BIOINFORMATICS

Vol. 17 no. 2 2001
Pages 137–148

The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation

Bruce A. Shapiro[1,*], Jin Chu Wu[2], David Bengali[1] and Mark J. Potts[3,4]

[1]Image Processing Section, Laboratory of Experimental and Computational Biology, Division of Basic Sciences, National Cancer Institute, Frederick Cancer Research and Development Center, National Institutes of Health, Bldg 469, Rm 150, Frederick, MD 21702, USA, [2]Science Applications International Corporation at Frederick, LECB, NCI-FCRDC, Frederick, MD 21702, USA and [3]SGI Inc., 12200-G Plum Orchard Drive, Silver Spring, MD 20904, USA

Received on June 13, 2000; revised and accepted on October 11, 2000

ABSTRACT

A massively parallel Genetic Algorithm (GA) has been applied to RNA sequence folding on three different computer architectures. The GA, an evolution-like algorithm that is applied to a large population of RNA structures based on a pool of helical stems derived from an RNA sequence, evolves this population in parallel. The algorithm was originally designed and developed for a 16 384 processor SIMD (Single Instruction Multiple Data) MasPar MP-2. More recently it has been adapted to a 64 processor MIMD (Multiple Instruction Multiple Data) SGI ORIGIN 2000, and a 512 processor MIMD CRAY T3E. The MIMD version of the algorithm raises issues concerning RNA structure data-layout and processor communication. In addition, the effects of population variation on the predicted results are discussed. Also presented are the scaling properties of the algorithm from the perspective of the number of physical processors utilized and the number of virtual processors (RNA structures) operated upon.

Contact: bshapiro@ncifcrf.gov; jcwu@ncifcrf.gov; bengalid@ncifcrf.gov; mark.potts@ieee.org

INTRODUCTION

The Genetic Algorithm (GA) is a non-deterministic optimization procedure (Holland, 1975). It was derived from the concept of biological evolution using operations based on the survival of the fittest, crossover, mutation, and selection to mimic the genetic process. Genetic Algorithms have been applied in a wide variety of fields in science and engineering (Goldberg, 1989; Holland, 1992; Miettinen *et al.*, 1999) and they have been developed for RNA sequence folding (Shapiro and Navetta, 1994; Gultyaev *et al.*, 1995; Shapiro and Wu, 1996, 1997; Wu and Shapiro, 1999).

As described in the literature (Goldberg, 1989), the original GAs used a fixed binary-string representation for the population of objects that evolved towards fitness. However, our GA for RNA molecular folding works differently. It is an evolution-like process applied in parallel to a large number of RNA structures that are built from a pool of helical stems which are generated from an RNA sequence. This parallel paradigm has been implemented on several different supercomputer architectures.

The GA was originally developed on the massively parallel architecture of a MasPar MP-2, a SIMD (Single Instruction Multiple Data) supercomputer with 16 384 physical processors. Recently, this parallel-RNA-folding paradigm has been adapted to MIMD (Multiple Instruction Multiple Data) parallel supercomputers, namely a 64 processor SGI ORIGIN 2000 and a 512 processor CRAY T3E. It should be noted that even though the number of processors available on these MIMD machines is substantially smaller than on the MasPar, their processor speeds are considerably faster. This adaptation has also made it possible to run the GA on a single or dual processor SGI OCTANE with the shmem (shared memory) library.

The GA is briefly described in the following section. The implementations of our GA on a SIMD machine and on a MIMD machine are then presented. Next the issues related to population effects are discussed, then visualization procedures associated with the GA are examined, leading to the final conclusion section.

*To whom correspondence should be addressed.
[4]Current address: HPC Applications Inc., 10080 Old Frederick Road, Ellicott City, MD 21042, USA.

THE MASSIVELY PARALLEL GENETIC ALGORITHM FOR RNA FOLDING

RNA sequence and structure

An RNA sequence consists of four different nucleotides: adenine (*A*), cytosine (*C*), guanine (*G*), and uracil (*U*). These nucleotides can form relatively stable base pairs, such as *C* and *G*, *A* and *U*, *G* and *U*, and vice versa. Other combinations are possible, but with less stability. A contiguous series of these base pairs can form a stem, which is described by a three-tuple which includes its 5′ start position, the 3′ stop position, and the number of base pairs in the stem. A fully zipped stem is defined as a stem in which maximal complementarity exists amongst its bases, i.e. a larger stem could not be generated using standard complementary base pairs. Partially zipped stems are stems in which some of the complementary base pairs at the ends are opened. The ends of stems may be free strands or may form loops which are considered to contain un-paired nucleotides. The RNA loops are classified as being a bulge loop, hairpin loop, internal loop, or multibranch loop, depending on their morphological structure.

A structure derived from an RNA sequence is a configuration of stems, strands, and loops generated from the sequence. It can consist of either secondary or tertiary structural elements. The secondary structure of an RNA sequence is a two-dimensional structure, and can be represented by a tree. Thus, it can be uniquely described by a table of stems. The tertiary structure of an RNA sequence consists of interactions between the secondary structural elements. Some of these interactions form structures called pseudoknots, which are created by the interactions between a loop and a free strand or a loop and a loop (Shapiro and Wu, 1997). The size of an RNA structure is usually defined as the number of stems in the structure.

The fitness of an RNA structure is determined by computing the change in Gibbs free energy for the formation of that structure relative to a fully single stranded molecule. It is the sum of the stem energies and the loop energies. The negative stem energies tend to stabilize an RNA structure, and the positive loop energies tend to destabilize the structure. The free energy of an RNA stem is a combination of the hydrogen bond energies between base pairs and the stacking energies formed by the stacking of one base pair on another. The free energy of pseudoknot structures is dependent on the size and base content of the stems constituting the pseudoknot, the structure and size of the related loops connecting the stems, and the structures at the junctions between the stems and loops. All stem energies, pseudoknot stem energies, loop energies, and eventually the entire RNA structure energy are computed by the use of various energy rules (Freier *et al.*, 1986; Abrahams *et al.*, 1990; Walter *et al.*, 1994; Matthews *et al.*, 1999).

The massively parallel genetic algorithm

The goal of RNA folding is to fold an RNA sequence into a biologically functional structure, that is stable with an optimal or suboptimal free energy. The non-deterministic GA has been adapted to folding RNA sequences and, in addition, has incorporated the ability to form simple pseudoknots in a natural way. In our GA, a large population of RNA structures is laid out over an extensive number of computer processors such that each processor holds one RNA structure. These processors are arranged in a rectangular configuration that is toroidally wrapped in both the x- and y-directions. This means, for example, that a processor to the north of a processor sitting on the northernmost border of the rectangle is located on the southernmost border. All RNA structures are evolved in parallel, one generation at a time (an iteration constitutes one generation) through a three-step procedure consisting of the three basic operators: selection, mutation and crossover, using the stems generated from a given RNA sequence. Minimal free energy is used as a criterion to improve the population of structures across all the processors.

Before starting the evolution process, the GA generates a stem pool consisting of all fully (or partially) zipped stems from the given RNA sequence, and stores this pool across all the processors. For a long RNA sequence, the stem pool can become quite large. For this reason, unzipping is usually limited to one or two base pairs at each end. At the start of the evolution process, the GA initializes a structure on each processor by stochastically picking stems from the stem pool. These stems are geometrically compatible and are subject to some restrictions (see below). A set of small, different structures are therefore formed on the set of processors simultaneously.

Then, the GA goes through the selection, mutation, and crossover processes to evolve RNA structures on all processors in parallel at each generation. First, the selection operator is applied. On each processor, the GA selects two RNA structures, P_1 and P_2, to be parent structures. This selection is done from the set of nine structures including the structure on the processor itself and the structures on its eight-neighbor processors. This choice of parents is made by using a ranked rule biased towards structures with better free energies.

Next, the mutation operator is applied. On each processor, the GA picks stems randomly from the stem pool to form two child-structures, C_1 and C_2. The same criteria for picking stems that are used during the GA initialization process are applied during this operation. The number of stems that are selected from the stem pool at each generation is in accordance with an annealing mutation operator (Shapiro and Wu, 1996). This operator allows a fairly large number of mutations to take place across all of the processors at the very beginning of the evolution-like process and reduces the total number of mutations at each generation as the size of the structures increases. Thus, the operator cools down the mutation process slowly as the GA proceeds (Press *et al.*, 1992). Usually, the longer the RNA sequence, the larger the stem pool generated from the sequence. Thus, our annealing mutation operator treats different length RNA sequences appropriately, producing more mutations for longer sequences.

After the mutation operator is applied, the crossover operator is used. On each processor, the GA performs a crossover operation between (P_1, P_2) and (C_1, C_2), distributing stems from P_1 and P_2 into C_1 and C_2 to complete the two new structures. During crossover, the same conditions for choosing stems that were used in the initialization procedure and in the mutation process are applied. Finally, the GA chooses the resultant child structure with the better free energy to become the new structure in the processor for the next generation. Thus, at the end of each generation, the GA produces new structures on all processors in parallel.

As the GA proceeds, after each generation there is a distribution of free energies over all the processors. Because of the annealing mutation operator, after hundreds of generations (depending on the length of the sequence), the distribution of free energies of structures over all the processors can converge. That is, the annealing mutation operator generates a statistical consensus of the free energies of the RNA structures over all the processors. This implies that as the generations proceed, more and more processors most likely contain the same RNA structure. The annealing mutation operator is very important for running the GA on long RNA sequences that are thousands of nucleotides in length as opposed to hundreds. Without the operator, no clear result can be obtained because of the large diversity of the population induced by the non-deterministic process. When a relative error that is computed from the weighted average of free energies obtains a value less than a specified uncertainty, the GA is said to have converged and thus is terminated. The structure that most processors possess is considered to be the solution structure that the GA has generated for the particular run. Since the GA is a non-deterministic process, statistics must be compiled for the stems contained in the GA's resultant structures for 20–50 runs.

The Boltzmann filter

During the initialization of the RNA structures, or during the mutation process, or the crossover process, stems are picked either randomly from the stem pool (initialization and mutation) or from the existing parent-structures (crossover), and an attempt is made to enter them into a new RNA structure. The stems picked must not geometrically conflict with the new RNA structure by containing overlapping nucleotides. Furthermore, a filtering process (Wu and Shapiro, 1999), based on the Boltzmann probability distribution in conjunction with Metropolis' stochastic relaxation algorithm may be applied. In this case, the acceptance or rejection of a stem into a new RNA structure is dependent on the change in free energy of the new RNA structure due to the addition of this stem.

The Boltzmann filter works as follows. In addition to passing the geometric conflict test, a stem making a structure more stable (decreasing the free energy of the structure) is accepted without any condition. However, a stem, making a structure less stable (increasing the free energy of the structure), is accepted or rejected based on the Boltzmann transition probability (the ratio of the Boltzmann probability of being in the new structure to the Boltzmann probability of being in the old structure). The decision is made by generating a random floating point number uniformly distributed in the interval $[0, 1)$ and comparing it with the Boltzmann transition probability. If the random number is less than the transition probability, the stem is accepted; otherwise, the stem is rejected. Therefore, for stems that do not geometrically conflict with a structure, their acceptance or rejection is not determined on an equal-probability basis. Thus, the Boltzmann filter may reduce the probability of the evolution of structures to structures which are stereochemically and thermodynamically locally unstable, and may facilitate the folding pathways of RNA sequences to be closer to those supported by the energy rules adopted from experiments. While searching for a local minima of free energies, the folding pathway guided by the Boltzmann filter can go uphill or downhill. Hence, the bias imposed by the energy rules may be reduced.

Indeed, statistically, the Boltzmann filter may significantly reduce the total number of positive stems (the GA's output stems), especially those occurring at lower rates. It also may increase the number of true-positive stems (stems output by the GA that match the phylogenetically or biochemically determined structure of the RNA sequence), and may substantially raise the appearance rates of these true-positive stems.

The random number generator

The non-deterministic GA assumes that the folding of an RNA sequence is a process driven by the stochastic characteristics of the molecule. In the evolution-like-process, random numbers, including floating-point numbers between $[0, 1)$, integers, and Boolean values, are needed. Random numbers are required, for example, while picking stems from the stem pool; while determining which structure is chosen to be the next structure on a processor in the selection process; while determining how many mutations are allowed by the annealing mutation operator; while determining how to distribute stems from the two parent structures to the two child structures in the crossover process; and while computing a probability to determine whether to accept or reject a stem in the Boltzmann filter. Therefore, the process of generating random numbers in the GA is very important.

An initial random number seed is either explicitly specified or is determined from a set of varying values obtained from some system parameters. If it is explicitly specified, the pattern for generating succeeding random integers and random floating-point numbers will remain the same. Hence, the subsequent performance and the result of the GA using these random integers and floating-point numbers can be fixed. In this way, it is very easy to keep track of a GA run and reproduce a GA run when necessary.

THE GA ON A SIMD MACHINE (MASPAR MP-2)

Computer system and configuration

The MasPar MP-2, on which our GA was originally developed, is a massively parallel SIMD (single instruction multiple data) architecture computer. It consists of a front end UNIX workstation and a Data Parallel Unit (DPU). The DPU consists of an Array Control Unit (ACU) and a processor array with 16 384 Processor Elements (PEs). Each PE has 64 kb of local memory for an aggregate total of 1 Gb. The processor array as a whole is wired as a two-dimensional mesh with toroidal wraparound. The ACU is a RISC processor, and does all bookkeeping such as opening a file, etc. and issues instructions serially (single instruction) to the 16 384 PEs which execute the instructions in parallel on each PEs local data (multiple data).

Communication between the PEs is accomplished by a form of message passing. This is facilitated by two interprocessor communication modes: X-Net and router. The X-Net is based upon a two-dimensional view of the processor array permitting uniform communications in any one of eight different directions from or to a given PE. The router, however, is based on a one-dimensional view of the processor array and is more general, permitting communication between arbitrary PEs. X-Net communication, when applicable, is faster than router communications. X-net message passing is used by the selection process since interprocessor communication occurs uniformly between a PE and its eight nearest neighbors (see below). The router, on the other hand, is used during stem pool initialization, and while picking stems from the pool. This is because the stem pool is distributed across the PEs' memory by layers and therefore random stem picking necessitates communication between a PE and any other PE.

Since the GA was originally developed on the architecture of the MasPar the population used in the GA was set at 16 384 RNA structures, which was uniformly distributed over the 16 384 processors—one processor containing one structure, as shown in Figure 1.

The selection process

The selection process occurs at every generation. On each processor, two parent RNA structures are selected from the nine available structures by using a ranked rule as discussed previously. The nine RNA structures include the structure on the processor itself and the eight structures around it. If either of the two parent structures is chosen from a processor other than that representing the center of the neighborhood, the structures must be communicated to the given processor. This is done by using the X-Net communication mechanism. Moreover, these actions take place on all 16 384 processors in parallel, making the selection process highly efficient.

The random number generator

At the very beginning of a GA run, a pre-specified or generated RNG seed, produces 16 384 random integers—one for each processor PE. Then, on each PE, this random integer acts as a seed to generate a series of uniformly distributed random floating-point numbers on the interval [0, 1) for subsequent usage.

THE GA ON A MIMD MACHINE (ORIGIN 2000 AND T3E)

Computer system

The SGI ORIGIN 2000 is a scalable SMP (symmetric multi-processor) computer that may consist of up to 512 physical processors. Unlike the MasPar, no specific processor on the ORIGIN does the bookkeeping such as opening a file, etc. Because the Origin 2000 is a parallel MIMD-architecture supercomputer, each processor executes its own instructions (multiple instructions) on its local data (multiple data) and uses special software instructions to synchronize execution across the physical processors when necessary.

The SGI ORIGIN 2000 has logically shared but physically distributed memory. To pass information between processors on the interconnect network, explicit calls to the shared memory data passing, 'shmem', library functions are used. This explicit message passing, as opposed to shared memory references, is used to achieve the extreme scalability desired for the GA as well as a relative ease of portability. In addition, the 'shmem' library is used instead of something like the MPI (message passing interface) library to achieve better performance for the very small data exchanges inherent in the algorithm.

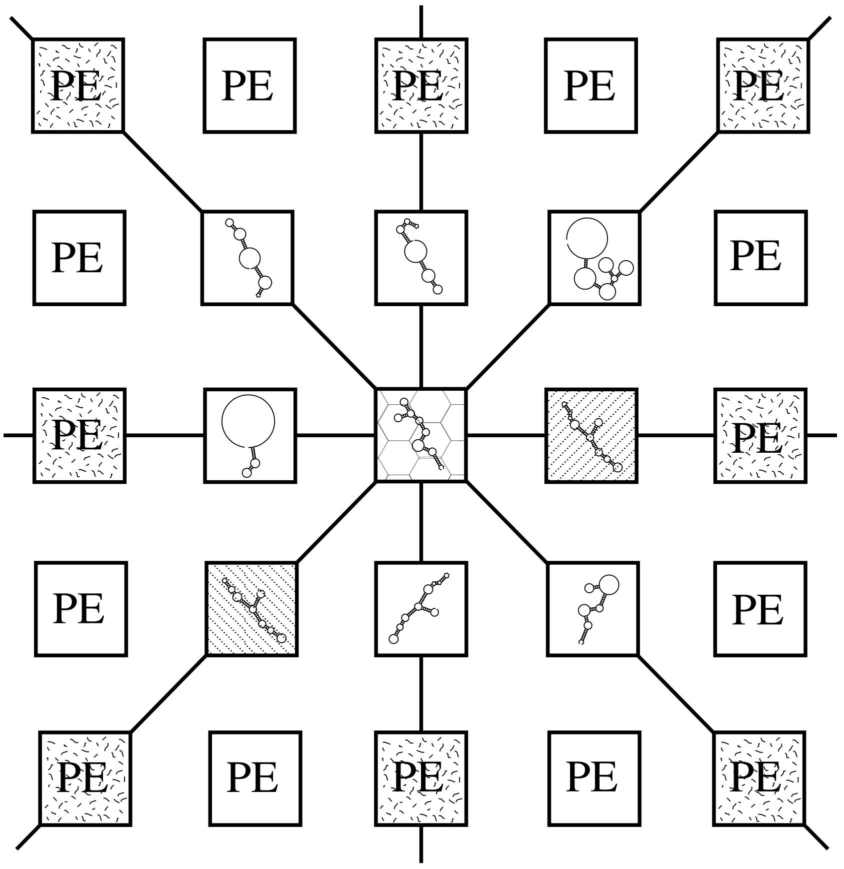

Fig. 1. The 16 384 PEs, on the MasPar MP-2 supercomputer, are wired as a two-dimensional mesh with toroidal wraparound. A population of 16 384 RNA structures in the GA is uniformly configured over the processors—one processor holds one structure. Around the above central processor are schematically drawn nine RNA structures, among which a selection process with respect to this processor takes place. Notice that these selection processes are occurring on all 16 384 processors in parallel at each generation of the GA.

Configuration

For performance purposes a one-to-one mapping of processes to physical processors is used. Thus, an N-process GA execution is allowed to use N physical processors on the machine. This number is much less than the population number, i.e. the number of RNA structures evolved in the GA. To mimic the GA paradigm that is implemented on the MasPar, a large number of RNA structures must be configured on this small number of physical processors. Hence, each physical processor's local memory must be divided into many locations, and each such location holds one RNA structure. Each of these memory locations and the associated code within each of the N processes to manipulate these locations on each physical processor is called a virtual processor. The data structure of the virtual processors on each physical processor is an array of RNA structures. For comparison, the MasPar may be viewed as having one virtual processor per physical processor.

To insure good performance for the algorithm, it is critical that the communication between the virtual processors be performed as efficiently as possible during operations such as selection. The least efficient communication is that which occurs between two virtual processors that are located on two different physical processors compared to those located on the same physical processor. Thus, a goal is to minimize the amount of communication that occurs between physical processors. This can be accomplished by minimizing the perimeter of the virtual processors on each physical processor. It is assumed that the total number of both the physical processors and the virtual processors are powers of 2 for digital simplicity. As a result, the best configuration must be as square as possible, that is, either a square or a rectangle in which the length of one side is to be exactly twice as long as the length of the other side. Thus, if 16 physical processors are used, they may have, for example, 2048, 4096, 8192, etc., virtual processors across them. An example is illustrated in Figure 2.

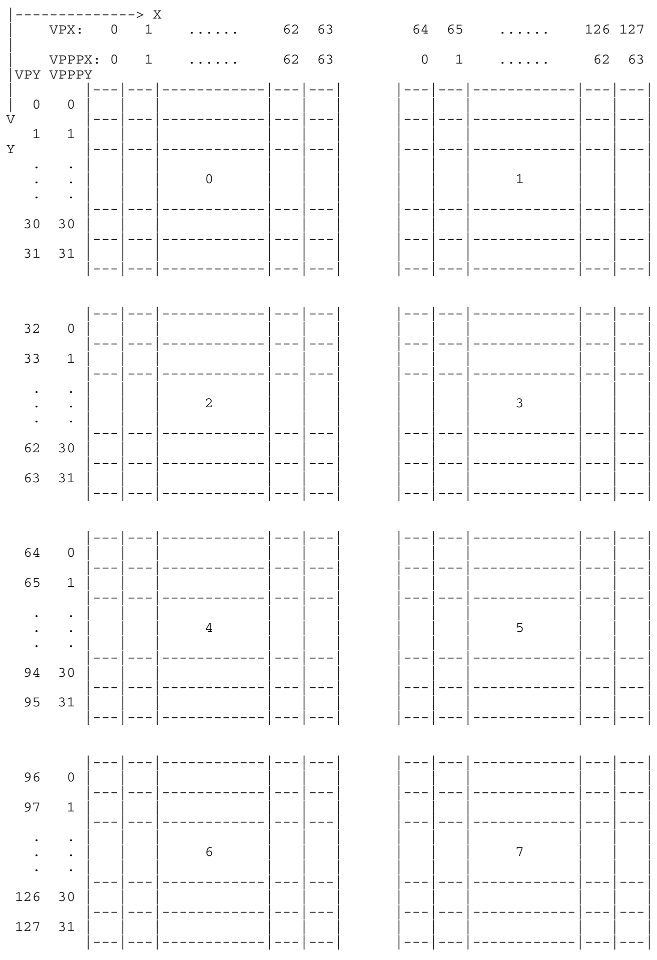
X
VPX: 0 1 ...... 62 63 64 65 ...... 126 127
VPPPX: 0 1 ...... 62 63 0 1 ...... 62 63
VPY VPPPY
V
Y
0
1
2
3
4
5
6
7

Fig. 2. The configuration of 16 384 virtual processors, (128 in the x-direction, VPX, and 128 in the y-direction, VPY) over eight physical processors. In this example, each physical processor possesses 2048 virtual processors (64 in the x-direction, VPPPX, and 32 the y-direction, VPPPY). It also illustrates the eight neighbors of each virtual processor, that may be located on other physical processors.

The selection process

In the selection process, each virtual processor communicates with its eight neighbors. As is indicated in Figure 2, some of these neighbors may be located on different physical processors. For example, in this figure, the eastern neighbor of the 63rd virtual processor on the zeroth physical processor is the zeroth virtual processor on the first physical processor; and its northeastern neighbor, due to toroidal wraparound, is the 1984th virtual processor on the 7th physical processor.

The GA on the CRAY T3E

The CRAY T3E, like the Origin 2000, is also a MIMD supercomputer with a distributed memory. Communication between different physical processors is also supported by the functions in the 'shmem' library. Essentially identical GA code can be run on both an Origin 2000 and a CRAY T3E. Given a specific RNG seed, a specific sequence, and the same physical and virtual processor configurations, the RNA folding results of the GA on the CRAY T3E and on the Origin 2000 are identical.

By comparison, at the time of the tests, the speed of a CPU of the CRAY T3E was 450 MHz while that of the Origin 2000 was 250 MHz. Given the cache friendly nature of the GA and identical code, the average run times for the GA on both machines using the 3′ UTR (untranslated region) of the Satellite Tobacco Necrosis Virus STNV-2 RNA with 619 nucleotides and three H-type pseudoknots (Danthinne *et al.*, 1991), as an example, were almost identical, namely about 1.3 s per generation. These runs utilized 16 physical processors and 16 384 virtual processors. This is about 3.0–3.5 times faster than the same sequence run on the 16 384 processor MasPar MP-2.

The T3E can be configured to contain more than 512 physical processors. Given this large capacity and the scalability of the GA it is possible to run the algorithm with populations in the millions. Computational experiments with these extremely high populations may lead to interesting results concerning structural accuracy and determinism (see below). Such experiments will be attempted shortly.

EFFECTS OF ALTERING THE POPULATION SIZE, PHYSICAL PROCESSOR NUMBER, AND VIRTUAL PROCESSOR NUMBER

As was mentioned previously, an important advantage of the MIMD implementation of the GA is the ability not only to obtain higher speeds by easily scaling a run to more physical processors, but also to vary the number of virtual processors and therefore the GAs population size. Two areas must be considered when examining the effects of such changes, namely, the impact of these variations on the efficiency of the algorithm and their effects on the accuracy of the algorithm. Both of these categories were studied for the GA's folding of the Potato Spindle Tuber Viroid (PSTVd) (Keese *et al.*, 1988) (359 nucleotides in length) and for the folding of the non-coding region of the polioviruses 1, 2, and 3 (Skinner *et al.*, 1989; Currey and Shapiro, 1997) (742 nucleotides in length). The trends shown below apply to other sequences tested but for the sake of brevity, results for PSTV and polio are presented in detail. The following employ measurements made on the SGI ORIGIN 2000, but virtually identical results should be expected on the CRAY T3E.

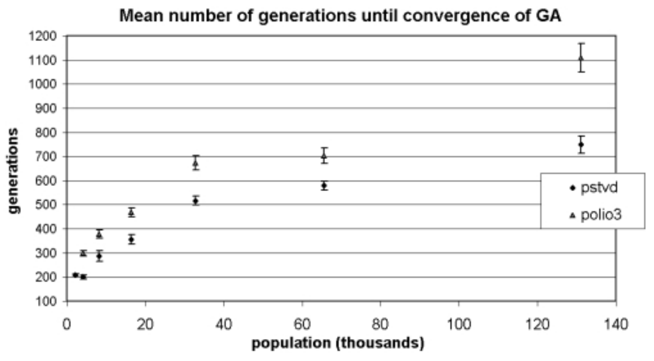

Fig. 3. Mean number of generations required until convergence of the GA as a function of population size. Mean was calculated over 20 runs. Error bars for data points are as indicated. Polio 3 is not shown for 2 K population (see the text for an explanation).

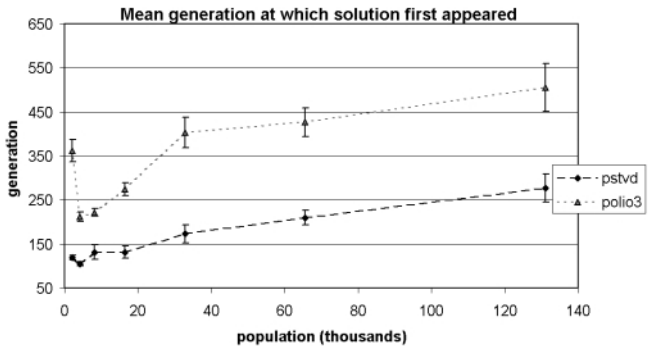

Fig. 4. Mean generation at which a solution first appeared for PSTVd and polio 3 as a function of population size. Mean was calculated over 20 runs of the GA. Error bar for data points are indicated.

Efficiency

Efficiency may be measured either in the real time necessary to complete the folding of a molecule or in the number of iterations of the algorithm needed to do so. Measurements of the latter are more significant since they are independent of the level of available computing resources, but both quantities were analyzed when considering the performance of the algorithm. As is evident from Figure 3, the average number of generations until convergence of the population (convergence as defined above) seems to increase with population size. Error bars in all the plots indicate the standard error of the mean plotted value for 20 samples at each population size. The generation of the first appearance within the population of the solution to which the GA eventually converged in each run is plotted in Figure 4. The increasing trend is still present. Note the deviation of values at population 2048 from the increasing trends in both plots. At population 2048, in fact, polio 3 never fully converged within 1500 generations, the chosen cutoff point. Perhaps because of the relatively high level of population-wide mutations allowed within the GA (Shapiro and Wu, 1996) for extremely low population sizes, populations $\leqslant 2048$ maintain a high level of diversity. Alterations to the minimum number of mutations allowed for small populations may improve the convergence properties, but may impact on the search of conformational space. In addition, the convergence properties of the two sequences are affected by the existence of metastable states. This will be discussed more fully later.

Measurements of the average wall-clock time required per generation of the GA were evaluated by increasing the number of virtual processors per physical processor (identified as VPPPT). This ratio was increased both by varying the number of virtual processors (VPT) from 2048 to 131 072 (2–128 K by powers of 2) while fixing the number of physical processors at a constant value of 8, and by varying the number of physical processors (PPT) from 1 to 16 (by powers of 2) while fixing the number of virtual processors constant at 16 384. The approaches, as evident from Figure 5, did not lead to identical results. When PPT is held constant, increasing VPPPT simply increases the load per physical processor, and the time per generation appears to scale nearly perfectly as $O(n)$ for n = VPPPT. That is, $R^2 = 0.9999$ (PSTVd) or 0.9995 (polio 3) for a linear regression with the y-intercept fixed at 0. When PPT is varied, however, communication and synchronization between physical processors is also altered and this affects the elapsed wall-clock time per generation, and time no longer scales as linearly ($R^2 = 0.9527$ for PSTVd and 0.9850 for polio 3). In addition, the apparent slope, *(time/generation)/(virtual processors/physical processor)* differs from that at a constant number of physical processors. Evidence of the significance of inter-processor issues can be seen in the fact that for a given ratio of virtual processors per physical processor, the time per generation is always greater for the larger number of physical processors.

Accuracy

A more significant aspect of the ability to vary the size of the GA population is the accuracy and fitness of the solutions as a function of that population size. For the purposes of this analysis, fitness was measured both from a thermodynamic standpoint and, more significantly, from a structural standpoint.

Free energy. Although it is known that biologically active structural configurations at times do not constitute a global minimum free-energy structure, biological evidence supports the theory that the change in free energy for the formation of the 'correct' structure of a particular

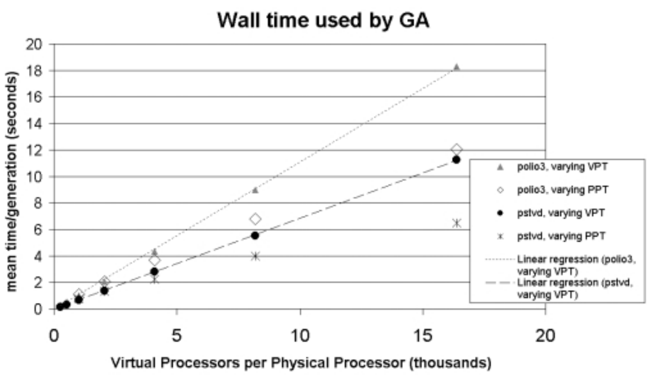

Fig. 5. Measurements of average wall-clock time per generation of the GA. Plots for 20 runs of PSTVd and polio 3 are shown. Almost perfect linear scaling is apparent for the case of varying the number of virtual processors per physical processor. When this is done by decreasing the number of physical processors in use, a slight non-linearity appears presumably reflecting the increased communication costs. It should be noted that the timing measurements were done with an under-subscribed system, i.e. fewer than 64 processes in total were running. Thus, it was not necessary to consider issues such as swapping.

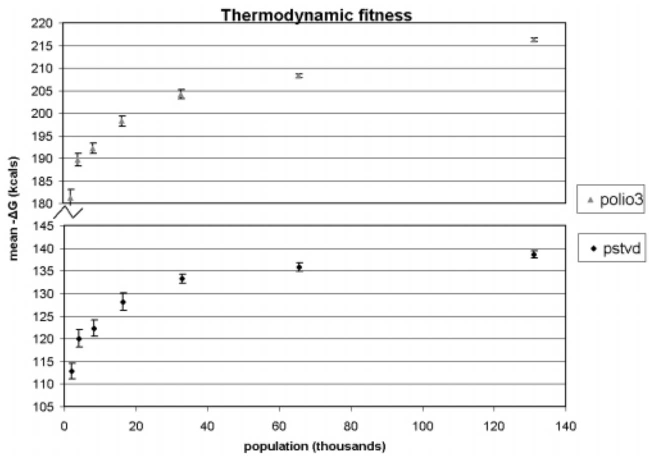

Fig. 6. Plots illustrating the improvements in thermodynamic stability as a function of population size for foldings of polio 3 (top) and PSTVd (bottom). The optimal computed free energy for PSTVd is −148.8 Kcal/mol and −239.3 Kcal/mol for polio 3. It should be noted however, that improved stability does not necessarily correspond to improved 'correctness'.

sequence will usually lie within the minimum 10% of all possible values of ΔG (free energy) for structures compatible with that sequence, as calculated under a specific energy rule set. In fact, many current approaches to RNA structure prediction are based entirely on determining an ensemble of possible structures with free energy nearest to the calculated minimum. Thus, it is reasonable to compare thermodynamic fitness of solutions generated at each population size when analyzing the performance of the algorithm. The plots of the average free energy of the GA convergence solutions versus population size shown in Figure 6 indicate that increasing the population of the GA increases the efficacy of the algorithm in locating a highly thermodynamically fit structure.

Structural accuracy. The primary measure of the comparative accuracy of the GA at varying population sizes is its ability to predict structures that are correct based on previous biochemical and biophysical experiments and on phylogenetic analyses. Our observation is that increasing the number of virtual processors utilized by the GA increases the similarity of the GAs solutions to proven structures. The GAs convergence solutions were compared to proven structures for PSTVd (Keese *et al.*, 1988) and poliovirus (Skinner *et al.*, 1989) both directly, calculating the percentage of known stems that were positively predicted by the GA, and via a structure/sequence-based fuzzy matching algorithm that calculates the frequency of occurrence, among an ensemble of solutions, of each of those known stems or of viable alternatives to known stems (Wu and Shapiro, 2001). The structural fitness measures plotted in Figure 7 were derived by calculating the average percentage occurrence of known correct stems (or their viable alternatives) among 20 GA solutions at each population size (fuzzy) and by calculating the percentage of known stems that passed the 55% frequency threshold in solutions (direct). Whereas folds of the UTR of polioviruses 1, 2 and 3 using 16 384 population elements failed to predict a number of known stems, and therefore required a consensus solution to be developed from the folds of all three sequences (Currey and Shapiro, 1997), folds of poliovirus 1 at 65 536 population elements correctly identified all but two of the known stems, and folds of poliovirus 3 at 65 536 population elements correctly identified every known stem but one in the Skinner model (Skinner *et al.*, 1989). For PSTVd (the entire native-state structure of which is known) both measures of fitness indicated a similar increase. At 131 072 population elements, folds of polio 3 deviate from these trends. For an explanation, see the section on determinism below.

A difficulty in the utilization of stochastic algorithms such as the GA for problems such as RNA folding simulation that constitute walks on fitness landscapes that may be highly rugged (Rook *et al.*, 1998), is the existence of local minima in which the population may become trapped. In the case of biomolecular structural configurations, high activation energy barriers may exist between two local minima on a particular landscape.

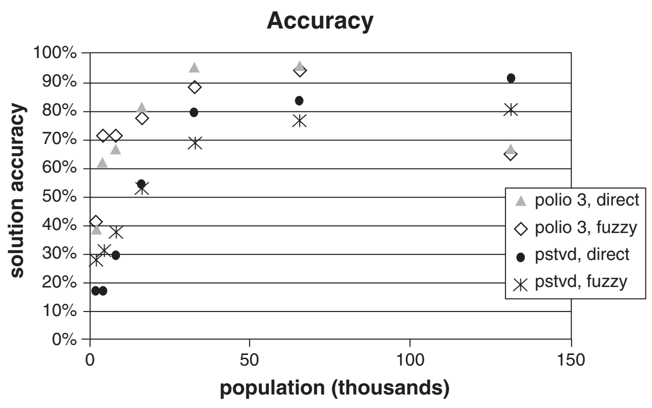

Fig. 7. Accuracy of solutions as a function of population size for PSTVd and polio 3. Accuracy was determined by two different methods, one involving a fuzzy match procedure and the other by a direct comparison.

In the case of PSTVd, such a situation is known from experimental evidence to exist; that is, while the native state of the molecule consists of a highly stable rod-like configuration, the folding pathway of the molecule includes a necessary transition through a highly bifurcated 'metastable' state (Riesner, 1990; Riesner *et al.*, 1979). Experiments indicate that the kinetic time constant for this highly favorable local minimum is relatively high and that the transition to the native-state is difficult (Riesner *et al.*, 1979). Simulating the folding of PSTVd with a larger population allows for a larger simultaneous sampling space and increases the probability that the kinetic barrier will be overcome. Thus, lower populations are more likely to remain trapped in this 'metastable' state, while higher populations are generally able to transition through the state to the native configuration. This fact hints at the possible utility of lower, 'less-effective' population sizes for identifying significant local minima that appear to exist as stages in the folding pathway as simulated at higher population sizes.

It should be noted that the circularity of PSTVd was simulated by embedding the left-hand terminal hairpin to eliminate the $5' - 3'$ gap, as in experiments detailed in Gultyaev *et al.* (1998). In addition, during the folding of PSTVd, two 'pseudostems,' i.e. stems composed of two actual single stems surrounding a 1×1 or 2×2 symmetric internal loop across which stabilizing coaxial stacking will occur, were included in the standard stem pool consisting of approximately 5000 (for PSTVd) single stems from which structural elements are randomly chosen for mutation into growing structures. The component stems of each pseudostem underwent the mutation process independently of one another and were treated as separate single stems once placed in a structure.

Determinism. Increase in the number of virtual processors operated on by the GA increased the determinism of the GA in predicting RNA structure. Such an increase in determinism was indicated by increased structural homology among the members of a set of solutions at a given population size as compared to that among members of a set of solutions at any lower population size. Specifically, at higher populations, more stems occurred with high frequencies among a given set of solutions while fewer stems occurred with low frequencies, and the average frequency of any stem within the ensemble of solutions increased. This can be observed visually in the difference in height between the stemtrace (Kasprzak and Shapiro, 1999) plots as seen in Figure 8, in which each horizontal band, color coded by frequency, represents a stem, and the height of each plot corresponds directly to the number of stems in the particular solution set represented by the plot. The deviation of PSTVd from these increasing trends at lower populations (<8 K) may be explained as the metastable local minimum, and an optimal population size for predicting this state appears to exist at 4 K, at which a peak exists in the measures of algorithmic determinism. As the population increases above 4 K, however, the algorithm appears to be able to explore the landscape beyond the local minimum and to move toward an additional peak in determinism.

Note that the thermodynamic fitness at 128 K (131 072) for polio 3 jumps significantly (by 8 kcal, a jump larger than all but that between 2 and 4 K) while the determinism suddenly falls off. This phenomenon is very similar to that which occurs with PSTVd at the 8 K population size, and, combined with the decrease in accuracy of the solution at 128 K, suggests strongly that, on the structure/fitness landscape defined by the particular energy rule set used, the 'correct' structure represents a local minimum only, and when the population is increased to 128 K, the GA once again (as with PSTVd) samples a large enough portion of the conformational space to explore the landscape beyond the correct local minimum and moves toward an 'incorrect' global minimum. Theoretically, continuing the increase in the population size would drive the GA to another peak in determinism (as with PSTVd) as it locates a more thermodynamically fit structure (although it would be less structurally correct). Perhaps measures of determinism that locate peaks such as the PSTVd peak at 4 K (where the metastable structure appears most strongly), the PSTVd peak that occurs as the linear state is approached, and the polio peak at 65 K (where all but one Skinner stem is located with very high frequency) would be useful tools in determining which GA solutions are most likely biologically functional. Again, this stresses the point that the biologically functional state may not be the optimal energy state, but, may be associated with states that are arrived at because of strong kinetic barriers.

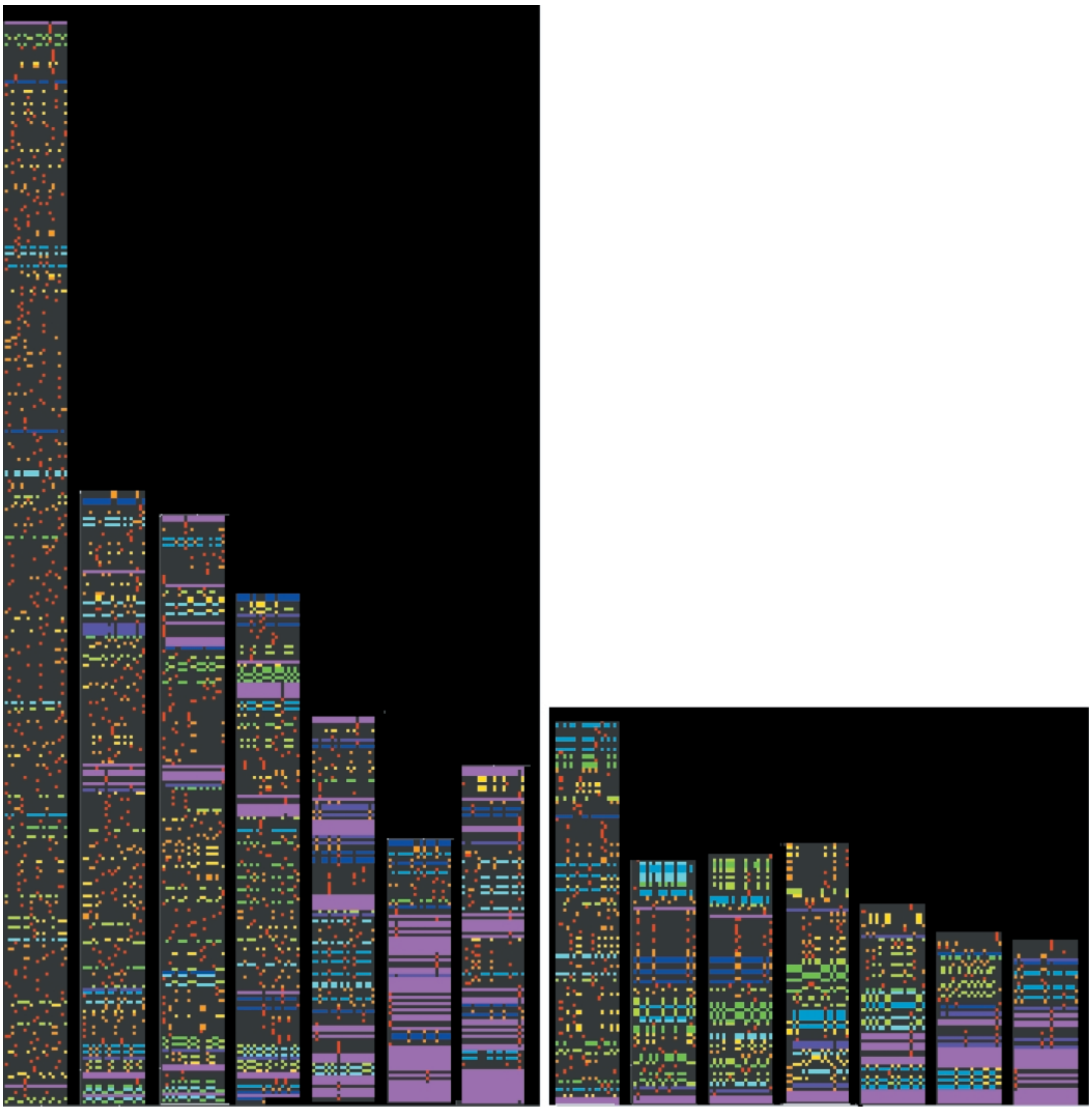

Fig. 8. Stemtrace plots showing the effects of population variation on 20 runs of polio 3 (left side) and PSTVd (right side). Each column represents 20 runs of the GA for populations of 2, 4, 8, 16, 32, 64 and 128 K elements. The height of each column is indicative of the relative diversity of results of the runs (taller columns indicating higher diversity). Constancy of color indicates the relative degree of determinism for the runs. Deviations from the downward trend in column heights can be explained by the population entering into metastable states as discussed in the text.

One final observation was made in regard to the effect of population variation on determinism. Lower population sizes can give an indication of what will occur at higher population sizes. For example, as the population is increased and the solution ensembles migrate from a consensus solution at one determinism peak to that at another, structures or structural elements contained in the higher population (and with higher fitness) will occur in increasing ratios to the entire solution ensemble size. Thus, a 'preview' of the solution that will appear at the next determinism peak may occur in increasing proportion as the population sizes increase.

The changing structure sets in solution ensembles as population size is increased often parallel the dominant subpopulations during a single run at a higher population size. For example, a 4 K run of PSTVd shows a determinism peak corresponding to a predicted metastable structure and a 128 K run shows a determinism peak indicating the linear PSTVd structure. Intermediate population sizes show an increasing proportion of linear structures to metastable structures as the population is increased. Similarly, in a single run that converges to the linear state, examining transient dominant subpopulations indicates that the metastable structure forms first and then transitions to the linear structure. Thus, comparing solution ensembles as the population size is increased can therefore give an approximation of the transitions that would occur within single runs at a higher population size, but with a substantially decreased quantity of required data analysis.

VISUALIZATION

A visualization component is also available with the MIMD version of the algorithm. This permits the color presentations of the fitness of the molecules in the processors, the number of pseudoknots found in a processor, as well as trace plots that permit the real time surveillance of the formation and/or disappearance of stems in structures. An additional aid is the interactive query capability that permits the display of energy and structural characteristics of any individual structure by a mouse click on the pixel representing that structure. Secondary structure displays can also be visualized in real time with the aid of STRUCTURELAB (Shapiro and Kasprzak, 1996).

CONCLUSION

An adaptation of the massively parallel GA for RNA folding to a scalable parallel paradigm has been presented. This scalable design permits the running of the algorithm on a single processor SGI OCTANE as well as a 512 processor SGI ORIGIN 2000 or a 2048 processor CRAY T3E. It conceptually follows the original design of the algorithm on a SIMD MasPar MP-2 supercomputer. Variations are possible with respect to the number of physical processors as well as the number of virtual processors (population size) involved in a computation. The transition from the SIMD paradigm to the MIMD paradigm mainly involved issues of processor (population) layout, as well as the impact of population variation on the results of runs.

In summary, the increase (relative to the SIMD implementation of the GA) in the number of virtual processors is enabled by the larger memory and faster processors on the MIMD machines. This in turn permits more efficiency per processor (less interprocessor communication per virtual processor). While an increase in the number of virtual processors per physical processor employed by the GA led to an apparent decrease in efficiency, this negative impact was matched and outweighed by an increase in both the determinism of the algorithm and the accuracy of the structural predictions it produced. In addition, when the algorithm becomes more highly deterministic, it then becomes possible to arrive at a statistically significant solution with relatively few runs of the algorithm, illustrating another benefit of the scalable MIMD architecture.

Varying the population size also has the added advantage of capturing the folding pathways of the molecule in metastable states, some of which have been shown to be significant for biological functionality. This was particularly illustrated in the folding of PSTVd. Further studies involving this phenomenon are now underway.

ACKNOWLEDGEMENTS

The authors thank Brian Furtaw and Lawrence Bertoldi of SGI Inc., for their efforts in the development of the GA visualizer; Wojciech Kasprzak from SAIC at NCI-FCRDC for his continued support in the development of STRUCTURELAB. We would also like to thank John Owens from the Laboratory of Experimental and Computational Biology, NCI for his help in making some of the figures.

REFERENCES

Abrahams,J.P., van den Berg,M., van Batenburg,E. and Pleij,C. (1990) Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Nucleic Acids Res.*, **18**, 3035–3044.

Currey,K.M. and Shapiro,B.A. (1997) Secondary structure computer prediction of the poliovirus 5′ non-coding region is improved by a genetic algorithm. *Comput. Appl. Biosci.*, **13**, 1–12.

Danthinne,X., Seurinck,J., van Montagu,M., Pleij,C.W.A. and van Emmelo,J. (1991) Structural similarities between the RNAs of two satellites of tobacco necrosis virus. *Virology*, **185**, 605–614.

Freier,S., Kierzek,R., Jaeger,J., Sugimoto,N., Caruthers,M., Neilson,T. and Turner,D. (1986) Improved free-energy parameters for predictions of RNA duplex stability. *Proc. Natl Acad. Sci. USA*, **83**, 9373–9377.

Goldberg,D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

Gultyaev,A.P., van Batenburg,F.H.D. and Pleij,C.W.A. (1995) The computer simulation of RNA folding pathways using a genetic algorithm. *J. Mol. Biol.*, **250**, 37–51.

Gultyaev,A.P., van Batenburg,F.H.D. and Pleij,C.W.A. (1998) Dynamic competition between alternative structures in viroid RNAs simulated by an RNA folding algorithm. *J. Mol. Biol.*, **276**, 43–55.

Holland,J.H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI.

Holland,J.H. (1992) *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence (Complex A)*. MIT Press, Cambridge, MA.

Kasprzak,W. and Shapiro,B.A. (1999) Stem trace: an interactive visual tool for RNA structure analysis. *Bioinformatics*, **15**, 16–31.

Keese,P., Visvader,J.E. and Symons,R.H. (1988) Sequence variability in plant viroid RNAs. In Domingo,E., J.,J.H. and Ahlquist,P. (eds), *RNA Genetics*. CRC Press, Boca Raton, FL, 3, pp. 71–98.

Matthews,D.H., Sabina,J., Zuker,M. and Turner,D.H. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, **288**, 911–940.

Miettinen,K., Neittaanmaki,P. and Periaux,J. (eds) (1999) *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming, and Industrial Applications*. Wiley, New York.

Press,W.H., Teukolsky,S.A., Vetterling,W.T. and Flannery,B.P. (1992) *Numerical Recipes in FORTRAN The Art of Scientific Computing*. 2nd edn, Cambridge University Press, New York, pp. 436–438.

Riesner,D. (1990) Structure of viroids and their replicative intermediates are thermodynamic domains also functional domains? *Sem. Virol.*, **1**, 83–99.

Riesner,D., Henco,K., Rokohl,U., Klotz,G., Kleinschmidt,A.K., Gross,H.J., Domdey,H., Jank,P. and Sanger,H.L. (1979) Structure and structure formation of viroids. *J. Mol. Biol.*, **133**, 85–115.

Rook,M.S., Treiber,D.K. and Williamson,J.R. (1998) Fast folding mutants of the tetrahymena group I ribozyme reveal a rugged folding energy landscape. *J. Mol. Biol.*, **281**, 609–620.

Shapiro,B.A. and Navetta,J. (1994) A massively parallel genetic algorithm for RNA secondary structure prediction. *J. Supercomput.*, **8**, 195–207.

Shapiro,B.A. and Kasprzak,W. (1996) Structurelab: a heterogeneous bioinformatics system for RNA structure analysis. *J. Mol. Graphics*, **14**, 194–205.

Shapiro,B.A. and Wu,J.C. (1996) An annealing mutation operator in the genetic algorithms for RNA folding. *Comput. Appl. Biosci.*, **12**, 171–180.

Shapiro,B.A. and Wu,J.C. (1997) Predicting RNA H-Type pseudoknots with the massively parallel genetic algorithm. *Comput. Appl. Biosci.*, **13**, 459–471.

Skinner,M.A., Racanaiello,V.R., Dunn,G., Cooper,J., Minor,P.D. and Almond,J.W. (1989) New model for the secondary structure of the $5'$ non-coding RNA of poliovirus is supported by biochemical and genetic data that also show that RNA secondary structure is important for neurovirulence. *J. Mol. Biol.*, **207**, 379–392.

Walter,A., Turner,D., Kim,J., Lyttle,M., Muller,P., Mathews,D. and Zuker,M. (1994) Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proc. Natl Acad. Sci. USA*, **91**, 9218–9222.

Wu,J.C. and Shapiro,B.A. (1999) A Boltzmann filter improves RNA folding pathway in a massively parallel genetic algorithm. *J. Biomol. Struct. Dyn.*, **17**, 581–595.

Wu,J.C. and Shapiro,B.A. (2001) Fuzzy matching of RNA stems. In preparation.