CHAPTER

3

Neural Networks, Adaptive Optimization, and RNA Secondary Structure Prediction

Evan W. Steeg

1 Introduction

The RNA secondary structure prediction problem (*2˚RNA*) is a critical one in molecular biology. Secondary structure can be determined directly by x-ray diffraction, but this is difficult, slow, and expensive. Moreover, it is currently impossible to crystallize most RNAs. Mathematical models for prediction have therefore been developed and these have led to serial (and some parallel) computer algorithms, but these too are expensive in terms of computation time. The general solution has asymptotic running time exponential in N (i.e., proportional to 2^N), where N is the length of the RNA sequence. Serial approximation algorithms which employ heuristics and make strong assumptions are significantly faster, on the order of N^3 or N^4, but their predictive success rates are low — often less than 40 percent — and even these algorithms can run for days when processing very long (thousands of bases) RNA sequences. Neural network algorithms that perform a multiple constraint satisfaction search using a massively parallel network of simple processors may provide accurate and very fast solutions.

This paper describes research into neural network algorithms for the pre-

diction of RNA secondary structure from knowledge of the primary structure. Some background on both the computer science and molecular biology aspects of the problem is provided, new methods are proposed, and the results of some simple, preliminary experiments are described [Steeg, 1989].

There are several goals motivating research into this area:

1. A fast and accurate algorithm for predicting RNA secondary structure is sought. It is hoped that an approach that formalizes the problem explicitly as an optimization problem and that incorporates a fine-grained parallelism and the machine learning ability of neural networks will lead to a good algorithm.

2. It is an interesting test of the ability of a neural net (and in particular the MFT neural net) to learn some of the key parameters of a natural structure-to-structure mapping, in this case RNA primary structure to secondary structure. Fast learning and good generalization are among the important goals in the learning experiments.

3. Finally, the work described may be thought of as an early testing ground for neural network and other parallel distributed processing (PDP) methods in molecular structure prediction — the *2˚RNA* problem is related to the more difficult problems of the prediction of protein secondary and tertiary structure.

1.2 Organization of the Chapter

In Section 2, the RNA secondary structure prediction problem is introduced, and the necessary mathematical definitions and physical and chemical terms are explained.

Section 3 defines the problem more formally in terms of a general class of search problems. The most commonly used search algorithms are discussed, and then a few of the most successful or important serial RNA secondary structure prediction algorithms are described in this context. This provides a brief historical summary of previous work within a unified formal framework.

Our methods are described in Section 4. We discuss neural networks and the particular class of Hopfield nets, Boltzmann Machines, and Mean Field Theory (MFT) networks used in our research. We then define the mapping of the *2˚RNA* problem onto the network, and explain the biochemical and physical assumptions implicit in our approach in terms of a simple graph theory problem. Finally, reference is made to some previous work in *protein* structure prediction with neural networks in order to illustrate the issue of representation of constraints.

Section 5 describes the results of the experiments. There is an analysis of the basic models employed in terms of speed of convergence, speed of learn-

ing, generalization abilities, accuracy of structure prediction, stability of solutions, and hardware/memory complexity.

Conclusions about the theory and experiments are offered in Section 6, along with some proposals for future work.

2. Secondary Structure of RNA: Its Importance and Methods of Determination

A molecule of RNA consists of a long chain of subunits, called ribonucleotides. Each ribonucleotide contains one of four possible bases: adenine, guanine, cytosine, or uracil (abbreviated as A,G,C,U respectively). It is this sequence of bases, known as the *primary structur*e of the RNA, that distinguishes one RNA from another.

Under normal physiological conditions, a ribonucleotide chain can bend back upon itself, and the bases can hydrogen-bond with one another, such that the molecule forms a coiled and looped structure. The pattern of hydrogen bonding is generally called the *secondary structure*, while the conformation of the molecule in 3-dimensional space is called the *tertiary structure.* The base-to-base interactions that form the RNA secondary structure are primarily of two kinds — hydrogen bonding between G and C and hydrogen bonding between A and U, as was first described by Watson and Crick in [1953]. (See Figure 1.) In fact, there is evidence of non-Watson-Crick basepairing in such nucleic acids as the tRNAs, but these are considered to derive from the tertiary structure produced by large regions of secondary structure containing Watson-Crick basepairing. For the sake of simplicity, such basepairing is mostly ignored in this paper.

Genetic information, the set of instructions that directs cell maintenance, growth, differentiation, and proliferation, is encoded in DNA molecules. RNA serves two biological purposes: It is the means by which information flows from DNA into the production of proteins, the catalysts and building blocks of cells; it also acts as a structural component of ribosomes and other complexes. It is the secondary structure, and the resulting tertiary structure, that determine how the RNA will interact and react with other cell components.

Work on the determination of RNA secondary structure has been carried out for decades by a number of research groups. The classical approach is direct observation of a molecule's secondary structure using X-ray crystallography. More indirect methods involve specific cleavage of the RNA by enzymes called ribonucleases. Much research has gone into the promising approach of computational *prediction* of secondary structure from knowledge of primary structure. The general method has been to search for configurations of maximum base-pairing or of minimum free energy.

There are two basic problems encountered in the prediction approach.

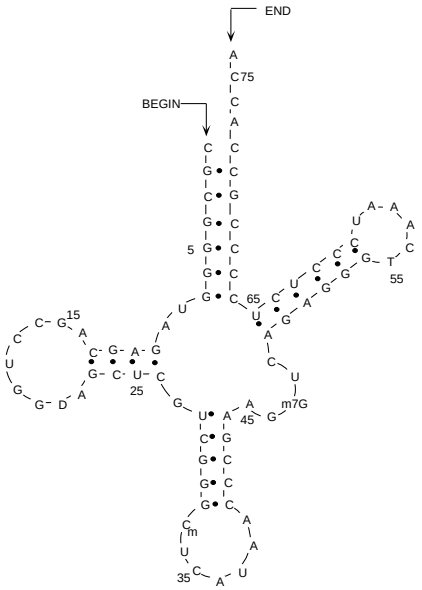

Figure 1: The figure is a 2-d representation of a tRNA molecule. The dots between bases (letters) represent basepairing. The numbers represent the numbering of bases in the sequence. (After [Sankoff et al. 1983]).

First is the need for accurate measures of the free energies of the various possible substructural components — of individual basepairs as well as stems, loops, and bulges. Second, the space of possible secondary structures for a given sequence is extremely large; a systematic search through all possible configurations for a minimum-energy structure can be prohibitively slow even on fast computers.

2.1 Structure and Free Energy—A Mathematical Model

We represent[1] an RNA molecule as a sequence S of symbols : $s_1\ s_2 \ldots s_n$, where s_i is one of G,C,A, or U. A subsequence of S may be called a "sequence" where no confusion will occur. A sequence or subsequence may also be called a "string".

Given a sequence S, we represent the secondary structure of S by the upper right triangular submatrix of an *n*-by-*n* matrix **A**. A_{ij} is 1 if *paired*(*i*, *j*), i.e., (for $i < j$), if the bases at positions i and j in the sequence are paired, and is 0 otherwise. (See Figure 3.) The secondary structure may then also be represented by a list **P** of pairs, where (*i*, *j*) is in **P** if and only if *paired*(*i*, *j*). A pairing itself will sometimes be referred to as $i \bullet j$.

The subsequence from s_i to s_j is written [**i**, **j**]. A subsequence is *proper* with respect to a secondary structure P if, for every paired element in the subsequence, its partner is also in the subsequence. If $i \bullet j$ is a pair and $i < r < j$ then we say $i \bullet j$ *surrounds* r. Likewise $i \bullet j$ *surrounds* $r \bullet s$ if it surrounds both r and s. (The rule against knots dictates that given $r \bullet s$, if $i \bullet j$ surrounds either r or s, then it surrounds both.) Subsequence [**i**,**j**] is closed with respect to a structure **P** if (i, j) in **P**. A pair $p \bullet q$ or an element r in proper string [**i**,**j**] is *accessible* in [**i**,**j**] if it is not surrounded by any pair in [**i**,**j**] except possibly $i \bullet j$. It is accessible from $i \bullet j$ if i and j are paired. A *cycle* **c** is a set consisting of a *closing pair* $i \bullet j$ and all pairs $p \bullet q$ and unpaired elements r accessible to it.

We can distinguish two kinds of constraints on the forming of an RNA secondary structure: *hard* and *soft* constraints (*constraints* and *costs* are the terms often used in optimization work). Hard constraints dictate that certain kinds of pairings cannot occur at all; soft constraints are those imposed by thermodynamics upon the classes of possible structures. Hard constraints determine which structures are "legal"; soft constraints determine which structures are *optimal*.. The hard constraints are:

1. (Watson-Crick pairing): If **P** contains (i, j) then s_i and s_j are either G and C, or C and G, or A and U, or U and A. (This may be easily extended to include the relatively rare GU pairings.)

2. There is no overlap of pairs. If **P** contains (i, j), then it cannot contain (i, k) if $k \neq j$ or (k, j) if $k \neq i$.

3. For all i, (i, i) cannot be in **P**.

4. Knots are not allowed: If $h < i < j < k$, then **P** cannot contain both (h, j) and (i, k).

5. No sharp loops are allowed: If **P** contains (i, j), then i and j are at least 4 bases apart.

The soft constraint on possible secondary structures **P** for **S** is simple: **S** will assume the secondary structure **P** that has *minimum free energy*.

A secondary structure **P** for **S** can be described in a natural and unique way as composed of substructures of four kinds: loops, bulges, stacked pairs (a stack of pairs is called a *stem*), and external single-stranded regions. The *cycles* of **P** are its loops, bulges, and stacked pairs. It is useful here to provide some definitions of cycles.

1. If **P** contains $i \bullet j$, $(i + 1) \bullet (j - 1)$, … $(i + h) \bullet (j - h)$, each of these pairs (except the last) is said to *stack* on the following pair. Two or more such consecutive pairs is called a *stacked pairs* cycle.

2. If **P** contains $i \bullet j$ but none of the surrounded elements $i + 1 \ldots j - 1$ are

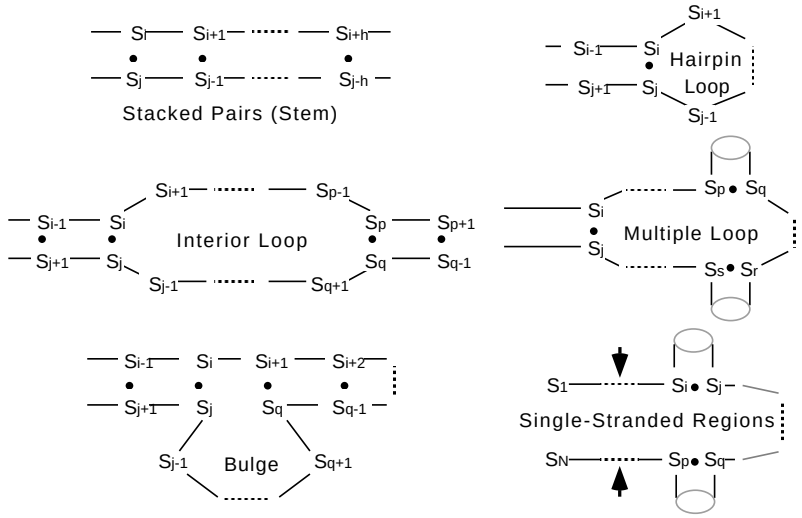

Figure 2: This figure illustrates the six basic kinds of RNA substructure. The indices S_i etc., represent base numbering, and the dots represent basepairing. (After [Sankoff et al. 1983]).

paired, then the cycle is a *hairpin loop*. (Many molecular biologists use "hairpin" to refer to a stem with a loop of size 0 or 1 at the end, i.e., a stem with virtually no loop. These structures are not allowed within our model, and it is not certain that such structures occur *in vivo* [Altona *et al.*, 1988].)

3. If $i + 1 < p < q < j - 1$ and **P** contains $i \bullet j$ and $p \bullet q$, but the elements between i and p are unpaired and the elements between q and j are unpaired, then the two unpaired regions constitute an *interior loop*.

4. If **P** contains $i \bullet j$ and $i \bullet j$ surrounds two or more pairs $p \bullet q, r \bullet s, \ldots$ which do not surround each other, then a *multiple loop* is formed.

5. If **P** contains $i \bullet j$ and $(i + 1) \bullet q$, and there are some unpaired elements between q and j, (or, symmetrically, if **P** contains $i \bullet j$ and $p \bullet (j - 1)$ and there are unpaired elements between i and p), then these unpaired elements form a *bulge*.

6. Let r be a sequence of elements in the sequence. If r is unpaired and there is no pair in **P** surrounding r, then we say r is in a *single-stranded region*.

In addition to these widely-accepted definitions of common substructure types, there is the interesting phenomenon of *pseudo-knots* [Waterman and Smith, 1978; Pleij, 1989]. Our current model makes no provision for the prediction of such anomalous structures.

The classical (Tinoco-Uhlenbeck) approach to specifying the free energy **E(P)** of a secondary structure rests on the hypothesis that the free energy is a sum of the free energy values of **P**'s cycles.

$$\mathbf{E(P)} = \sum_i \mathbf{E}(c_i)$$

Even if we accept as a working assumption the equation given above, we are left with the task of specifying free energy values for the primitive substructures. For this we must turn to empirical biochemistry.

2.2 The Tinoco-Uhlenbeck Theory

Much progress has been made on the problem of assigning free energy values to substructures. Although considerable theoretical work has been done, the most useful free energy data have been extrapolated from experiments on particular kinds of RNA. Much of the most important work has been carried out by Tinoco and Uhlenbeck [Tinoco, Uhlenbeck and Levine, 1971; Tinoco *et al*., 1973].

A reasonable first attempt at solving the *2˚RNA* problem would probably incorporate a detailed physical model of molecular structure. A mathematician might define a ball and stick model (balls for nucleotides, sticks for bonds) of an RNA molecule of length N, with $2N - 4$ variable angles and $(N - 1)(N - 2)/2$ potential energy functions for all pairwise hydrogen bond interactions. But the number of possible conformations is then an exponential function of the degrees of freedom. Such a model would prove computationally intractable for even the smallest RNA molecules.

Fortunately, Tinoco and others have simplified the problem, arguing that only the existence or nonexistence of particular hydrogen bonds matters; they have also provided empirical evidence that this simpler model has predictive power. Methods for relating free energy values to the size, shape, and base composition of secondary substructures, sometimes known as the "Tinoco Rules", can be viewed as a means of abstracting away from much of the complex thermodynamics of hydrogen bonding, Van der Waals forces, rotation of covalent bonds, and steric hindrance.

The Tinoco free energy data may be found in [Tinoco, Uhlenbeck and Levine, 1971; Tinoco *et al*., 1973]. Summarized below are the most important general ideas. It is important to qualify these ideas by noting that the *E(c)* free energy estimates for cycles are only estimates. The values cannot be determined with great accuracy, but they serve as useful, if sometimes crude, approximations of physical reality.

The most stable secondary structures, those having the lowest free energy, are long chains of stacked pairs. That is, a stem is the only kind of cycle which contributes negative free energy to the structure. The particular free energy value for a given stacked pair depends upon the two bases that are bonding, as well as a *local context*, i.e., the base composition of the closest

stacked pairs to its upper right and/or lower left in the matrix. Loops and bulges raise the free energy roughly in proportion to their size, that is, the number of elements that are left unpaired between the two elements that are paired. Beyond a certain size, loop and bulge energies seem to grow proportionally to the *log* of the unpaired length. Thus, a certain minimum number of stacked pairs is required to support a loop or bulge interior to the stacked pairs.

2.2.1 On The Minimal Free Energy Assumption. Molecular biologists commonly accept as an axiom that a full-length RNA molecule exists in its lowest energy thermodynamic state. After transcription, the molecule "breathes" in solution; that is, weak, non-covalent molecular interactions (Van der Waals forces, hydrogen bonds, etc.) form, break, and reform. Finally, the molecule settles into its lowest energy state — the secondary structure which the neural net algorithms described herein attempt to predict.

There are certain exceptions and caveats to the above axiom. The sequential generation, during transcription, of an RNA molecule can trap it in a local optimum. An example of this is attenuation, a regulatory mechanism for some bacterial operons (summarized in [Stryer, 1981]). In other cases, an RNA molecule will be forced into a particular configuration by its interaction with other molecules, usually proteins. However, the minimum free energy configuration provides a baseline with which the in vivo molecule can be compared. Moreover, the techniques described herein are able to accommodate such phenomena; particular substructures can be "clamped" while the rest of the representation of a molecule is folded using the free energy constraints. With these qualifications in mind, let us proceed with the description of the theory and techniques used by others and in this project, while following the classical simplifying assumption that minimal free energy is the determinant of secondary structure.

2.3 Serial Algorithms

The development of serial algorithms for the *2˚RNA* problem starts with the following equation. Let **P** be a secondary structure and suppose that [**i**, **j**] is proper. Consider the secondary structure P_{ij} on [**i**, **j**] induced by **P**. P_{ij} consists of all pairs from **P** whose elements belong to [**i**, **j**]. Suppose P'_{ij} is any other secondary structure on [**i**, **j**]. If we substitute P'_{ij} for P_{ij} in **P**, then the result is a valid secondary structure. That is,

$$P^* = (P - P_{ij}) \cup P'_{ij}$$

is a valid secondary structure. Then, from the above equation it follows that

$$E(P^*) = E(P) - E(P_{ij}) + E(P'_{ij})$$

Therefore, if **P** is optimal on [**1**, **N**] then P_{ij} is optimal on [**i**, **j**].

These facts serve as the mathematical basis for the serial algorithms for RNA structure prediction, and they define a basic recursive scheme which

most of the algorithms follow [Sankoff *et al*., 1983]. The reason that the algorithm in general is exponential is that every subsequence is broken into all of its possible cycles and single-stranded regions, and each of these defines a subsequence that must be analyzed and broken into its cycles, and so on. The intolerable running times can be avoided by using dynamic programming in order to compute the energy of each cycle only once, and by relying on strong assumptions about the free energy function. (Mainly, the assumption is that one need not calculate the energies of all subcycles in order to obtain the free energy of a cycle; thus, the expensive recursion is avoided.) The details of these algorithms differ and will be explored in more depth below.

The other class of algorithms in use sacrifices claims of optimality in order to obtain small polynomial running times. In these algorithms, the basic idea is to collect a set of pairs of complementary subsequences (subsequences which could form chains of stacked pairs). This is a simple $O(N^2)$ operation. Then some heuristic is employed to combine these into possible secondary structures and more heuristics are applied in order to choose "good" secondary structures.

3. Search Algorithms

3.1 *2˚RNA* as a Search Problem.

The development of the simplified Tinoco model of free energy in RNA secondary structure has allowed researchers to work with a high-level descriptive model that is independent of the low-level physics. In effect, the focus upon mere existence or non-existence of hydrogen-bonded basepairing reduces the problem to a discrete space that can be put into one-to-one correspondence with a planar graph. Global energy minimization is implemented as a discrete optimization problem. A discrete optimization problem can be formulated as a *search* problem — a search through the space of possible structures for the optimal structure.

Given the definitions of basepairing and secondary structures in RNA, it is easy to see that an RNA sequence may have unimaginably many secondary structures. Even if restricted to the chemically possible structures, as defined by the hard constraints given in Section Two, the number of secondary structures for a given RNA can be unmanageably high. To attempt to find an *optimal* secondary structure, then, is to perform a search for the one (or few) "very good structures" among the multitude of "pretty good" and "bad" ones. The problem then is to find ways to restrict the search space and/or to speed up the search through parallelism or by using finely tuned physical parameters to recognize quickly the thermodynamically best structures.

Looking at the *2˚RNA* as a search problem will enable us to make clear

and insightful comparisons of the many solution methods proposed. In this section, the RNA secondary structure problem is formally defined in terms standard to search problems in computer science.

3.1.1 Defining the Search Space. Let N, a natural number, be given.Let Σ (as in "s" for "sequence") be the set of RNA sequences of length N, that is,

$$\Sigma = \{G, C, A, U\}^N .$$

Let Π (as in "p" for "pairs") be the set of secondary structures for sequences of length N, that is,

$$\Pi = \{0, 1\}^{N^2}$$

Then if T (as in "Tinoco") is a function that assigns free energy values to secondary structures, our search space for the problem is, formally,

Given some T,
$\{(S, P), \mathsf{T}(P)\}$ where $S \in \Sigma$, $P \in \Pi$ and $\mathsf{T}: \Sigma \rightarrow \Re$

(Where a particular sequence S is considered, and no confusion should result, we will omit the S argument and use $\mathsf{T}(P)$.)

The problem then is:

Given a sequence $S \in \Sigma$, construct a secondary structure $P \in \Pi$ such that $\mathsf{T}(P) \leq \mathsf{T}(P')$ for all $P' \in \Pi$

3.2 Classes of Search Algorithm

We discuss here some standard classes of search algorithms, because their analysis sheds some light on the previous approaches to RNA structure prediction as well as on our new methods.

3.2.1 Optimal Algorithms and Exhaustive Search. In terms of the search space formulation given earlier, what does an exhaustive search algorithm do? Clearly, it considers *every point* **P** in the space of valid secondary structures for sequence **S**. For each such point, it calculates the free energy T(P). It then chooses the **P** with the lowest T value, and outputs **P**.

A simple recurrence relation defining the exhaustive search may be derived by considering the assumptions and definitions from Section 2:

The assumption about the additivity of free energy among cycles:

$$\mathbf{E}(P) = \Sigma_i \, \mathbf{E}(c_i)$$

Substitutivity of structures:

If **P** is optimal on [**1**, **N**] then $\mathbf{P_{ij}}$ is optimal on [**i**, **j**], if [**i**, **j**] is proper.

Taking these and performing some algebra, one may derive the following rule. The free energy value for the optimal secondary structure on [**i**, **j**] is given by

$$\mathsf{T}(i, j) = min\left\{0,\ C(i, j),\ \min_{i \leq h < j}[\mathsf{T}(i, h) + \mathsf{T}(h + i, j)]\right\}$$

where $C(i, j) = \min_{k \geq 1} \min_{\mathbf{c}}[E(S) + \Sigma_{(a_h,b_h)} C(a_h,b_h)]$, **c** representing *k*–cycles from *i* to *j*, is the optimal energy for [**i**, **j**] if we close it (i.e., if *i* and *j* are paired). If [**i**, **j**] is not *closable*, that is, if (*i*, *j*) is not a Watson-Crick pair or if $j - i < 4$, then $C(i, j)$ is taken to be infinity.

This analysis is due to Sankoff *et al*., and further details may be found in [Sankoff *et al*., 1983]).

A search algorithm simplistically defined in terms of such a recurrence relation is inherently inefficient. The algorithm recursively recomputes the same answer — the T value — for each subsequence many times. *Dynamic programming* methods work by filling in a table of subproblem values for subsequent repeated lookup, and offer dramatic time savings for problems which are composed of only a polynomial number of subproblems. However, the complexity concerns for the *2˚RNA* problem do not derive solely from the choice of full recursion versus dynamic programming. The complexity in the general algorithm stems from the number of possible cycles that have to be considered for each substring if multiple loops of arbitrarily high order are allowed. That is, the number of subproblems is exponential in the size of N. This fundamental complexity is not decreased if the algorithm looks up the energy for each possible cycle in a table instead of visiting each cycle substring recursively.

3.2.2 Approximation and Heuristic Algorithms. An exhaustive search for the best secondary structure is not feasible. Some restrictive assumptions have to be made, and some potential structures ruled out of consideration. Exactly which assumptions to make, and which classes of structures to ignore is a deep and difficult problem. Some heuristic choices have to be made, and in the end one has to settle for an approximation algorithm, an algorithm which is believed to "often" provide a "good" solution, if not always the optimal one.

3.2.3 Restrictions on the Search Space. In [Sankoff *et al*., 1983], the basic exhaustive search algorithm is shown to have time complexity of $O(r^2 (rN)^{2crN})$, where *r* is the proportion of unpaired bases in the sequence (often taken to be around 3/4), and *c* is a number providing a proportional limit on the order *k* of a loop within some [**i**, **j**] (for example, it is found that usually $k \leq (j - i)/7$, so $c = 1/7$). The simplest way to modify the exhaustive search algorithm described above in order to make it efficient is to restrict the number and kind of secondary structures to be considered, and the complexity of energy functions, while employing the same basic algorithmic structure.

Two obvious and useful restrictions to make are

1. to ignore multiple loops with order $k > k_0$ for some small k_0, and/or
2. to weaken the requirement that the algorithm work for arbitrary functions T(*s*) for any cycle *s*.

Before reviewing previous work on *2˚RNA* algorithms, it is instruc-

tive to consider a few more general search methods.

3.2.4 Local Search. A large and interesting class of algorithms is the class of local search algorithms. A local search algorithm has this basic form: Assume we have a measure **g(x)** of the "goodness" of a solution **x**. (Goodness may be defined analytically or in terms of heuristic and subjective criteria.)

1. Start with a random solution (or some easily-obtained solution).
2. Apply to the current solution some transformation L_i from a given set of transformations. The resulting solution becomes the new current solution.
3. Repeat until no transformation in the set improves the current solution.

The resulting solution may or may not be globally optimal. Of course, if the set of transformations includes every transformation that can take one solution into another, then we have essentially the exhaustive search method — and its high computational expense. The point of local search algorithms is to use a set of transformations that can be considered (hence the set should be small) and applied (so they must be efficient) in a small amount of time and space. If the set is small and the transformations easy, then the solutions that can be transformed one to another are considered "near", and hence the transformations are "local". The result of a local search is a *locally optimal* solution, also called simply a *local optimum*. The best solution overall is the (or a) *global optimum*. One can hope, if the transformation set is a good one, that the local optima found are at least very close to the global optimum (optima).

A Hopfield network, as discussed below, is a highly parallel neural net method of local search. The Boltzmann Machine and MFT networks represent (stochastic and deterministic, respectively) ways to change this into a more *global* search method.

3.2.5 Greedy Algorithms. The local search algorithms build whole solutions and then transform them repeatedly. A large number of heuristic algorithms build solutions incrementally. The greedy algorithms are in this class.

In a greedy algorithm, at any stage in the building of a solution it is the locally optimal step that is chosen. For example, a greedy Traveling Salesman algorithm would, having computed a path from C_1 through C_k, choose for the next section the city which is closest to C_k, though that choice might result in a suboptimal final path. A simplistic greedy algorithm for *2˚RNA* might calculate a structure for an ever larger segment of the RNA sequence. At step *k* it would have a secondary structure for [**1, k – 1**] and would grow the solution by finding the best way to force the *kth* base onto the current structure. In general, this would generate a rather poor solution; however, Martinez [1984] has a biological justification for a particular greedy approach, and his method achieves good results on some RNA sequences.

3.2.6 Monte Carlo Methods, Simulated Annealing. "Simulated annealing" is a method, adapted from statistical mechanics and inspired by annealing procedures in metallurgical processing, of employing stochastic functions in search procedures. Derived from theoretical work in [Metropolis *et al.*, 1953] and popularized in [Kirkpatrick, Gelatt and Vecchi, 1983] as an optimization technique, simulated annealing is one example of a "Monte Carlo" method of probabilistic numerical simulation.

The simulated annealing procedure is a way of doing a local search without becoming stuck in "bad" local minima. The method is simple:

1. Define a set of local transformations L_i (as in any local search) on the solution space.
2. Define a stochastic function Φ from solutions (states) and *temperature* values ($T \geq 0$) to transformations, such that
   - The probability $\pi_i(x, T)$ of picking transformation L_i, for some constant temperature T, when the current solution is x, varies directly with its "goodness" as a move, i.e.

     If $g(L_i(x)) > g(L_j(x))$ then $\pi_i(x, T) > \pi_j(x, T)$.
   - The degree to which the probability of a move depends on its goodness is higher as T is lowered. In other words, T is a measure of the *randomness* in the move-generation process. Φ is the move-generation function, which employs the probabilities π_i.
3. Choose a *temperature T*, $T > 0$.
4. Repeat while $T > 0$:
   (a) Choose a transformation and transform the current solution **x** by $\mathbf{x} := \Phi(\mathbf{x}, T)(\mathbf{x})$
   (b) If acceptable(x) then quit.
   (c)Decrement T

As Kirkpatrick, Gelatt and Vecchi [1983] have pointed out, the simulated annealing technique is a type of adaptive divide-and-conquer, with the basic features of a solution appearing at high temperatures, leaving the specific details to be determined at lower temperatures. There is a very natural way to map simulated annealing onto neural nets, and this, the Boltzmann Machine, is discussed in Section 4.4

3.3 Previous Work on *2˚RNA*

Historically, the systematic investigation into prediction of nucleic acid secondary structure has been marked by three major phases, each represented by a particular approach that dominated: 1) heuristic search over the secondary structure matrix or a large space of possible stacking regions, 2) dynamic programming approaches to building an optimal structure in a few passes, and 3) incorporation of auxiliary information and kinetic or evolu-

tionary assumptions into folding rules.

Pipas and McMahon [1975] designed an algorithm that performs a search in three passes. The first pass constructs a list of all possible stems of a certain size. The second pass scans this list for stacking regions that are *compatible* (meaning they form no knots and share no bases in common). In the final pass the algorithm performs an exhaustive search for the set of compatible stacking regions with the lowest free energy (using the Tinoco rules [Tinoco, Uhlenbeck and Levine, 1971]).

In terms of search spaces, this algorithm can be viewed as using two passes to construct a subset of Π — the subset consisting of those structures containing stacking regions of at least a certain size. The third pass then searches Π exhaustively for the P which minimizes $\mathsf{T}(P)$ for some given T.

The Studnicka algorithm [Studnicka *et al.*, 1978], like Pipas and McMahon's, begins by constructing a list of all the possible stacking regions. In the second stage, the algorithm enforces compatibility constraints between sets of regions. Instead of ignoring conflicting regions, as the Pipas-McMahon algorithm does, the Studnicka algorithm pares down the regions until compatibility is achieved for the now-smaller regions. The next pass combines the regions into large structures of order $k \leq 2$ (i.e., multiple loops not allowed). A final stage permits the user to combine these large structures into secondary structures of arbitrary complexity.

Such an algorithm can examine structures of high order k (the number of loops in a multiple loop) without the exponential time complexity seen in the general recursive algorithm. This is because the set of high-order structures that the algorithm can construct is severely restricted by the initial constraint of building structures with a set of existing stacking structures. For example, if building a structure from stems A, on [**i**, **j**], and B, on [**p**, **q**], one already rules out all combinations of structures over subsequences [**r**, **s**] where $i < r < j$, or $i < s < j$, or $p < r < q$, or $p < s < q$. An exponential explosion of possible structures is excised from the search space *a priori*.

Nussinov's group [Nussinov *et al.*, 1978; Nussinov and Jacobson, 1980] was among the first to apply dynamic programming to the *2˚RNA* problem. The Nussinov algorithms build an optimal secondary structure (subject to certain restrictive assumptions) in one pass. The algorithms are similar in structure to a basic dynamic programming version of the general recursive search algorithm [Sankoff *et al*., 1983], except that Nussinov made simplifying assumptions about structure and energy. The first version [Nussinov *et al*., 1978] ignores the destabilizing effects of loops, and simply attempts to maximize basepairing. The second version imposes a simple linear penalty on loop size.

All of the above display either unrealistic assumptions about the free energy of substructures, and/or have high time complexity ($O(N^5)$ for Studnicka) or space complexity.

3.3.1 Recent Advances with Dynamic Programming. With the interesting exceptions of the Martinez work and Major, *et al* [1991], the current "state of the art" in serial *2˚RNA* algorithms is a set of recent dynamic programming approaches.

Sankoff, Kruskal, Mainville, and Cedergren [Sankoff *et al*., 1983] have described algorithms that restrict the order of multiple loops to $k \leq 2$. In the case of arbitrary energy functions for loops, they report running times of $O(N^4)$. When the energy function for loops is restricted to the linear case, the result is an $O(N^3)$ algorithm. Zuker [Zuker and Stiegler, 1981], and Waterman and Smith [1978] have proposed similar algorithms within this range of time complexity.

In a theoretical computer science paper [1988], Eppstein, Galil, and Giancarlo describe an algorithm with running time $O(N^2 log^2 N)$ for the $k \leq 2$ case where the energy function for loops is assumed to be convex function of the number of exposed bases (bases accessible from the loop's closing pair). (See [Eppstein, Galil and Giancarlo, 1988] for the definitions of convex and concave.)

Several of the dynamic programming algorithms can be parallelized quite effectively, typically by using a wavefront method to trade $O(N)$ processors for an $O(N)$ factor in running time.

3.3.2 Martinez. Martinez [1984; 1988] takes a very different approach to minimizing free energy of molecules. Instead of building a structure using purely combinatorial methods, his *kinetics*-based method simulates the folding as it might actually occur in the molecule.

The Martinez folding rule, which defines the order in which parts of the final secondary structure are built in his algorithm, is simple:

Of all the remaining unformed stems which are compatible with those constituting the current structure, choose the one with the largest equilibrium constant (of association). This structure is the one whose formation is the most thermodynamically favored chemical reaction.

In search method terms, Martinez's method is a form of greedy algorithm. In particular, it has the property that it removes $(j–i)^3$ points from the search space of possible remaining structures at each step, where i, j are the beginning and end indices of the subsequence which supports the chosen stem. The time complexity of this algorithm is only $O(N^2)$.

The Martinez method is very promising. It has been shown to work on some medium-length (200-500 bases) sequences. The method is based on a fairly speculative but interesting evolutionary assumption, and is expected to be most successful in predicting the structures of RNAs whose secondary structure is essential to function (e.g., tRNAs and rRNAs).

3.4 The MFT Network Search for Optimal RNA Structures

In terms of the search model discussed above, our neural network method

may be described as a highly parallel distributed search, wherein each possible RNA secondary structure representation is distributed over many "processing units" (one unit for each possible base-pairing) and wherein several potential secondary structures for the input sequence are represented simultaneously. *Conflict* (w.r.t. constraint violation) between possible substructures is implemented by inhibitory connections between units in the respective substructures, and *support* (stem compatibility) is implemented by excitatory constraints.

Points in the *2˚RNA* search space are considered many at a time, as they *compete* during the MFT network relaxation process. The MFT relaxation algorithm is intended to avoid bad locally-optimal points in the space in favor of more globally-optimal solutions. The MFT learning algorithm is intended to make this search easier by refining the parameters of this competition over many trials with a training set of sequence and structure data. Connection weights constrain the dynamics of network relaxation, and can be seen as an implicit representation of *knowledge*, both analytic and heuristic, that aids the search process by pushing the network state transition process in particular directions and towards particular solutions in the solution space (π, $\{T(P)\}$).

4 Methods

This section describes our methods, and in particular it defines the neural network model of RNA secondary structure used in the experiments. The model is an example and an extension of the Mean Field Theory (MFT) machine originally proposed by Hopfield and Tank [1985] and later described and used by Peterson and Anderson [1987], among others. The MFT machine is a deterministic approximation of a Boltzmann Machine, which is a stochastic variant of a Hopfield net. The representation used is one wherein an RNA secondary structure matrix is mapped directly onto a Hopfield net, with every unit representing a basepairing.

In the first subsection, neural networks are introduced and some reasons behind the choice of neural networks, and specifically one-layer nets, are offered. Next, the MFT network and its intellectual roots (Boltzmann Machine and Hopfield network) are introduced. Then we define our mapping of the *2˚RNA* problem onto a Hopfield net architecture. Finally, some issues in the modelling of molecular structure are discussed with reference to our work as well as other work on neural networks and molecular structure prediction.

4.1 Neural Networks

Artificial neural networks are models of highly parallel and adaptive computation, based very loosely on current theories of brain structure and activity. There are many different neural net architectures and algorithms, but the

basic algorithm which all artificial neural nets share is the same. Assume a collection of simple processors ("units") and a high degree of connectivity, each connection having a *weight* associated with it. Each unit computes a weighted sum of its inputs (then may plug this sum into some nonlinear function), assumes a new level of activation, and sends an output signal to the units to which it is connected. In many of the models, the network settles into a stable global state under the influence of external input that represents an interpretation of the input or a function computed on the input. This settling process, called relaxation, performs a parallel search.

4.1.1 Neural Network Applications. Besides being the focus of *connectionist* research into models of brain function and cognition, neural networks have been applied with some success to optimization problems and function approximation. Optimization problems attacked with neural nets include the Traveling Salesman problem (TSP) and graph partitioning [Peterson and Soderberg, 1989], and process scheduling [Hellstrom and Kanal, 1990].

The *2°RNA* problem possesses several important features of the kind of problem at which neural network methods excel:

- Complexity: The problem has a large space of variables (search space).
- Redundancy: The set of reasonable, though not necessarily optimal, solutions is large, and many roughly equivalent solutions have variable values in common.
- Parallelism: Neural nets, of course, bring a parallel approach to any problem. Some problems seem inherently parallel (e.g., low-level vision), and the simultaneous consideration of and competition between possible solutions might well be the correct paradigm for the molecular folding prediction problems.
- Noise-tolerance: The problem may require the processing of very noisy or incomplete input data, and one would still like a reasonable answer.

In addition to these general neural net advantages, there are reasons for favoring the particular architectures chosen in this project. The stochastic nature of the simulated annealing procedure of the Boltzmann Machine might model well the thermodynamics of free energy minimization of an RNA molecule. The relationships among statistical mechanical models of spin glasses, neural networks, and biological macromolecules is an active research area [Stein, 1985; Anderson, 1988].

4.2 Architectures

A particular neural network architecture may be defined by specifying the following (list taken from [McClelland, Rumelhart and the PDP research group, 1986]):

- A set of processing units
- A set of activation states for the units
- An output function for each unit
- A pattern of connectivity among units
- A propagation rule for propagating patterns of activity through the network
- An activation rule for combining a unit's inputs with its activation level to produce a new activation level
- A learning rule whereby patterns of connectivity are modified by experience

In most neural net models of computation, the processing of an input vector, i.e., the solving of a particular instance of a problem, occurs through the network relaxation process. This is the process wherein the units in the network change their individual states in accordance with an update rule in order to maximize some global measure of "harmony" or minimize "energy" or constraint-violation.

4.3 Learning in Neural Networks

In most neural net models of computation, the processing of an input vector, i.e., the solving of a particular instance of a problem, occurs through the network relaxation process. The information needed for a neural net to solve a problem is largely stored in the connection weights between units. The process whereby the weights are modified in order to improve network performance is called, appropriately enough, *learning* or *training.*

There are several kinds of network learning procedures, but most fall into one of two broad classes: the *supervised* and the *unsupervised* learning procedures. The research described in this report concerns only supervised learning procedures.

In supervised learning, the connection weights between units are modified in order to reduce some measure of error — the error being a weighted difference between what the network outputs in response to a particular input and what one desires the network to produce in response to that input. Just as the relaxation process may be seen as a search through the space of possible network activation states for the state(s) with the lowest energy (lowest error, highest harmony, etc.), the learning process is the search through the weight space of the network for the set of connection weights that minimizes error in processing the training inputs. However, one desires that the learning procedure demonstrate some degree of *generalization.* That is, the weight modifications should enhance performance on whole classes of possible inputs represented by the trial patterns – not just on the trial patterns themselves.

The focus of this report is on *2˚RNA*, which is essentially a large optimization problem; but we require the networks also to learn a function, a mapping between RNA sequences and secondary structures, by successively refining estimates of a few network variables in order to reduce predictive error.

4.4 Hopfield Nets, Boltzmann Machines, and MFT Networks

Hopfield [1982] formalized the idea of a massively parallel and highly interconnected network performing a constraint satisfaction search. He introduced a cost function, termed *energy*, which is a measure of system-wide constraint violation. He then showed that the connection weights in a network encode locally minimum energy states, and that, using a suitable activation updating rule, these minimum energy states are exactly the stable states of the network. In particular, a unit u_k's contribution to the network's energy can be computed locally:

$$\Delta E_k = E_{(a_k=0)} - E_{(a_k=1)} = \left(\Sigma_i\, a_i\, w_{ki}\right).$$

where a_i is the activation level of the *ith* unit, and w_{ij} is the connection weight between the *ith* and *jth* units. The unit turns/remains on/off depending on which state lowers the network's energy.

Since the absolute value of the energy is bounded by the weights, the search is guaranteed to converge, if asynchronous node updating is used. However, like other gradient search methods, the procedure may only find a locally optimal solution. (This was not especially problematic in Hopfield's early work, because the networks were intended as a model for content-addressable memory. The "memorized" states were exactly the locally minimum states, and all the network was required to do was to *complete* one of the stored states, i.e., to fall into a local minimum.)

In order to design Hopfield-like nets that can escape local minima, researchers have adopted the *simulated annealing* technique, and thereby developed the Boltzmann Machine (BM) [Ackley, *et al* 1985] . In a BM, the following stochastic updating rule (or a variant thereof) is used:

> Each unit sets its activation state to 1, regardless of previous state, with probability
>
> $$P_k = 1/(1 + e^{-\Delta E_k/T})$$

where *T*, called *temperature*, is a measure of randomness in the system. At higher temperatures, high randomness permits state changes to be relatively independent of the energy, thus allowing the system to escape from local minima. At lower temperatures, configurations with low energy are more heavily favored. Thus, by using an annealing schedule whereby the temperature is gradually lowered or alternately raised and lowered, it is possible to avoid certain local minima and to find more globally optimal solutions. Actual performance in practice depends greatly on the topology of the energy sur-

face for the particular problem and encoding.

An alternative method for avoiding local minima and generally speeding up the search for some problems is to use *continuous activation* models. In [1985] Hopfield and Tank introduced a model in which activation levels (and hence outputs) take values from a fixed interval (in this case [0,1]) instead of the set {0, 1}. Such networks are based on a *Mean Field Theory* approximation to Boltzmann Machines and are thus called *MFT machines* or *MFT networks*. The MFT algorithm replaces the stochastic state transitions of the BM with a set of deterministic equations. The solutions to these equations, for each given temperature, represent *average* values of the corresponding quantities (correlations or co-occurrences between states of all units) computed from extensive and time-consuming sampling in the BM. The continuous output has the effect of smoothing the energy surface. In the binary model, the search procedure can be viewed as moving along the edges of an *M*-dimensional hypercube (where *M* is the number of units); whereas, in the continuous model, the search can move smoothly *within* the hypercube. In terms of an energy surface, a Boltzmann Machine performs stochastic hillclimbing (or hill-descending); MFT recasts it into deterministic motion through a smoother landscape. "Rather than scaling hills, one takes them away" [Peterson and Anderson, 1987]. Like the binary model, the continuous model is not guaranteed to find globally optimal configurations. Nevertheless, simulations of such a net which encoded the Travelling Salesman Problem did produce reasonably good solutions in a short period of time. In contrast, solutions found using the binary model proved to be only slightly better than random [Hopfield and Tank, 1985]. Peterson and Anderson [1987] have extended this approach and have tested the algorithm on several important optimization and learning problems, with favorable results.

The details of the derivation of the MFT model from the Boltzmann Machine model may be found in [Peterson and Anderson, 1987]. It turns out that the update rule for each of the continuously-valued units in a network is

$$V_i = \tanh\left(\sum_j \frac{w_{ji} V_j}{T}\right)$$

and the iterative algorithm becomes

$$V_i^{new} = \tanh\left(\sum_j \frac{w_{ji} V_j^{old}}{T}\right)$$

The above equations define the MFT relaxation scheme. The learning algorithm is equally straightforward. V_i is really an estimate of $\langle a_i \rangle$, the time average taken for the state of unit u_i. What is needed for learning is the equivalent, V_{ij}, of the correlations $\langle a_i a_j \rangle$ between connected units sampled in the Boltzmann Machine learning algorithm:

$$V_{ij} = \frac{1}{2}\left(tanh\left(\sum_k \frac{w_{jk}V_{ik}}{T} \right) + tanh\left(\sum_k \frac{w_{ik}V_{jk}}{T} \right) \right)$$

which reduces under certain conditions to

$$V_{ij} = V_i V_j$$

Let V_{ij}^{+} be the V_{ij} value for units u_i and u_j in a relaxation run wherein the input units are activated by an input vector v_{in} and the output units are "clamped" (forcibly set and maintained) to represent vector v_{out}; and let V_{ij}^{-} be the V_{ij} value when the machine runs with no clamping of the output units in response to the input v_{in}. Then the weight update (learning) rule is the following.

If $V_{ij}^{+} > V_{ij}^{-}$ then increment w_{ij}.
If $V_{ij}^{+} < V_{ij}^{-}$ then decrement w_{ij}.

The increment must be proportional to $V_{ij}^{+} - V_{ij}^{-}$ and the V_{ij} quantities are usually averaged across cases (learning samples).

4.5 Defining an MFT Machine Model of *2˚RNA*

4.5.1 The Underlying Architecture: A Hopfield Net The Hopfield net was chosen primarily because there is a very natural mapping of the *2˚RNA* problem onto this architecture. Basically, the network is a direct representation of an RNA secondary structure matrix. Each matrix position is represented by a single unit. An activation value of 1 means that the corresponding matrix element has a 1, and hence an hypothesis that the corresponding two bases in the RNA sequence are paired; a 0 in the same unit represents an hypothesis that the bases are not paired. A value between 0 and 1, if analog values are used, stands for a relative probability of the two bases being paired.

Symmetric connections are chosen because there must be signals (inhibitory or excitatory) between units and there are no privileged units — all the matrix positions are, *a priori*, equally valid, although particular combinations of them are invalid.

4.5.2 Representing the Problem: Deriving the Energy Function. Recall the Hopfield result [1982] that the equations of motion for a symmetrically connected network lead to convergence to a stable state. A stable state is one in which the outputs of all units remain constant. Under certain conditions (asynchronous node update according to the local updating rule), the network's stable states are the local minima of the quantity

$$E = -\frac{1}{2}\sum_{i=1}^{M}\sum_{j=1}^{M} w_{ij} a_i a_j - \sum_{i=1}^{M} a_i I_i$$

where E is a measure of *energy*. (This term is not to be confused with the free energy of an RNA molecule; we will attempt to make it clear which use is intended in each situation.) M is the number of units, a_i is the activation level of the ith unit, and w_{ij} is the connection weight between units i and j. I_i is a level of external input to the system, a *bias* for each unit.

In mapping the *2˚RNA* problem onto a Hopfield net, the network must be described by an energy function in which the lowest state corresponds to a legal and optimal RNA secondary structure. Low energy in the network must correspond to low free energy in the simulated RNA molecule.

Assume that the network has $M = N(N-1)/2$ units (where N is the length of the RNA molecule), so that, intuitively, the network represents the upper right triangular submatrix of the RNA secondary structure matrix. (See Figure 3.) Assume that each unit receives input telling it the composition of the two bases whose possible pairing it represents (e.g., it receives a constant

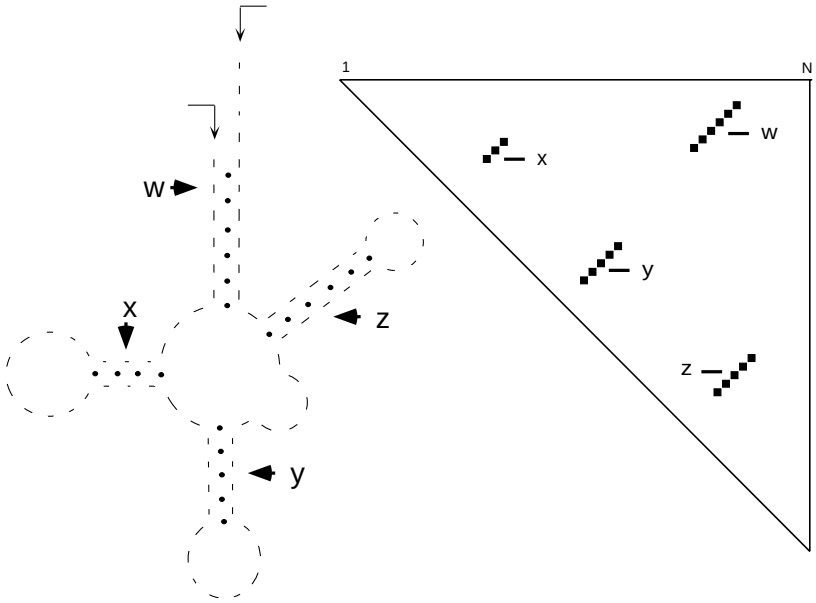

Figure 3: Mapping the problem onto a neural network: The diagram on the left represents a tRNA secondary structure. On the right is its representation on a secondary structure matrix, where each cell corresponds to a possible base pairing. The large diagonal in the upper right of the matrix represents the main stem; the other diagonals represent the smaller, subordinate stems.

signal of 00 01 for (G,C), etc.).

First, we want a term in the energy definition which tells us that the opti-

$$E = \alpha / 2 \sum_R \sum_i \sum_{j \neq i} a_{R_i} a_{R_j} + \beta / 2 \sum_i \sum_R \sum_{C \neq R} a_{R_i} a_{C_i} +$$
$$\gamma / 2 \sum_R^N \sum_C^N \sum_{z=0}^{C-N} \sum_{y=0}^{N-y} a_{RC} a_{R+z,C+y}$$

mal states are ones which enforce the hard constraints on RNA secondary structure. There should be only one unit active in any row, only one active in any column, and there should be no knotting. Consider the local minima of the following quantity:
(all summations are from 1 to N except where indicated), where R ranges over rows of the network, C over columns, and i and j count up to the length of the respective row or column.

Note that the first term is 0 iff each row R contains only one or fewer active units; the second term is 0 iff each column C contains only one or fewer active units; and the third term is 0 iff there are no knots. Therefore if a network incorporates this energy function, the stable states of the network favor representations of legal RNA secondary structures.

There remains the task of representing constraints on optimal secondary structures. Basically, what is wanted is this: Favor the formation of stacked pairs, with the precise negative energy contribution given by the Tinoco local context value for two adjacent stacked pairs. Impose a penalty for large loops and bulges, the penalty growing with the size. (In the experiments performed, the local context values were not represented, and their omission did not prevent good performance on the small tRNA sequences. However, it is expected that accurate predictions for longer molecules will require local context values.)

Add to the equation for E, the global network energy, the following terms:

$$\mu / 2 \sum_R^N \sum_C^N \sum_{z=0}^{C-R} \sum_{y=0}^{N-y} f_d(R, C, R = z, C - y) a_{RC} a_{R+z,C-y}$$

and

$$\sum_R^N \sum_C^N f_b(R, C)$$

where $f_d(i, j, k, l)$ is some function of the distance between two units (i, j) and (k, l), and $f_b(i, j)$ is a function of the indices of a unit which returns some value representing the tendency of a basepair to form. (For example, a high value is returned for indices representing a $G \bullet C$ pair, and low or zero value returned for a $C \bullet U$ pair.)

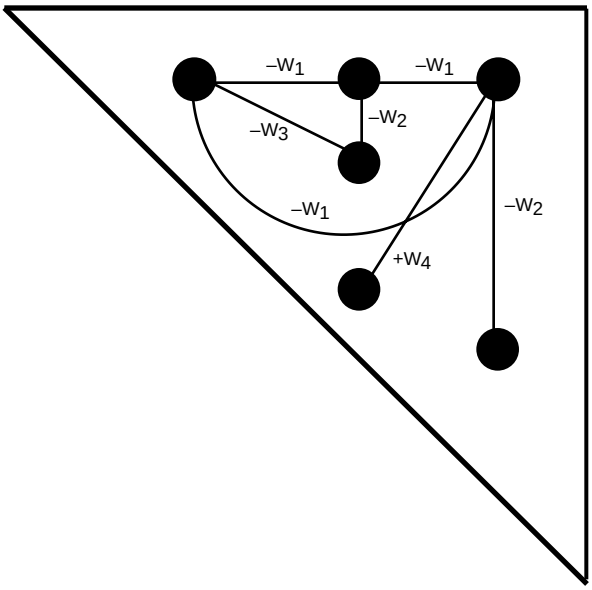

Figure 4: The structure of the basic network used for RNA secondary structure prediction. W_1 is the inhibitory signal between elements of a row; W_2 is the inhibitory signal between elements of a column; W_3 is the inhibitory signal that prevents knotting; W_4 is the excitatory signal between elements of possible secondary structures.

Setting the Connection Weights and Node Weights. Through the energy function definitions above and the earlier definition of energy minima for Hopfield nets, a set of connection weights is implicitly defined. The first three terms define inhibitory connections, with weights α,β,γ respectively, between elements in the same rows or columns or elements whose conjunctions form knots. The fourth term defines excitatory connections, with weight μ, between elements diagonal (in the "good" direction, i.e., not the knot direction) to each other. The fifth term defines the node weights, the bias for each unit. The bias of a unit influences the tendency of a unit to be active, irrespective of the influence of other units acting on it via connection weights. The bias is used in this model to represent the basic probability of a particular basepairing, independent of constraints imposed by other possible pairings. (Figure 4 illustrates the connection structure of the basic network.)

It is important to understand the limitations of this representation and how this relates to heuristic knowledge and to machine learning. The global parameters α,β,γ, μ (and through these, the connection weights w_{ij}), f_d, and f_b must embody a combination of information about both the problem and parameters particular to the computational architecture. Analysis of the problem and of Hopfield networks has led to this mapping of problem to architec-

ture, but it is not currently possible to derive analytically, from biochemical knowledge, precise optimal values for the global parameters. It is therefore necessary to make educated guesses at initial values (i.e., employ heuristic knowledge) and to adapt the weights over several learning trials in order to improve the representation and improve performance. It may be that adaptive networks also learn so well that they, in addition to improving their ability to represent existing biochemical knowledge, actually derive unknown or refine known physical parameters of the RNA folding process.

Input, Output, and Hidden Units. The structure of the network as defined for this project differs somewhat from most other Hopfield or Boltzmann applications. There are no separate input or output units, and no hidden units. The input sequence (binary representations of the bases G, C, A, U, and perhaps modified bases) is read into the rows and columns so that each unit receives input representing the identities of the two bases whose possible pair the unit represents — one represented by the row index and one by the column index. The bias term I_k for each unit u_k is then set accordingly. The connection weights are already determined and remain fixed during the processing of the particular sequence.

The output of the network is the set of activation levels of all the units, measured in analog (as numbers between 0 and 1) or binary, and preferably represented in a format like the RNA secondary structure matrix.

There are no hidden units in the models used thus far, although there are models under consideration which may use hidden units to represent the more complex higher-order relationships between distant substructures that will probably be needed for accurate prediction of very long RNA sequences.

4.6 Learning the Connection Weights and Learning RNA Structure

In this project, an effort was made to take as much advantage as possible of regularities in the problem (in the search space) in order to define an architecture wherein fast and useful learning is possible. If most work on predicting protein secondary structure [Qian and Sejnowski, 1988] seems to assume that all the important information in the sequence-to-structure mapping is *local*, the work described here assumes only that such information is either *local or uniform* across the net.

In particular, we hypothesize that there are a few (fewer than ten) important parameters, potentially obtainable through learning from examples, that determine the global sequence-to-structure mapping for RNA. Our current model employs α, β, γ, μ. These few quantities, once learned, can then be combined and replicated in some recursively specifiable way across the net in order to construct connection weights. The problem, then, is to define these few parameters as variables defining a learning space, and to devise a simple way to construct connection weights from these variables. Then the learning procedure would, instead of incrementing or decrementing each in-

dividual weight w_{ij} on each pass through the net, only update the corresponding global learning variable. After learning, the weights would then be constructed from the variables and used in the relaxation search. This is in fact what was done in this project, and the details are given below.

Such a learning scheme is beneficial in three ways. First, it probably provides for more accurate and robust learning, as it tends to ensure that key global parameters — and not just positional correspondences found in the learning samples — are what is learned. Second, the very small variable space (less than 10 instead of $O(N^4)$ or $O(N^2)$) that is optimized during learning makes for huge decreases in learning time. Third, and perhaps most interesting, it probably allows for some degree of scale-invariant learning. That is, it should be possible to achieve useful learning and structure prediction on a set of RNA sequences of different sizes, since the indices of particular units and connections do not have to match up exactly. Such scale-invariance over a very small range of sequence lengths is demonstrated in the experiments described below. In sum, the parameterization is a regularization scheme.

4.6.1 The Learning Variables and Learning Procedure. Following the derivation of the energy function given above in the introduction to the Hopfield net representation of *2˚RNA*, one sees the obvious candidates for global learning variables. Corresponding to the α, β, γ, and μ are RowInhibition, ColumnInhibition, KnotInhibition, DiagonalExcitation.

The MFT learning algorithm, modified for our global learning method, becomes (for a network of M units):

```
for each training iteration k
   for each learning example sequence s
      Phase+(s)
      Phase-(s)
      for i = 1 to M
         for j = i +1 to M
            δ := η((ViVj)+ – (ViVj)–)
            if (ui; uj are in the same row) then
               RowInhibition := RowInhibition – δ
            else if (ui, uj are in the same column) then
               ColumnInhibition := ColumnInhibition – δ
            else if (ui, uj form a knot) then
               KnotInhibition := KnotInhibition – δ
            otherwise
               DiagonalExcitation = DiagonalExcitation – δ
endfor; endfor; endfor; endfor
```

η is the *learning rate parameter*. $Phase^+(s)$ normally means to run the network relaxation process on input *s* with the output clamped (this provides the

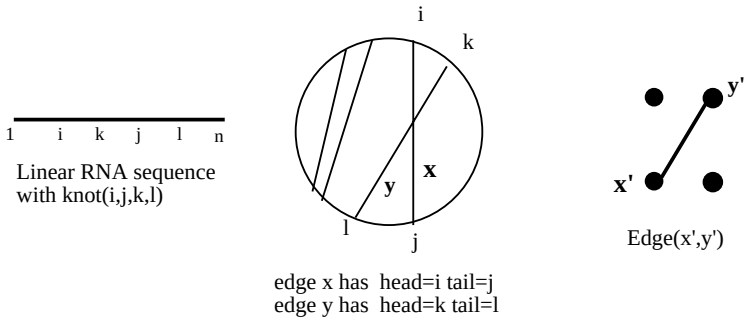

Figure 5: Graph theoretical interpretation of RNA secondary structure. On the left is a linear representation of the RNA sequence 1-n with a knot involving i,j,k and l. In the center is a circle graph (Referred to as G *in the text) derived from the linear sequence. On the right is an edge adjacency graph (G' in the text) derived from the circle graph.*

teaching input, the supervision in supervised learning). *Phase*$^-$(*s*) means run the network relaxation on input sequence *s* with the output unclamped. Similarly, the $+$,$-$ superscripts refer to the correlations gathered in the clamped and unclamped phases of learning. (In our described experiments the networks contain no hidden units, and therefore no distinct *Phase*$^+$*(s)* relaxation is needed. Instead, the $(V_iV_j)^+$ numbers are simply and quickly calculated from a vector representing the desired network outputs

4.6.2 Constructing the Weights from the Learning Variables. The mapping from learning variables to connection weights is quite straightforward, and follows from the definition of the system energy function *E* given in an earlier section.

The inhibitory weights used are exactly the corresponding global learning variables, e.g., (w_{ij}:= *KnotInhibition*) if the bases represented by units u_i, u_j form a knot. For the other, excitatory connections, the DiagonalExcitation is multiplied by the value returned by a distance function applied to *i* and *j*. The unit bias terms I_i are not affected by the learning procedure. They are determined by basepair identity as described earlier, along with a factor representing the distance ($i - j$), used to account for loop penalties.

4.7 Graph-Theoretic Analysis of the Methods

It is possible to explain the mapping of the *2˚RNA* problem onto neural networks in terms of the mathematical theory of circle graphs and edge-adjacency graphs, and thereby to relate the problem to a set of well-known combinatorics problems.

Consider an RNA sequence **S**: $s_1s_2 \ldots s_n$, where s_i is one of G, C, A, or U. A secondary structure **P** for **S** may be represented by a circle graph **G**,

wherein the *n* nodes are points around a circle and the edges (*i, j*) correspond to basepairs (*i, j*). (See Figure 5.) Edges that cross correspond to knots. All of the substructure types and constraints on structures defined in Section 2 may be described in terms of the circle graph. (Details may be found in [Nussinov and Jacobson, 1980; Takefuji *et al*., 1990].)

Consider next a graph **G'** with the number of nodes equal to the number of edges in **G**, and an edge (x', y') in **G'** for each pair of edges $x = (i, j)$, $y = (k, l)$ in **G** that intersect. **G'** is the edge-adjacency graph for **G**. (See Figure 5.)

It is clear from the above that the *2˚RNA* problem is closely related to the problem of finding the maximal independent set (the largest set of nodes such that none of them are connected to another node in the set) of the edge adjacency graph **G'** of a circle graph **G** for an RNA sequence S. Takefuji and coworkers [Takefuji *et al*., 1990] designed a neural network that finds locally optimal solutions of the general graph MIS problem, and pointed out the connection to predicting RNA structure.

While the similarity to the graph theoretic problem provides useful insights into the essential complexity of the *2˚RNA* problem, it is important to recognize the limits of this similarity. To solve the graph MIS problem, even exactly (an NP-complete task), corresponds to finding the largest set of possible basepairs such that none of them form knots. Clearly, this is not necessarily the optimal RNA secondary structure. Some attempt must be made to represent different stacking energies, the contribution of one stem to another in a multiple loop, and other real biochemical and physical constraints. Such attempts were made in our work, and machine learning was made central to our model in order to further refine the representation, but there is room for future work in exploring, possibly through graph theory, better problem representations for parallel networks.

4.8 Evaluating Models and Mappings

Having defined the particular kind of one-layer neural network model used, and the mapping of the *2˚RNA* problem onto the network model, it is instructive to review the nature of the constraints and the information flow within problem representations and some of the issues in representing in particular the RNA molecule and the molecular folding process.

4.8.1 Global and Local, First-Order and Higher-Order. When analyzing the constraints inherent in a problem, and before choosing a representation, one may consider three dimensions along which the adequacy of a representation (and hence a solution) will be judged. These three dimensions are *locality*, *order*, and *completeness of* information.

A representation for molecular structure prediction may capture simultaneously information on all parts of the molecule, or it may only represent a piece at a time. We say that a representation is *local* if it contains information

from only a section of *k* elements (bases, or amino acids) at any time. It is *global* if *k* = *length*(*molecule*). A representation is:

1. *first-order* if it captures interactions between primitive elements. In the *2˚RNA* problem, this means it captures base-pairing.
2. *second-order* if it captures interactions between interactions between elements. For example, the representation of stems, as resulting from interactions between basepairs, is second order.
3. *third-order* if it captures interactions between second-order objects. The representation of thermodynamic competition between possible RNA substructures is third-order.
4. *nth-order*, generally, if it captures interactions between (n-1)st-order objects and events.

Completeness refers to how much of the information, in a given local "window" and of a particular order, is captured.

Obviously, one ought to strive for a global, complete, and higher-order representation of a problem. Also obvious is that there is a trade-off involved: The more global and complete a representation is, and the higher its order, the higher is the computational complexity.

The serial algorithms for RNA secondary structure prediction are slow precisely because they compute with higher-order information in finding optimal substructures, and they do this serially over the entire molecule. A sequential computer generally can act globally only by covering a section at a time and iterating over the whole sequence, and this leads to long running times. Generally, the (non-neural) parallel algorithms do not differ drastically in their logical structure from the serial programs. Rather, they perform essentially the same steps but do it over $O(N)$ processors and thus achieve an $O(N)$ time savings.

The other work with neural nets on molecular structure, including Qian and Sejnowski's protein secondary structure prediction project [1988] (and see also [Bohr *et al*., 1988]) is based on local approaches. Sejnowski used feed-forward nets which captured only local ($k = 13$) information. The fairly low accuracy of local methods (of which neural net methods are the best) and the surprising fact that, as Qian and Sejnowski discovered, higher-order representations did not seem to raise the predictive accuracy, indicate that more global information must be captured, perhaps using global constraint-satisfaction [Friedrichs and Wolynes 1989] or pattern recognition [Greller *et al* 1991]. Research groups also use local representations and neural networks to predict local aspects of protein *tertiary* structure [see Bohr *et al*., 1990 and Holbrook, Muskal and Kim, this volume]. In RNA secondary structure, global interactions are probably more common than in protein secondary structure, and absolutely must be captured in the representation used by any com-

putational method.

The neural net models used herein are intended to be global and higher-order. The representation is explicitly higher-order, as the primitives (the processing units) stand for possible basepairs. Connection weights between the (2nd-order) units represent constraints on substructure types and competition between possible structures, which is third-order information. This premise implies a particular hardware complexity: $O(N^2)$ units and $O(N^3)$ or $O(N^4)$ connections, depending on how much of, and how globally, the third-order relationships are to be captured. (All of the 2nd-order information is represented, for the whole molecule, thus requiring $N(N-1)/2$ units.)

This hardware complexity is very expensive — prohibitively so, for very large RNA molecules. Thus one of the long-term goals of this project is to find ways, using separate processing stages, approximation methods, and forms of "time-sharing" on parallel hardware, to reduce this hardware cost significantly.

On the other hand, this neural net approach offers a large potential advantage in terms of complexity. It is believed that the costs of computation on this problem are "paid all at once", in the amount of parallel hardware, when using these methods. There are *no additional incremental orders of complexity added* when more general structures are handled. Relaxing an assumption about the energy contribution of loops, or about the complexity of multiple loops that are allowed, for example, can raise the runtime complexity of a serial algorithm from $O(N^3)$ to $O(N^6)$ or worse; handling the general case — all secondary structures are possible — mandates an exponential algorithm. However, to handle the general case with the simple model presented in this report requires only an $O(N^2)$-processor, fully-connected Hopfield net.

5. Experiments and Results

5.1 General Assumptions and Methodology

In the main set of experiments the basic problem representation outlined in Section Five, with $O(N^2)$ units, was used. The RNA sequences were limited to a standard length of 30 bases. In particular, the first thirty bases of each of 41 tRNAs were used (35 training, 5 test, and 1 for playing around with). These truncated sequences do not represent autonomous naturally-occurring molecules. However, in order to make the experiments more manageable, this limitation was considered necessary.

Because the 30-base sequences do not occur naturally except as components, there are no published secondary structures for them. Therefore, the secondary structures used in the training set of the supervised learning experiment were those determined by Zuker's serial program for RNA secondary structure prediction, described in [Zuker and Stiegler, 1981]. The Zuker pro-

gram is widely used and we assume it can be trusted to find the best structures for such very small sequences.

As mentioned in an earlier section, in these experiments we did not represent the local context effects on stacking energies. This omission was made for the sake of simplicity, and it did not prevent the nets from achieving good results on small RNAs (length 30, and, in a few preliminary experiments with our multiresolution model, lengths from 100 to 130). However, success on much longer sequences probably requires the local context information, or indeed a very different problem-to-network mapping.

5.1.1 Update Algorithm and Annealing Schedule. The network relaxation algorithm used in all experiments was the MFT algorithm described in Sections Four and Five. A sweep through the network consisted of $N(N-1)/2$ node updates; the choice of which node to update was made randomly each time. The updating was also asynchronous. The network was run for a given number of sweeps, unless stability (thermodynamic equilibrium) was achieved before reaching that number; a check for stability was made after each sweep.

In every experiment, the annealing schedule followed the same basic form: $T_{init} = 100$, $T_{final} = 1$, and $\Delta T = (T_{init} + 1)/n_{sweeps}$ for each sweep.

5.1.2 Dissimilarity and Predictive Accuracy. We define D, the structural dissimilarity between two RNA secondary structures (true or predicted) to be the proportion of possible basepairings on which they disagree:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{2[round(a_{ij}) - b_{ij}]^2}{N(N-1)}$$

where a_{ij} and b_{ij} are the units in row i and column j in the secondary structure matrix representation of the network and of the known secondary structure, respectively. The *round*() function rounds a number $n \in [0, 1]$ to 0 or 1.

Predictive accuracy of a method for a given RNA sequence is therefore measured as the number of correct positional matches between the secondary structure matrix (prediction) and the actual known secondary structure, as a percentage of the total number of cells in the matrix. That is, it is $100/D$.

There are many ways to look at the problem of dissimilarity among sequences and structures. It is true that tRNAs are known to share a particular overall shape — often called the "clover leaf" pattern — that is, a main stem and loop with three smaller subordinate stem/loop structures. However, it is very important to note that the forty tRNAs used in these experiments differ significantly both in terms of base composition and the particular places in the sequences where the stems occur. They also vary somewhat in length — from 107 to 131 bases.

The initial subsequences used in the first experiments display even more diversity than the full-length tRNAs. Their thermodynamically optimal sec-

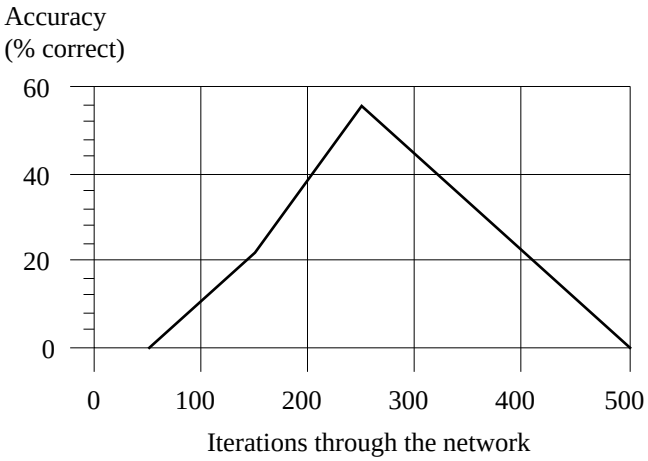

Figure 6: Predictive accuracy as a function of relaxation time, for the network with pre-set weights, with no learning

ondary structures, as determined by the Zuker program, do not all share the same overall structure. Some have one stem, others have two or three. The positions of the stems and loops also differ greatly. These general judgments concerning dissimilarity are valuable, but quantitative measures are also needed. The measurement of sequence homology is of course a standard tool in molecular biology. However, it is not especially relevant in these experiments, as sequences with a high degree of homology can produce very different secondary structures, and sequences with very similar structures can have a low degree of homology. Therefore, the positional matching measure of structure similarity described above, though not an ideal measure, is preferable in this case. The average structural dissimilarity among the training and test structures for the length-30 subsequences experiments was calculated to be 83 percent.

5.2 Pre-set Connection Weights, No Learning

The first experiment tested the predictive ability and examined the dynamics of a network with pre-set connection weights processing a particular 30-base sequence (initial subsequence of a tRNA from *Clostridium pasteurianum*).

The weights were set by hand, based on rough calculations of Hopfield-energy values for some desired network states corresponding to "legal" (though not actual) RNA secondary structures and the results of 20 trial-and-error runs of the simulator. The sequence was read into the network, the bias terms I_i initialized, and the network allowed to run the MFT algorithm for 500 sweeps through the net. Figure 6 is a plot of the number of sweeps against predictive accuracy for the simulation described.

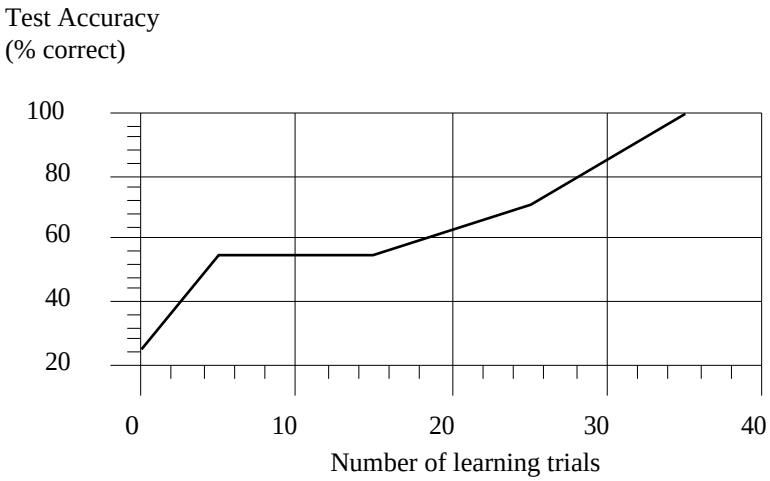

Figure 7: Predictive accuracy as a function of number of learning passes, 500 iterations on learning trials, 200 iterations on test.

Clearly, the preset network made some progress from a random state towards reasonable guesses at secondary structure. However, the predictive accuracy remained low (less than 60%), and the best solution obtained was not stable: after 250 iterations the network moved toward less optimal structures. The next section shows the degree to which learning improved the accuracy and stability of structure prediction.

5.3 Learning the Connection Weights

The second experiment was intended to test the capability of the basic MFT network to learn to predict RNA secondary structure, and to use the learned information to produce faster, more accurate, and more stable predictions than the network whose weights were preset.

5.3.1 The Learning Algorithms. The MFT learning algorithm described in Sections 4 and 5, modified to fit the small learning space approach, was used in the learning experiments. A set of 35 sequence/structure samples was used for supervised learning. Each sample was run through the "plus" and "minus" (clamped and unclamped) phases once. Thus there were only 35 learning passes in each experiment. The global learning variables, from which the connection weights were derived, were updated after each pass. A pass using the *test sequence*, the 30-base segment from the *C. pasteurianum* tRNA, was tried at various intervals throughout the learning experiment to test the abilities of the changing network. Note that a) the test sequence was not among the training sequences, and b) the learning variables and weights were not changed after a pass using the test sequence.

In the learning experiments, the network's initial weights were set to a

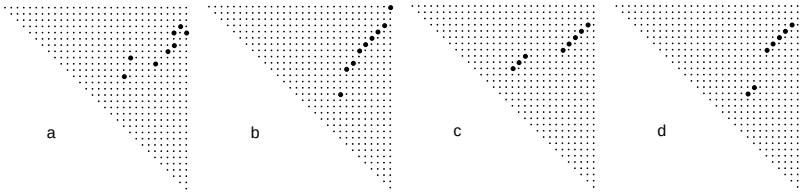

Figure 8. Network after 0; 5; 15 and 25; and 35 Learning Passes. (a) represents the net after 0 learning sweeps, (b) is 5 sweeps, (c) shows the net after both 15 and 25 sweeps, and (d) shows the output at the end, after 35 sweeps.

configuration that achieved better than random performance (15%) on a sequence that was neither a test nor a training sequence.

5.3.2 Results. Two experiments were performed, on five different test sequences: initial subsequences from tRNAs from *C. pasteurianum*, *Bacillus megaterium*, *Streptococcus faecalis*, *Mycoplasma capricolum*, and *Mycoplasma mycoides capri*. The experiments used the same initial weight configuration, the same learning rate, and the same annealing schedule. The results were very similar, so we describe the results for the same *C. pasteurianum* tRNA fragment used earlier.

In each experiment 200 sweeps were made through each annealing (minus phase) of the training sample. Only 100 sweeps were made through the network during the relaxation on the test sequence. The next diagram (Figure 7) is a plot of the accuracy of performance of the network after 0; 5; 15; 25; and 35 learning passes. Figure 8 displays network activation (output) diagrams for these snapshot points. Figure 8d is the correct structure (as predicted by the Zuker program). It is clear that the network did improve its performance drastically over the learning period. Also, its performance became quite good: It achieved perfect predictive accuracy after 35 learning passes. It is interesting also to note the steep improvement during the first 5 passes.

In a recent paper, Takefuji *et al.* [1990] report similar results with their non-learning neural net method for solving near-MIS of circle graphs. Their method found good structures (in 2 cases, structures more stable than those predicted by serial methods) for three RNA sequences of lengths 38; 59; and 359 bases.

6. Conclusions and Future Work

6.1 Conclusions

This report presents a new class of methods for predicting RNA secondary structures from sequences. The methods use artificial neural networks based

on the Boltzmann Machine model and the Mean Field Theory deterministic approximations to the Boltzmann Machine.

The methodology described in this paper is:

- Formulate the *2˚RNA* problem as problem of optimization and search.
- Map the search problem onto a Hopfield network (typically a method of highly parallel local search). This mapping can be understood in terms of some simple combinatorics and graph theory.
- Avoid the problem of Hopfield local minima by using the MFT network training and relaxation algorithms, which implement a deterministic and "analog" (continuous) version of the stochastic Boltzmann Machine.
- Use the MFT learning algorithm and a highly-structured connection weight-sharing scheme to adjust a small set of parameters in the network's representation of the sequence-to-structure mapping, in order to improve performance over time.

After an introduction to the problem and a review and analysis of search techniques and neural network algorithms, the results of some preliminary experiments were reported.

Experiments were performed on a set of 35 tRNA sequences and fragments thereof, using neural network simulators implemented on serial computers. Conclusions drawn from the experiments include:

- At least on small RNA sequences, a properly-configured MFT neural network can learn a few global parameters from which connection weights can be derived that enable the accurate prediction of secondary structures. Related work on neural network methods without learning [Takefuji *et al*., 1990] demonstrates that these methods may also work on moderate-sized RNAs.
- The learning can be very efficient. 35 learning examples, each passed through the network once, sufficed in the experiments. On each learning pass, fewer than 400 iterations through the network always sufficed. This fast and powerful learning is made possible by a method that constructs a very small learning space (4 or 5 variables) for an RNA secondary structure network. This method also enables a degree of scale-invariant learning.
- With the learned weights, the networks were able to converge quickly to exactly accurate solutions on the test sequences. 200 or fewer iterations were required.
- A degree of generalization is possible. The test sequences were not part of any of the learning sample sets, and in fact differed significantly from many of the learning samples.

6.2 Future Work

6.2.1 Interpreting the Network Dynamics; Representing Molecular Dynamics. The dynamics of MFT networks configured for the *2˚RNA* problem are interesting enough to warrant further study. In particular, one would like to discover whether the dynamics of a relaxation search implement a particular, realistic *folding pathway* for an RNA molecule. The networks *seem* to mirror Martinez's kinetically-driven sequential cooperativity search: large stems form first and constrain the set of possible small stems; and at each stage the most thermodynamically favored stems appear. It would be interesting to investigate whether such behavior is inherent in the MFT approach, or whether it is an artifact of particular sequences or particular parameter settings.

In general, the currently prevailing models for molecular structure prediction with neural networks share an essentially *static* approach to the problem. In other words, a mapping is sought between sequence and final secondary or tertiary structure, making no use of intermediate stages of folding. Future research should explore ways to integrate kinetic effects and dynamical processes.

6.2.2 Interpreting Continuous Activation Levels., Throughout this project, we chose to round the activation states of the units to 0 or 1 when reading them as output. The MFT relaxation algorithm also tends to drive the activation values very close to their extrema. Thus all discussion was in terms of elements being paired or unpaired. However, it has been pointed out that in many cases network activation levels, read as real numbers, can be interpreted in terms of probabilities. Perhaps the network models could be used, like several other RNA structure prediction programs, to predict ensembles of near-optimal structures.

6.2.3 Bringing More Knowledge to Bear on the Problem. The simple representations of ribnucleotides (or amino acids) used typically in computer programs are very limited. A ribonucleotide is in reality more than a symbol; it is a molecule, with physical structure and several important physical and chemical properties that can be measured. Good simulation/prediction programs should probably represent these structural units as vectors, linear combinations of basis vectors representing physical and chemical characteristics, like molecular weight, steric configuration, charge, magnetic dipole, and hydrophobicity [Nakai, *et al* 1988; Hunter, 1992].

Phylogeny is another source of information. How can knowledge of the structure and sequence of one molecule be used to predict the structure of another molecule whose sequence is homologous or whose function is the same? Major *et al* [1991] describe a system for RNA secondary and tertiary structure prediction that combines geometric, energetic, phylogenetic and functional information within a symbolic constraint satisfaction framework.

Their results are impressive and lead one to wonder whether the addition of statistical induction (machine learning) methods might refine their existing knowledge base and produce even better results.

6.2.4 Tests on Longer RNA Sequences. Obviously, the methods described herein should be tested on larger RNAs, despite the space complexity problems afflicting the current versions of the basic algorithm (see below). The adaptive optimization approach seems to perform very well on small pieces of RNA when we have a reasonably large set of representative training data. However, a very similar but non-adaptive method [Takefuji, *et al*, 1990] has also apparently been made to work well on some small RNAs. I do not believe that neural networks with weights set "by hand", or graph-planarization algorithms that ignore thermodynamic subtleties, will scale up well to larger structure prediction problems. Larger scale experiments with adaptive algorithms will help us to determine whether a sufficient number of RNA sequences and solved structures are available to allow machine learning methods to refine and augment the very incomplete communal knowledge --- theoretical and empirical --- on the thermodynamics, kinetics, and molecular evolutionary history of RNA folding.

6.2.5 Coarse-Grained and Multiresolution Search. The representation described in Section 4 is useful because it is based on an obvious mapping of the RNA secondary structure matrix onto a neural net model, and is therefore easily understood. However, it is not very efficient in its use of hardware (simulated hardware, in the near term). The representation employs $O(N^2)$ processing units. (In fact, it employs exactly $N(N–1)/2$ units.) We have begun to develop ways to approximate such large nets with much smaller ones, by making assumptions about the covariance of members of clusters of units and then using single units to represent clusters of units. In the RNA secondary structure prediction problem, the clusters of interest are those diagonal clusters of units (corresponding to diagonals in the *2˚RNA* matrix) representing *potential stems* in the secondary structure.

The key to making such coarse-grained or multiresolution methods work, indeed the key to making any neural network method successful in molecular structure prediction, is to find a sensible mapping of the problem onto the network. Neural networks offer challenges in representing the static and dynamic structures of RNA (or protein), but success brings the benefits of massive parallelism and the simple, computable, and differentiable error measures needed for gradient descent and adaptive improvement in structure prediction. Solving the representational problems will lead to an entirely new class of fast, parallel, adaptive methods for predicting and simulating physical systems.

Notes

1. Much of the notation and many of the definitions in this section are adopted from [Sankoff et al., 1983]. We found their exposition on secondary

structure types and constraints to be the clearest by far, and we hope that their notational conventions become a standard in the field. Their text is copyright©, 1983, American Telephone and Telegraph Company, and used by permission.

Acknowledgements

The author would like to thank Dr. Geoffrey Hinton for helpful and insightful suggestions and critiques, and Dr. Rick Collins for his careful review of the original thesis. Thanks are also due to Dr. Carol Miernicki Steeg and Sarah Lesher for proofreading and helpful discussions and to Desirée Sy for editing and formatting help. The author appreciates the help of Raj Verma, Sudarsan Tandri, and Tim Horton with graphics and diagrams and Carol Plathan and Marina Haloulos with Macintosh tasks. Conversations with Dr. Larry Greller, Dr. F. Ray Salemme, and the UofT Connectionist Research Group have been very helpful in clarifying some ideas first described in the 1989 M. Sc. thesis.

Finally, the author gratefully acknowledges the support of the Natural Sciences and Engineering Research Council of Canada, the University of Toronto and its Connaught Fellowship Fund, and the E.I. Du Pont de Nemours & Co. in Delaware.

References

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985) A Learning Algorithm for Boltzmann Machines. Cognitive Science, 9:147-169.

Altona, C., van Beuzekom, A. A., Orbons, L. P. M., and Pieters, M. L. (1988).Minihairpin Loops in DNA: Experimental and Theoretical Studies. In *Biological and Artificial Intelligence Systems*, pages 93-124. ESCOM Science Publishers B.V., Kingston, New York.

Anderson, P. W. (1988). Spin Glass Hamiltonians: A Bridge Between Biology, Statistical Mechanics and Computer Science. In Pines, D., editor, *Emerging Syntheses in Science*. Addison-Wesley, Santa Fe, NM.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R. M., Lautrup, B., Norskov, L., Olsen, O. H., and Petersen, S. B. (1988). Protein Secondary Structure and Homology by Neural Networks: The Alpha-helices in Rhodopsin. *FEBS Letters*, 241:223-228.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R. M. J., Fredhom, H., Lautrup, B., and Petersen, S. B. (1990). A Novel Approach to Prediction of the 3-Dimensional Structures of Protein Backbones by Neural Networks. *FEBS Letters*, 261:43-46.

Eppstein, D., Galil, Z., and Giancarlo, R. (1988). Speeding up Dynamic Programming. In *Proceedings: Foundations of Computer Science. IEEE*.

Friedrichs, M. S. and Wolynes, P. G. (1989). Toward Protein Tertiary Structure Recognition by Associative Memory Hamiltonians. *Science*, 246:371-373.

Greller, L. D., Steeg, E. W., and Salemme, F. R. (1991). Neural Networks for the Detection and Prediction of 3D Structural Motifs in Proteins. In *Proceedings of the Eighth International Conference on Mathematical and Computer Modelling*, College Park, Maryland.

Hellstrom, B. J. and Kanal, L. N. (1990). Asymmetric Mean-field Neural Networks for Multiprocessor Scheduling. Computer Science UMIACS-TR-90-99, University of Maryland, College Park, Maryland.

Holley, L. H. and Karplus, M. (1989). Protein Secondary Structure Prediction with a Neural Network. *Proceedings of the National Academy of Sciences U.S.A.*, 86:152- 156.

Hopfield, J. and Tank, D. (1985). Neural Computation of Decisions in Optimization Problems. *Biological Cybernetics*, 52:141-152.

Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences U.S.A.*, 79:2554-2558.

Hunter, L. 1992 . Representing Amino Acids with Bitstrings. Submitted to *Computer Applications in the Biosciences*; code available by electronic mail from hunter@nlm.nih.gov.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220:671-680.

Major, F., Turcotte, M., Gautheret, D., Lapalme, G., Fillion, E., and Cedergren, R. (1991) The Combination of Symbolic and Numerical Computation for Three-dimensional Modeling of RNA. *Science*, 253:1255-1260.

Martinez, H. M. (1984). An RNA Folding Rule. *Nucleic Acids Research*, 12:323-334.

Martinez, H. M. (1988). An RNA Secondary Structure Workbench. *Nucleic Acids Research*, 16(5):1789-1798.

McClelland, J. L., Rumelhart, D. E., and the PDP Research Group (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume II*. Bradford Books, Cambridge, MA.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equations of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 6:1087-1091.

Nakai, K., Kidera, A., and Kanehisa, M. (1988) Cluster Analysis of Amino Acid Indices for Prediction of Protein Structure and Function. *Protein Engineering*, 2(2):93-100.

Nussinov, R. and Jacobson, A. (1980). Fast Slgorithm for Predicting the Secondary Structure of Single-stranded RNA. *Proceedings National Academy of Sciences, U.S.A.*, 77:6309-6313.

Nussinov, R., Piecznic, G., Grigg, J. R., and Kleitman, D. J. (1978). Algorithms for loop matchings. *SIAM Journal of Applied Mathematics*, 35(1):68-82.

Peterson, C. and Anderson, J. (1987). A Mean Field Theory Learning Algorithm for Neural Networks. MCC Technical Report EI-259-87, Microelectronics and Computer Technology Corporation, Austin, TX.

Peterson, C. and Soderberg, B. (1989). A New Method for Mapping Optimization Problems onto Neural Networks. *International Journal of Neural Systems*, 1.

Pipas, J. M. and McMahon, J. E. (1975). Methods for Predicting RNA Secondary Structures. In *Proceedings of the National Academy of Sciences, U.S.A.*, volume 72, pages 2017-2021.

Pleij, C. W. A. (1990) Pseudoknots: a New Motif in the RNA Game. *Trends in Biochemical Sciences*, 15:143-147.

Qian, N. and Sejnowski, T. J. (1988). Predicting the Secondary Structure of Globular Proteins Using Neural Network Models. *Journal of Molecular Biology*, 202:865-884.

Sankoff, D., Kruskal, J. B., Mainville, S., and Cedergren, R. J. (1983). Fast Algorithms to Determine RNA Secondary Structures Containing Multiple Loops. In Sankoff, D. and Kruskal,

J. B., editors, *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley, Reading, MA.

Steeg, E. W. (1989). Neural Network Algorithms for RNA Secondary Structure Prediction. Master's thesis, University of Toronto Computer Science Dept.

Steeg, E. W. (1990). Neural Network Algorithms for RNA Secondary Structure Prediction. Technical Report CRG-TR-90-4, University of Toronto Computer Science Dept., Toronto, Canada.

Steeg, E. and Takefuji, I. (1991) Comments on Parallel Algorithms for Finding a Near Maximal Independent Set of a Circle Graph; and Author's Reply. *IEEE Transactions Neural Networks*, 2(2):328-329.

Stein, D. L. (1985). A Model of Protein Conformational Substates. *Proceedings of the National Academy of Sciences, U.S.A.*, 82:3670-3672.

Stryer, L. (1981). *Biochemistry, 2nd Edition*. W. H. Freeman and Company, San Francisco.

Studnicka, G. M., Rahn, G. M., Cummings, I. W., and Salser, W. A. (1978). Computer Methods for Predicting the Secondary Structure of Single-stranded RNA. *Nucleic Acids Research*, 5(9):3365-3387.

Takefuji, I., Chen, L.-L., Lee, K.-C., and Huffman, J. (1990). Parallel Algorithms for Finding a Near-maximum Independent Set of a Circle Graph. *IEEE Transactions on Neural Networks*, 1(3):263-267.

Tinoco, I., Borer, P. N., Dengler, B., Levine, M. D., Uhlenbeck, O. C., Crothers, D. M., and Gralla, J. (1973). Improved Estimation of Secondary Structure in Ribonucleic Acids. *Nature New Biology*, 246:40-41.

Tinoco, I., Uhlenbeck, O. C., and Levine, M. D. (1971). Estimation of Secondary Structure in Ribonucleic Acids. *Nature (London)*, 230:362.

Waterman, M. S. (1978). *Advances in Mathematics: Supplementary Studies Vol. I, Studies in Foundations and Combinatorics*. Academic Press, New York.

Waterman, M. S. and Smith, T. F. (1978). RNA Secondary Structure: A Complete Mathematical Analysis. *Mathematical Biosciences*, 42:257-266.

Watson, J. D. and Crick, F. H. C. (1953). Molecular Structure of Nucleic Acids. A Structure for Deoxyribose Nucleic Acid. *Nature (London)*, 171:737-738.

Wilcox, G. L. and Poliac, M. O. (1989). Generalization of Protein Structure from Sequence Using a Large Scale Backpropagation Network. UMSI 89/22, University of Minnesota Supercomputer Institute, 1200 Washington Avenue South, Minneapolis, MN 55415.

Wolpert, D. H. (1990) Constructing a Generalizer Superior to NETtalk via a Mathematical Theory of Generalization. *Neural Networks*, 3(4):445-452.

Zuker, M. and Stiegler, P. (1981). Optimal Computer Folding of Large RNA Sequences Using Thermodynamics and Auxiliary Information. *Nucleic Acids Research*, 9:133-148.