth of DNA Sequence Data," has been due in part to the spread of sequencing as a singularly effective means of enquiry, and also to continual improvements in the efficiency of sequencing techniques. As sequencing became more efficient, GenBank had also to continually improve the efficiency of the data-entry process, or else merely collecting the data would have taken an ever increasing share of the community's resources.

GenBank pioneered in making use of the whole community's expertise to greatly increase the efficiency of the data-collection effort. How this was accomplished is described in "Electronic Data Publishing in GenBank."

The main issue in retrieval is availability. Currently many people in the GenBank user community are retrieving data from copies of GenBank updated by hand on local machines—copies that are often months out of date. These users fail to benefit from the rapid entry of newly available data into the central GenBank master copy. However, the same software that enabled us to implement the Electronic Data Publishing paradigm allowed us to easily log all changes to the database and send the resulting logs to so-called satellite copies of the database, thus updating those copies automatically. This mechanism provides a means by which an arbitrary number of copies of GenBank around the world can be brought up to date daily.

Even more difficult than keeping many databases and database copies up to date is the problem of selection and retrieval: data are only available if one can find them. For the average user it is a significant problem to find out which database(s) might contain the needed data, and then finding out how to query the relevant database(s). The problem is compounded when the answer to a user's question is spread across a number of related databases—for example map information for a gene might be found in the Genome Data Base at Johns Hopkins University, the sequence of the gene in GenBank, and related literature listed in MedLine at the National Library of Medicine.

This suggests that a key current need in information management is to make a large number of disperse and independently maintained databases appear to users as a single collection with a single query language.

Both academic computer scientists and commercial vendors have made inroads on actually integrating multiple databases, each with some autonomy, into what appears to users as a single virtual collection. However at present the multiple databases must all be managed by the same vendor's software for this to be a workable solution.

At present several groups in the molecular-biology community do provide partial solutions to this problem. One approach, implemented, for example, in the Chemical Substances Information Network system developed by the Computer Corporation of America, the National Library of Medicine, and Bolt Beranek and Newman Laboratories, is to make a smart piece of interface software that incorporates a great deal of knowledge about many individual databases. The difficulty is that as the world changes this kind of software requires a great deal of expensive maintenance. Another, more common, approach is to import copies of many databases to a single machine, and convert them all to a single format. Here, again, updating the collections and maintaining the format conversion-software is a continuing difficulty.

In data collection, Electronic Data Publishing led to a great increase in efficiency by decentralizing responsibility

SIGMA: *system for integrated genome map assembly*

Michael J. Cinkosky, James W. Fickett, William M. Barber, Michael A. Bridgers, and Charles D. Troup

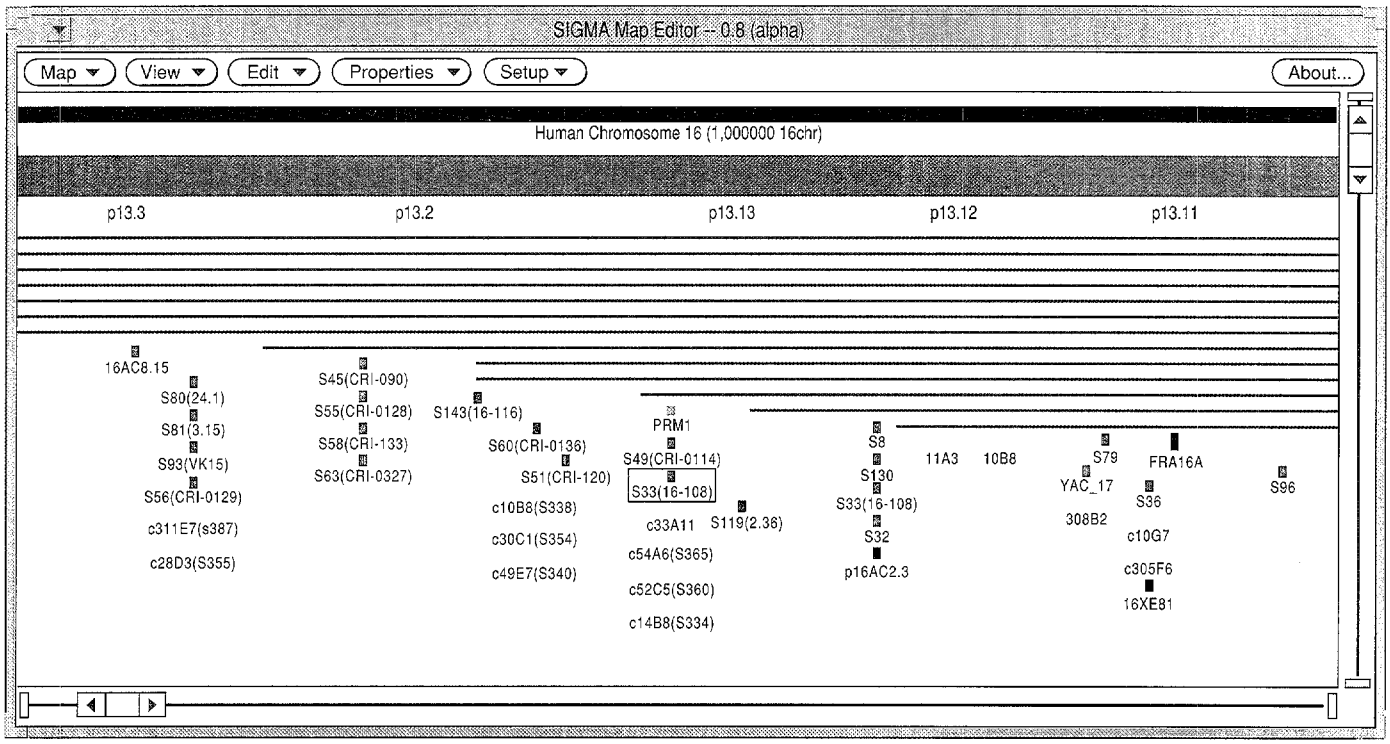
With high-quality road maps available at stores everywhere, it is easy to forget just how much effort went into the production of the first accurate geographical maps. Even maps only a few hundred years old contain glaring errors, such as the early maps of North America that show California as an island. However, when one considers how difficult it was to obtain accurate information on which to base those maps, one can understand why the maps were so inaccurate. The human genome is at present about as difficult to explore as that early wilderness was.

Although biologists have for some time been able to examine small regions in great detail, they are only now developing the experimental techniques that will allow the generation of reasonably detailed maps of each chromosome. Even now, data on the lengths of map elements and the distances between them are too fragmentary to use in building precise maps of entire chromosomes. In fact, with fragmentary data coming from many different types of experiments where even the units of measurement are incompatible, the present situation is remarkably similar to that of early cartographers who relied on the (doubtless contradictory) reports of numerous travelers returning from the area being mapped.

Unlike early explorers, however, biologists today can bring the power of computers to bear on the problem. To this end, we are producing a special-purpose tool for building accurate genome maps called SIGMA (System for Integrated Genome Map Assembly). SIGMA applies several modern ideas including object-oriented databases, optimization theory, genetic algorithms, and interactive computer graphics.

Building maps in SIGMA involves two basic activities: collecting information and drawing working maps (representations of the structure of the genome that are in reasonable agreement with experimental data). At the heart of the SIGMA system is an object-oriented database that stores all the data used in the map-building process, including all of the (potentially inconsistent) data on which the maps are based.

Maps in SIGMA can be constructed either automatically (by routines discussed below) or by users. The primary interface to SIGMA is the interactive graphical map editor shown in the figure on the next page. With this editor, users can see the positions assigned to map elements and change the positions to build or improve maps. The editor works like computer-aided drafting and design tools to let users easily view and edit the map without requiring them to understand the structure of the database in which the map is stored. Furthermore, because the software was



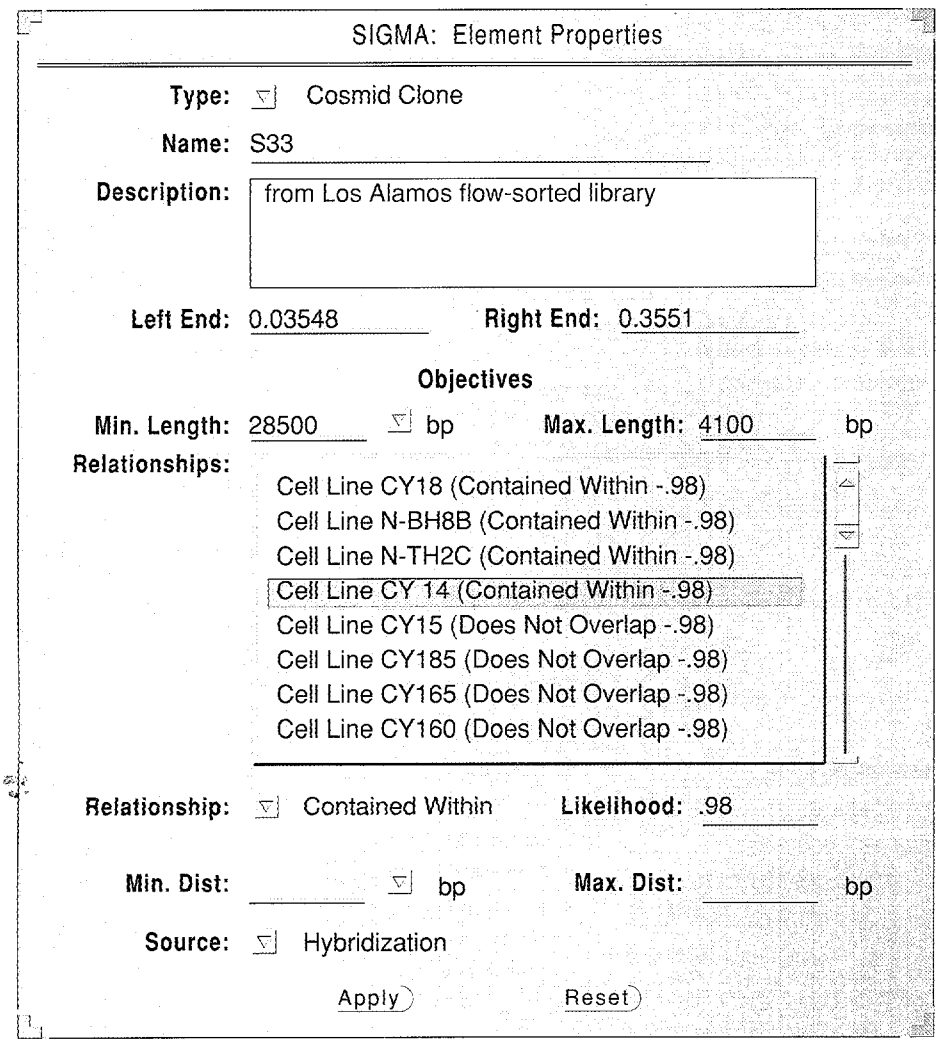
A SIGMA window is shown above as it might appear on a user's computer screen. The window contains the SIGMA map canvas, here showing a portion of a map of human chromosome 16. The display includes several different types of map elements: chromosome bands (thick bars at the top of the canvas), chromosome fragments from hybrid-cell lines (thin blue lines), anonymous DNA markers from the Genome Data Base (red bars), cosmid clones from the Los Alamos mapping effort (orange bars), YACs (blue bars), genes (green bars), and fragile sites (black bars). (The clones and fragile sites are not drawn to scale in this view because they would be too small to see.)

designed explicitly for genome maps, users have a wide choice of styles in which maps can be displayed, depending on the particular question of interest.

One problem in integrating genome maps is that conversions between the various units employed vary from one region of the chromosome to another and are even non-linear. In SIGMA, the different scales are integrated by dividing the map into regions of arbitrary size in which users can specify linear conversions between various units. For instance, in one part of the chromosome a centimorgan (the unit of genetic distance) may be set equal to a million base pairs, while in another part a centimorgan may correspond to half a million base pairs. Users can freely change the units in which the map is displayed. In the figure above the chosen linear scale is spatial distance along a metaphase chromosome as observed under a microscope. Therefore SIGMA shows element lengths and inter-element distances given in base pairs, say, according to the conversion between base pairs and spatial distance assigned for the part of the chromosome in which the elements lie.

SIGMA handles the problem of fragmentary data by treating the map-assembly process as an optimization problem. In optimization theory, one is presented with a number of (possibly inconsistent) statements that should be true about a solution to a particular problem. These statements, perhaps in conjunction with estimates of their certainty, are called "objectives". The goal is the generation of one or more solutions that satisfy the objectives as well as possible.

For genome maps, an objective is typically either a statement about a single element in the map (such as, "This YAC is about 400,000 base pairs long"), or a statement about the positional relationship between two elements (such as, "These two clones probably overlap by about 10,000 base pairs"). Even a map of only modest complexity can be based on literally millions of such objectives, far more than a human can sensibly handle. SIGMA, on the other hand, easily tracks this quantity of information and can help users find maps that meet the objectives as closely as possible. The figure opposite shows the user's view of how SIGMA manages objectives.



To demonstrate how SIGMA handles map objectives, one element, clone S33, has been selected in the map canvas; consequently its properties appear in the Element Properties Window (left). That window displays, in addition to the type, name, and description of the element, the graphical coordinates of the element in the canvas and some of the objectives involving the element. The first two objectives shown give the minimum and maximum lengths of clone S33 consistent with experiment. The objectives that follow state relationships inferred from experiments in which clone S33 was hybridized with a panel of hybrid-cell lines, each containing only a portion of chromosome 16. For each hybrid-cell line that the clone hybridized with, an objective has been created indicating that the clone lies within that chromosome fragment. For each hybrid-cell line that the clone did not hybridize with, an objective has been created indicating that the clone and that chromosome fragment do not overlap. All those objectives have been assigned a 0.98 probability of being correct, based on the uncertainty of the experiments. Finally, the last two distances in the window are the maximum and minimum values of the distance between the left endpoint of the clone and the left endpoint of the highlighted hybrid-cell line. (If the two elements overlapped, the length of the overlap would be given; if they did not touch, the distance between them would be given.)

SIGMA includes special optimization routines to automate map assembly. (The routines currently use only objectives concerning clone lengths, clone-overlap probabilities, and lengths of overlaps, which are the data used in constructing contig maps.) The optimization is performed by algorithms inspired by natural genetics, called "genetic algorithms". (See the discussion of genetic algorithms in the main text.) Whether a map was made by the optimization routines or by hand, SIGMA can automatically evaluate how well it fits the objectives. Thus the user can edit the map interactively, seeing how each change affects the map's agreement with the data.

As the map grows and new data become available, the collection of map objectives grows. Old objectives are never discarded unless a user explicitly deletes them. Because the objectives can be passed along to other users as part of a map, subsequent users of the map have access to all the information on which it is based, allowing them to make their own judgements about the correctness of the conclusions. This ability is very important when one laboratory's data appear to conflict with prior results from another group. Instead of being limited to the final product of the earlier work, the second team can look "inside" the map, examining the assumptions on which the map is based to find the specific causes of discrepancies.

Finally, SIGMA was designed from the beginning to be used with Electronic Data Publishing (see the sidebar "Electronic Data Publishing in GenBank" immediately following). Not only can users easily share data with other SIGMA users, but they can prepare submissions to the public mapping databases with just a few keystrokes. ■

Electronic Data Publishing in GenBank

Michael J. Cinkosky, James W. Fickett, Paul Gilna, and Christian Burks

Improvements in DNA-sequencing technology in the mid-1970's enabled researchers around the world to determine the exact sequence of nucleotides in samples of DNA much more easily than before (see "Decades of Nonlinearity: The Growth of DNA Sequence Data" above). Computers were the most convenient way to handle the large quantities of sequence data discovered using the new methods. Furthermore, since many people became interested in applying computer technology to interpreting those data, the data needed to be readable by computers. To meet those needs, Walter Goad created the Los Alamos Sequence Library in 1979, which in 1982 became GenBank.

Like many scientific databases at that time, GenBank was designed as a curated data repository. For its first several years of operation, the data were collected from published articles containing DNA sequence data in figures. The sequence data and related annotation (for example, information about the function and structure of the sequence) were typed into a computer and formatted into complete database entries, which were then distributed to users in both electronic and printed form.

The limitations of this style of operation became obvious fairly early. The volume of data being generated continued to grow dramatically. It became increasingly difficult for the database staff to keep up with the flow of data, and the delay between publication of an article and appearance of the data in the database grew accordingly. At the same time, the data were becoming increasingly important to biologists, which aggravated the problem of slow turn-around time for data processing.

Another problem was that a growing body of data would never, as the situation stood, appear in the database because it would never appear in print. Journals were already beginning to limit the amount of space that they would devote to printing nucleotide sequences; therefore, authors began omitting "uninteresting" sequence data (such as introns and other non-coding regions) from their papers. For computational biologists, however, those data are potentially of great interest and not having them in the public database would severely hinder some types of studies. Furthermore, in 1986 both the DOE and NIH began to talk about the Human Genome Project. If undertaken, that project would result in the generation of at least a thousand times the quantity of data that was already in the database, and probably far more. It was becoming critical to develop a different approach to building and maintaining the database.

Electronic Data Publishing

Reconsidering the problem made it clear that sequence data and results based on those data should be handled by completely separate communication methods. Whereas

scientific results needed peer review and an essentially free-form medium like the printed page, sequence data needed a largely automatic form of quality control and a highly structured, electronic format to be useful. To meet this need, we created what we call Electronic Data Publishing.

In Electronic Data Publishing, the originators of the data retain responsibility for the data in much the same way that they retain responsibility for the contents of published articles. Rather than being communicated primarily through journal articles, the data are deposited directly into an electronic database, and a separate article referring the reader to the appropriate database entries is published in a traditional journal. The database staff provides tools to help the originators get their data into the database, as well as software to provide automatic checks on the quality and integrity of the data.

To speed the transition to this new model, we enlisted the aid of many of the editors of the journals in which most of the sequence data were appearing. Because they were as acutely aware of the problems as we were (they were particularly interested in reducing the number of pages devoted to the printing of sequence data), many agreed to require submission of the data to the database before a paper discussing the data could appear in their journals. Within a year we were receiving a significant percentage of our data in electronic form before the related article appeared in print.

Table 1. Divisions of GenBank

Division	Number of entries (June 1992)	Change in number of entries since March 1992	Number of bases (June 1992)	Change in number of bases since March 1992
Bacteriophage	779	18	1,102,766	-13,880[1]
Other viruses	7,750	1,238	11,883,566	1,007,715
Bacteria	7,965	760	13,732,370	1,290,821
Organelles	2,241	130	3,721,811	409,921
Plants and fungi	6,196	682	10,713,664	1,436,907
Invertebrates	6,079	868	8,422,573	977,127
Rodents	12,737	909	13,942,988	964,730
Primates	15,996	1,257	17,258,180	1,620,375
Other mammals	2,660	215	3,537,274	355,010
Other vertebrates	3,250	276	3,915,314	342,341
RNA	2,698	162	1,517,776	134,686
Unannotated	1,649	-360[2]	1,532,138	-297,009[2]
Synthetic[3]	1,282	27	857,738	42,302
Total	**71,282**	**6,220**	**92,165,158**	**8,270,506**

Implementation of the Electronic Data Publishing model also required the development of a large software system with several major components. First, we designed and built a relational database to store the data in a far more structured manner than was practical with our original ASCII-text database format. Then we built an interactive, window-based interface to this database, called the Annotator's WorkBench, which enables people to work directly on the contents of the database. We also worked with the European Molecular Biology Laboratory and the DNA

[1]As part of our curation of GenBank, we often combine duplicated sequence data into a single representation. In the Bacteriophage division between January and March 1992, the amount of data submitted was less than the amount of duplicate data merged, so the net change during that period was a decrease.

[2]The Unannotated division of the database was formerly used to distribute data quickly by releasing them to the public in raw form prior to the more detailed work of annotation. No data have been added to this division for some time. We continue to relocate sequences from this division to their appropriate taxonomic division through annotation, resulting in a decrease of the amount of data classed as unannotated.

[3]Synthetic DNA includes such laboratory-constructed DNA as short oligonucleotide probes, cloning vectors, expression vectors, synthetic genes, etc., which cannot readily be considered as originating from single taxonomic species.

Table 2. Amount of Sequence Data from Well Studied Organisms

Organism	Bases sequenced	Number of genome equivalents sequenced	Percent of total data in database
C. elegans (nematode)	0.54×10^6	0.007	0.7
E. coli (bacterium)	2.81×10^6	0.597	3.6
S. cerevisiae (yeast)	2.95×10^6	0.203	3.8
D. melanogaster (fruit fly)	3.02×10^6	0.018	3.9
M. musculus (mouse)	6.89×10^6	0.002	8.9
H. sapiens	13.44×10^6	0.005	17.4

Databank of Japan to develop systems for sharing data, so that researchers need enter data into only one of the three databases. Finally, we created a format for automatically processable database submissions and wrote software to aid in the preparation of these submissions, which is distributed freely to anyone requesting it. Data submitted in that format are run directly into the database, where the database staff can easily use validation software that we have written to check the data for biological consistency. (As a simple example, the software checks that exons do not contain stop codons).

The impact of these changes on our operation has been dramatic. We now receive about 95 percent of our data directly from researchers, mostly in automatically processable form. In 1984, we processed sequences containing approximately 1.38 million nucleotides. At that time, it was taking, on average, more than one year from publication for data to appear in the database at a cost of approximately $10 per base pair. In 1990, we processed 10 times as much data (about 14.1 million nucleotides) with an average turn-around time of two weeks at a cost of roughly $0.10 per base pair. Further, we have been able to maintain this performance since 1990, despite the fact that the rate of submissions has more than doubled to 30 million base pairs per year in the first half of 1992.

A brief survey of the contents of GenBank indicates the extent of sequence data and the areas in which biologists have been particularly interested. Table 1 shows the contents as of release 72 (June 1992) broken down by taxonomic and other categories of origin. Approximately half the data are from expressed regions, the rest being primarily introns and sequences immediately upstream and downstream of genes. A new development is the submission of thousands of rough sequences, each a few hundred base pairs long, from human cDNAs (see pages 136–139 in "Mapping the Genome").

About 2850 organisms (including viruses) are represented in GenBank. The only completely sequenced genomes are from viruses and cell organelles (mitochondria and chloroplasts), ranging in size from a few hundred base pairs for certain plant viruses to more than 200 kilobase pairs for the cytomegalovirus. Table 2 gives information (as of December 1991) on the organisms to which the most sequencing effort has been devoted. (The heading, "number of genome equivalents," means the ratio of the number of bases sequenced from that organism to the number in its genome, without the subtraction of any duplications in the database.) In one notable recent change, the amount of sequence in the database from the nematode *Caenorhabditis elegans* increased by a factor of about 7.7 between December 1988 and December 1991, 2.5 times larger than the increase of the database as a whole. ■

for the data. The idea of using computer networks to decentralize information management and place responsibility for different tasks wherever these tasks may most efficiently be placed, we term Commonwealth Informatics. We think Commonwealth Informatics will be an important strategy for many aspects of managing information in the next decade.

For retrieval from multiple databases, Commonwealth Informatics would be implemented by building multi-database access software that depends on a single protocol for integrating databases across the network, while leaving the connection between the multi-database access system and the individual databases up to the team at each database site (for example, GenBank and GDB). Aside from the software development, the only centralized component of this retrieval scheme is an index to the available databases and the kinds of information in each. A first step in this direction is the Listing of Molecular Biology Database (LiMB), a database of databases currently maintained by G. Redgrave under the direction of C. Burks at Los Alamos.

We are beginning to learn to read the genetic program

As DNA sequence accumulates and is made widely and flexibly available by means of networked databases, significant progress is being made in learning to read the genetic programming language. The fundamental question is this: given the sequence of some region of the genome, can we discern where the genes are, under what conditions they are expressed, and what the function of the products might be?

Even very simple and partial answers to these questions have great practical importance. For example, until a few years ago diabetes was treated with either porcine or bovine insulin, available as by-products of the meat industry. Now human insulin is routinely made by means of a synthesized gene implanted in a genetically altered bacterium. (Though this artificial human insulin is widely used, not everyone agrees that it is an improvement over animal insulin. Some studies indicate that artificial human insulin produced in genetically engineered plants may undergo more human-like post-processing of the protein product than artificial insulin produced in bacteria.) While the protein products of the synthetic bacterial gene and the natural human gene are identical, the two genes are quite different. In fact, the natural human gene would not even function in a bacterial cell. The human gene has two introns; because bacteria cannot excise introns, the synthetic gene must have none. The human gene has control elements that turn on the gene only when needed. The bacterial version has a control element that maintains maximum production levels at all time. Even the codons that are used in the synthetic gene, while specifying the same sequence of amino acids as those in the natural gene, have been chosen to maximize the rate of production. The design and implementation of this synthetic gene is made possible by a very incomplete, but still very powerful, understanding of the bacterial programming language.

Again, many genetic diseases are far better understood now than they were only a few years ago, because the region of the genome in which the defect lies has been located and the cause of the disease studied directly. Sickle-cell anemia results from a single-nucleotide change in the alpha hemoglobin gene (a 0.0000000002 percent change in the genome). The gene whose corruption

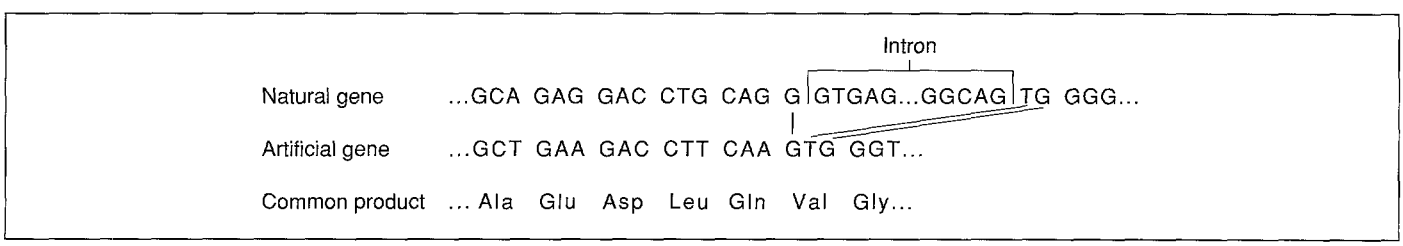


Figure 6. Comparison of Part of the Natural and Artificial Human Insulin Genes

The regions of the natural gene just preceding and just following the second intron are shown, along with the corresponding part of the artificial gene. The intron has been deleted in the latter. Note also that the two genes have different sequences, but the same protein translation.

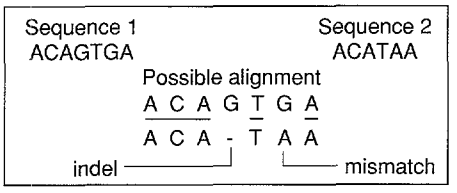


Figure 7. A Simple Alignment
Shown is one possible alignment between the two sequences ACAGTGA and ACATAA. In order to match bases near each end, a deletion has been introduced in the second sequence (shown by '-'). A horizontal line between the two sequences indicates a matching base; a space indicates a mismatch.

causes cystic fibrosis was located in 1989, and the change in function of the encoded protein is now being elucidated. Several specific forms of gene therapy, in which the defective region of the genome is repaired, are being tested in clinical trials. One important component of elucidating and treating genetic defects is the computational technology for analyzing sequence data to find and interpret genes.

The tool most used for analyzing sequence data is calculation of similarity. When a gene is newly sequenced it is very desirable to discover its biochemical means of action. The state of our knowledge does not allow us to predict the enzymatic activity of a protein from its sequence, but we often can shed important light on the function of a newly sequenced gene by comparing it with all other known sequences (as one who is just learning a foreign language can guess at the meaning of a phrase by comparing it with similar sounding known phrases). If there is a similarity to some gene that has already been studied, anything known about the biochemistry of the previously sequenced gene may help decipher the workings of the newly sequenced one. Of course, such comparison normally suggests hypotheses and further experiments, rather than completely elucidating the function of the sequence.

There are a number of difficulties in finding meaningful alignments between pairs of sequences. At the root of these difficulties is the fact that biologically meaningful alignments contain both mismatches and indels (short for "insertions or deletions"). Figure 7 gives a simple example of an alignment; a longer example without indels appeared in Figure 2.

The most basic alignment algorithm is the so-called dynamic-programming algorithm, first described in print by S. Needleman and C. Wunsch, and still widely used in several variations. The purpose of this algorithm is to find that alignment which has the lowest cost, where the cost is the number of mismatches times a preset mismatch penalty, plus the number of indels times a preset indel penalty. If the sequences are $A = a_1 a_2 \ldots a_M$ and $B = b_1 b_2 \ldots b_N$, the algorithm proceeds by calculating inductively all optimal alignments between initial segments of A and initial segments of B. That is, let A_m be the string consisting of the first m characters of A (where $1 < m < M$), and B_n be the string consisting of the first n characters of B (where $1 < n < N$). Then the algorithm calculates the best alignment between every A_m and every B_n by extending shorter alignments one base at a time. The scores of those alignments can be laid out in an $M \times N$ matrix in which the (m, n) element, in the $(m+1)$st column and $(n+1)$st row, is the optimal score for aligning the first m characters of the first (top) sequence with the first n characters of the second (side) sequence. Figure 8 shows such a matrix.

The first alignments constructed are the trivial ones between the A_m's and the empty sequence as well as those between the B_n's and the empty sequence; their scores are the costs of deleting those segments, which are the indel penalty times m or n respectively. Those scores appear in the top row and left column of the matrix in Figure 8. The remaining alignments and their scores are calculated as follows. The best alignment between A_m and B_n is the best of these three possibilities, all based on previously calculated alignments between shorter sequences: (1) the best alignment of A_{m-1} with B_{n-1}, followed by a match or mismatch of a_m with b_n, or (2) the best alignment of A_m with B_{n-1}, followed by the deletion of b_n, or (3) the best alignment of A_{m-1} with B_n, followed by the deletion of a_m. Constructing all the alignments of initial segments results in calculating the best alignment of A with B as the culmination of the process. Figure 8 shows how the process aligns the sequences in Figure 7.

Quite different optimal alignments may result, depending on whether untranslated gene (nucleotide) or translated protein (amino acid) sequences are compared, and depending on what scoring scheme is used. Current consensus is that the most functionally meaningful alignments between related genes are found by aligning protein sequences with a scoring scheme that takes into account chemical similarity between different amino acids.

Speed is a major concern in searching databases for similar sequences. When a sequence is newly determined, the investigator will normally want to compare it to every sequence in GenBank, both to find out if the DNA fragment has been sequenced before, and to try to discover the function of the DNA sequenced by comparison with other, related, sequences (from the same or different organisms). The straightforward dynamic-programming algorithm described above would, if applied to a typical sequence of 1000

bases, take on the order of a day on the fastest general-purpose single-processor computers. Faster response time is very desirable, so considerable effort has gone into accelerating comparison between a single "query" sequence and a database.

Specialized hardware can greatly increase the speed of searching a database. For example the problem is almost trivially parallelized: R. Jones of Thinking Machines has written algorithms, for the CM2 connection machine with 64,000 processors, that split the database among the processors so that each one only does a few comparisons. In another direction, T. Hunkapillar of the California Institute of Technology has implemented the dynamic-programming algorithm in hardware, producing the so-called BISP (Biological Information Signal Processing) VLSI chip. The BISP chip is not yet widely available, but is reported to be capable of comparing a query sequence (of any length) against a database at the rate of 12,000,000 database nucleotides per second. This makes database access, rather than algorithm speed, the rate-limiting step for most applications.

Databases are most often searched on personal computers and workstations. Thus another approach that has been extensively pursued is to narrow the search and make detailed searches only in promising areas. First the database is pre-indexed by making a so-called "hash table" of all "words" (subsequences) of a given length (typically 4-10). Then each time the program is run, all locations in the database of all words of the chosen length from the query sequence are found using the hash index. Finally, where there are promising "clumps" of matches, more detailed comparisons are made using the dynamic-programming algorithm. W. Pearson (U. Virginia) and D. Lipman (National Library of Medicine) pioneered this approach with an algorithm called FASTA.

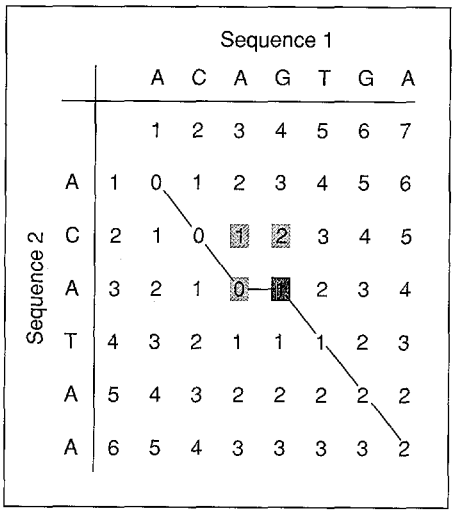


Figure 8. An Illustration of Dynamic Programming

The two sequences are those shown in Figure 7, and the scoring scheme is the simple one where the cost of both mismatches and indels is 1. The matrix shows the scores of all optimal alignments of initial segments of the two sequences. The first row and first column of the matrix give the trivial initialization scores, equal to the costs of simply deleting the corresponding initial segments. The matrix is then filled in one row at a time, from top to bottom and left to right. The induction step described in the text may be illustrated with the matrix cell containing a 1 boxed in red [the (4, 3) element]. The value of 1 in this cell is calculated on the basis of the values in the (3, 2), the (4, 2), and the (3, 3) cells (all highlighted) as follows. The best score for aligning ACAG of the top sequence with ACA of the side sequence must logically include one of three shorter alignments: (1) An alignment of ACA from the top sequence with AC from the side sequence. The best score of such an alignment is 1 [in the (3, 2) cell]. (2) An alignment of ACAG from the top sequence with AC from the side sequence. The best score of such an alignment is 2 [in the (4, 2) cell]. (3) An alignment of ACA from the top sequence with ACA from the side sequence. The best score of such an alignment is 0 [in the (3, 3) cell]. In case 1 the rule given in the main text calls for extending the alignment of ACA with AC to an alignment of ACAG and ACA by a mismatch of G with A, which would give a score of 2 for the boxed element. In case 2, the alignment between ACAG and AC is extended to an alignment between ACAG and ACA by a deletion of the A at the end of the second sequence, giving a score of 3. Finally, case 3 requires a deletion of G from the first sequence, resulting in a score of 1. The best of these three scores is 1, so this is what appears in the box. Once the matrix is full, the program chooses the best score along the right and bottom edges, and works backwards through the matrix to find what shorter alignments gave rise to this best score. The black line shows the set of best shorter alignments, and hence the best alignment, for these two sequences. Given the scoring system used, the best alignment is that shown in Figure 7.

An even faster algorithm called BLAST (Basic Local Alignment Search Tool) has been developed by S. Altschul, W. Gish, W. Miller, E. Myers, and D. Lipman, at the National Library of Medicine, Pennsylvania State University, and University of Arizona. BLAST first compiles a list of the words in the query sequence, then expands it to include all words "near" these—that is, such that the score of a no-gap alignment with one of the words in the query sequence meets a certain cutoff—and then uses the hash table to find promising sequences for more detailed analysis. On such sequences BLAST extends the word matches to longer segment matches, but does not perform the full dynamic-programming algorithm. Running with typical parameters on a Sun Sparcstation, BLAST can search GenBank in about twenty seconds. With these algorithms there is always a chance of missing an unusual alignment that does not fall within the initial pre-screening criteria. However, most investigators consider

the trade-off of sensitivity for speed to be quite acceptable.

The amount of sequence data will continue to grow rapidly. However with accompanying advances in hardware, and with refinements of current algorithms, it appears that comparing new sequences with the corpus of known data will remain practical, and an important source of insight.

Finding genes. The central functional component of the genome is the gene, which may be defined in computational terms as a pattern imposed on the DNA sequence, resulting in a protein (or, sometimes, RNA) product. (See "The Anatomy of a Gene" in "Understanding Inheritance.") At the present time there is no sure way, either experimental or computational, to locate all the genes in a DNA sequence. However computational techniques can provide a very useful starting point in locating likely candidates for genes. The techniques are particularly well developed for finding genes coding for RNA that is not translated into protein. For instance, G. Fichant and C. Burks of LANL have developed a highly effective algorithm for finding tRNA genes. The rest of this discussion will refer only to the more complex problem of finding protein-coding genes.

Computer recognition of genes is not a simple problem. As far as we understand at the moment, there are no simple, local, key patterns that one can use to detect the presence of genes. For example, every triplet of bases that occurs as a codon in genes (where it stands for a particular amino acid; see "The Genetic Code" in "Understanding Inheritance") also occurs many millions of times outside of genes, with no meaning that we yet recognize. Thus solving the gene-recognition problem depends on integrating information from a number of clues spread out over thousands of bases of sequence.

In the long run, as we seek to understand how the genome works, our hope is to know how the cell recognizes genes. That is, we want to know what the enzymes are that control gene expression and how they recognize control sites on the DNA. Much progress has been made in elucidating the control of transcription and translation of genes in prokaryotes (simple one-celled organisms without nuclei). But the control of gene expression in humans is much more complicated, and the computer recognition of human control elements is still in its infancy.

There is, however, another approach. While we do not yet know enough about DNA-protein interaction to recognize genes the way the cell does, we can recognize certain patterns in a gene region that are side-effects of the way the gene is built. The simplest pattern is called an open reading frame. Reading frames are the six possible ways in which any stretch of DNA can be interpreted as a string of codons, depending on which strand is read and on whether a given base is interpreted as the first base of a codon, the second, or the third. A reading frame is said to be open in a region where it contains no stop codons, which are the triplets of bases that signal the end of translation of mRNA into protein. (See "The Genetic Code" and "Protein Synthesis" in "Understanding Inheritance.") In most organisms the stop codons on the sense strand of a gene are TAG, TAA, and TGA. (The sense strand has a base sequence equivalent to that of the mRNA.) Figure 9 shows the three reading frames of one strand of a viral sequence; stop codons are marked.

Since the genes of prokaryotes (and bacterial viruses) are uninterrupted, the protein-coding portions of their genes must lie in long continuous open reading frames. Most prokaryotic genes consist of at least fifty codons, and more typically hundreds, which do not include any stop codons. On the other hand, an entirely random sequence of bases contains stop codons on average about once in twenty-one triplets in each reading frame. Therefore long open reading frames in prokaryotic and bacteriophage genomes are likely to contain genes. The third reading frame in Figure 9 is an example.

To find genes in eukaryotic genomes, one must look for more subtle patterns, mainly because eukaryotic genes are divided into exons (protein-coding regions) separated by introns (non-coding regions). Long open reading frames are still good candidates for exons, but some exons are as short as ten base pairs. Moreover, eukaryotic genomes contain long open reading frames that are not expressed. Therefore attention has turned to sequence patterns that distinguish coding from non-coding sequence. In the main, these patterns arise because coding sequences obey what are called codon preference rules. In most cases the same amino acid can be specified in genes by any of several synonymous codons. This latitude in choice of codon seems to be exploited systematically, in such a way that different bases are more common in different codon positions. For example T occurs more often at the second position of codons than at the first or third. The periodicity arising from these preferences is strikingly illustrated in the autocorrelation functions of the individual bases. Figure 10 shows the autocorrelation functions for the occurrences of T in coding and non-coding regions.

A variety of statistical techniques can be used to detect the nonrandom choice of triplets in coding regions. Such measurements can give an algorithm which, on a sample of about 150 bases of sequence, can differentiate protein coding from noncoding regions about 95 percent of the time.

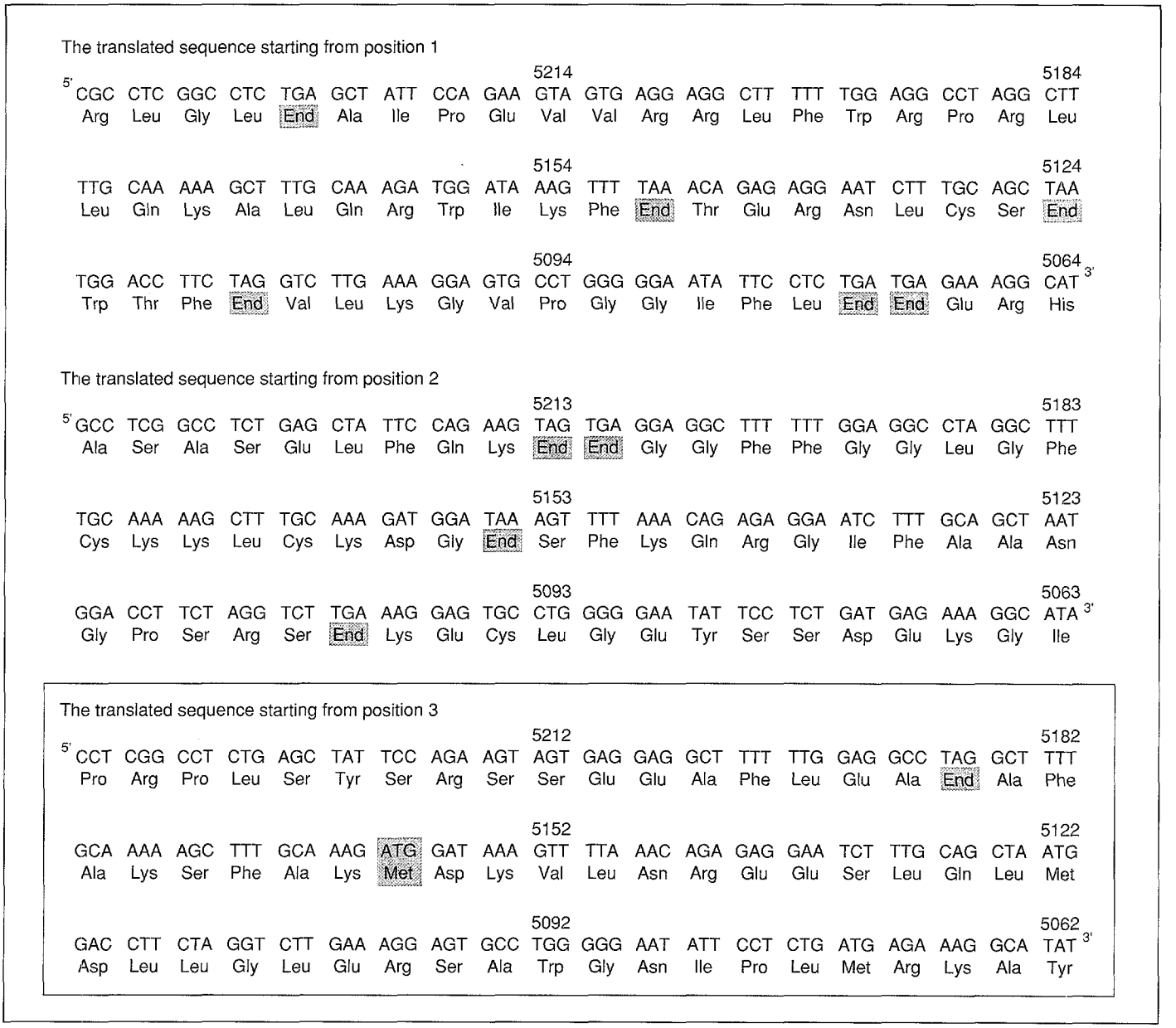


Figure 9. A DNA Sequence in Three Reading Frames

The nucleotides numbered 5243 to 5062 of the genome of the simian virus SV40 are shown. (The strand depicted is the one known to be the sense strand in this region.) Also shown are the three possibilities for the translation of the sequence, each using a different reading frame, or division of the sequence into triplets of nucleotides. In this part of the SV40 genome, the first two reading frames depicted contain many stop codons (translated as "END" and highlighted), so the region does not code for proteins when read in those frames. In the third reading frame (boxed), on the other hand, there is a long region without stop codons—a promising candidate to be a protein-coding region. In fact, experiments have demonstrated that the sequence shown does include the beginning of a gene, whose translation starts with the highlighted ATG codon. (In the great majority of mRNAs, translation starts with AUG, corresponding to ATG in the sense strand of the DNA and to methionine in the protein product.) (Adapted from a figure by Maxine Singer and Paul Berg. *Genes and Genomes: a Changing Perspective.* Mill Valley, CA: University Science Books, 1991.)

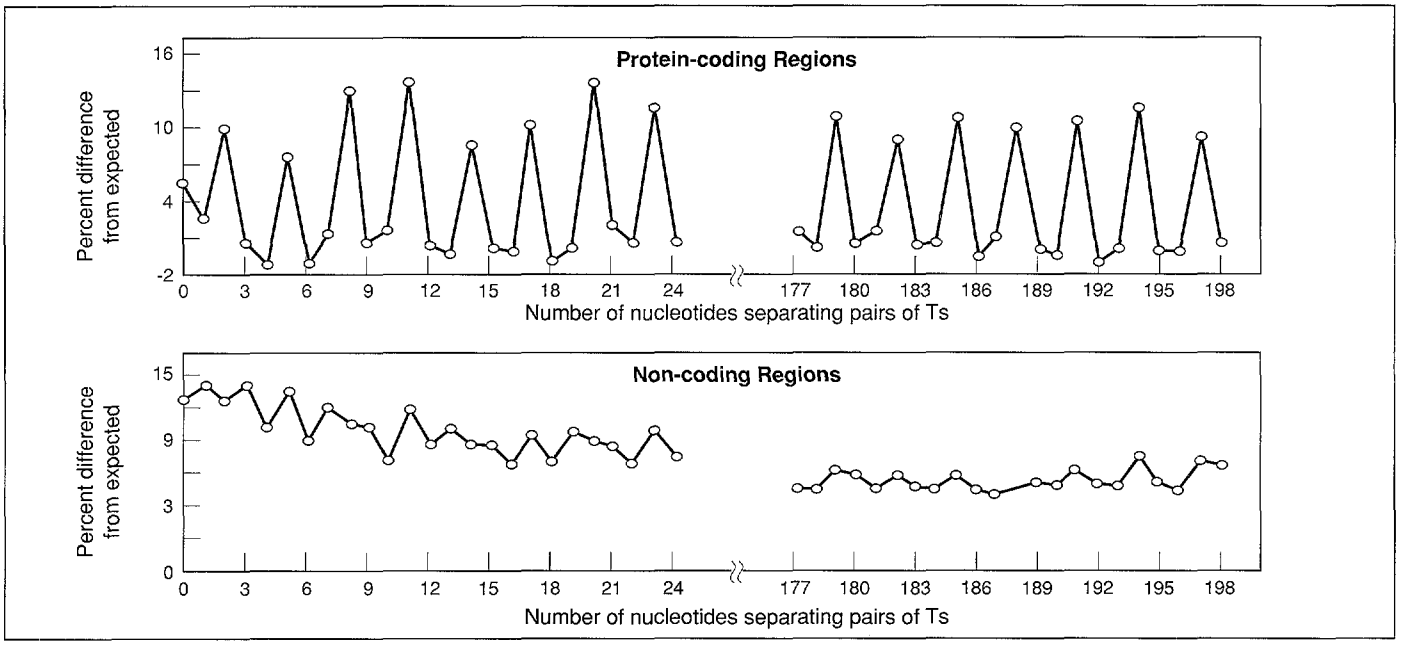


Figure 10. Periodicity of T Due to Codon Preference Rules

For each value of the separation *n*, the number of occurrences of the pattern T . . . T, with *n* nucleotides between the two T's, was counted. Plotted is the percent difference of that number from the number of such pairs expected if nucleotides occurred at random. Results are shown for all the coding sequences (a) and all the non-coding sequences (b) in GenBank when this study was performed (1982). Since T occurs preferentially at the second codon position, in coding regions the percent difference at $n = 2, 5, 8 \ldots$ is noticeably large. T's separated by two nucleotides, for instance, are at corresponding positions in consecutive codons. No such pattern appears in non-coding regions. Results for the other bases and for pairs of unlike bases show similar differences between coding and non-coding regions.

Research continues to find ever more accurate discrimination methods. R. Farber, A. Lapedes (both of Los Alamos), and K. Sirotkin (National Institutes of Health) report that a single-layer neural net reading each group of six consecutive bases can differentiate exonic from intronic sequences 180 bases long with a sensitivity well over 99 percent. With accuracies of 95 percent and sensitivities of 99 percent already in hand, the main hindrance to further development may soon be the accuracy of the databases. Though every care is taken by both investigators and database staff to make annotation both complete and correct, it is quite possible that the database annotation as to whether regions are coding or non-coding, by which these algorithms are measured, contains errors or omissions of a few percent.

All known algorithms depending on codon preference (so-called region methods) are rather poor at picking out the precise endpoints of coding regions. Thus current emphasis in this field is shifting towards combining region methods with recognition methods for biochemically active sites of transcription and translation initiation, intron splicing, etc. Two such systems have now been described in print and publicly disseminated: GM (for Gene Modeler), written by C. Fields (National Institutes of Health) and C. Soderlund (Los Alamos), and GeneID, written by R. Guigo (Los Alamos), S. Knudsen (University of West Florida), N. Drake (Tufts University) and T. Smith (Brown University).

Both of these programs analyze many different patterns over a large stretch of sequence, integrate the results, and present the user with a number of possible ways in which a gene or genes might be encoded in the sequence. The state of the art is that programs can suggest possible genes, and that the real genes in the region are likely to be at least variants of the ones proposed. It is not possible at present to predict the precise form of the gene or the conditions under which it is expressed.

Prediction of structure and function of proteins. Current techniques for the interpretation of sequence data are almost universally of what one might

call a linguistic nature: they depend on the existence or frequency of certain simple patterns of letters in the DNA-sequence string. However it is not to be forgotten that the basis for all the effects of DNA in the living cell is the three-dimensional shape and charge distribution of biomolecules. In the long run, our understanding of the biochemistry of DNA, and therefore of the principles underlying the DNA programming language, will depend on our ability to relate the nucleotide sequence of DNA, and amino-acid sequences of protein, to the three-dimensional molecules of life. The promises of this infant science, a part of structural biology, are great, but remain mostly in the future.

Summary

The information gained from the Human Genome Project will reside in a very large database listing and describing the program for constructing and running the human body. With the development of new information-management techniques this information will be efficiently gathered from, and distributed to, a loosely coordinated and global community of scientists. Analysis tools are being developed to read the genome program and describe its functionality. While our knowledge of the biological programming language, and the tools we have to interpret it, are both at an early stage, they are also both very powerful, giving daily fundamental new insights into the workings of cells, organs and organisms, and leading to more powerful biotechnology. ■

Acknowledgments

I am grateful for insightful remarks from Christian Burks, Michael Cinkosky, Doug Sorensen, David Torney, and the editors.

Further Reading

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. 1990. Basic Local Alignment Search Tool. *Journal of Molecular Biology* 215:403-410.

M. Cinkosky, J. Fickett, P. Gilna, and C. Burks. 1991. Electronic data publishing and GenBank. *Science* 252:1273-77.

R. F. Doolittle, editor. 1990. *Methods in Enzymology* 183. San Diego: Academic Press. (A special issue of the journal devoted to sequence-analysis methods)

G. A. Fichant and C. Burks. 1991. Identifying Potential tRNA Genes in Genomic DNA Sequences. *Journal of Molecular Biology* 220:659-671.

R. Guigo, S. Knudsen, N. Drake, and T. Smith. 1992. Identification of genes in DNA sequences. *Journal of Molecular Biology*, in press.

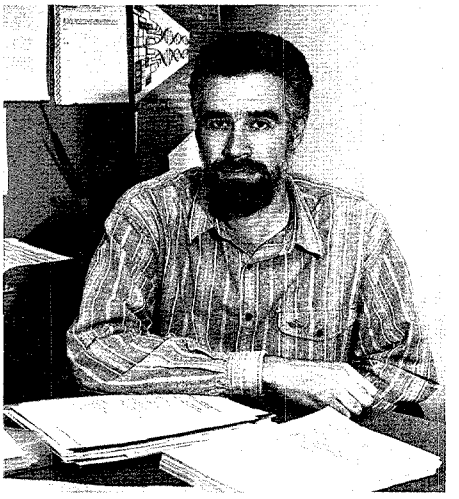
A. Gupta, editor. 1989. *Integration of Information Systems: Bridging Heterogeneous Databases.* New York: IEEE Press.

J. Holland. 1975. *Adaptation in natural and artificial systems.* Ann Arbor, Michigan: University of Michigan Press.

J. R. Lawton, F. Martinez, and C. Burks. 1989. Overview of the LiMB database. *Nucleic Acids Research* 17:5885-5889.

S. G. Oliver *et al.* 1992. The complete DNA sequence of yeast chromosome III. *Nature* 357:38-46.

M. L. Pearson and D. Söll. 1991. The human genome project: a paradigm for information management in the life sciences. *The FASEB Journal* 5:35–39.



James W. Fickett is the head of the Human Genome Information Resource and section leader for Genome Analysis and Informatics. [See "Members of the Human Genome Center at Los Alamos National Laboratory" for biographical details.]