A PARALLEL EVOLUTIONARY ALGORITHM FOR RNA SECONDARY STRUCTURE PREDICTION

by

Andrew Hendriks

B.Sc. (Information Technology, TechBC), Simon Fraser University, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
in the School
of
Interactive Arts and Technology

© Andrew Hendriks 2005
SIMON FRASER UNIVERSITY
Summer 2005

All rights reserved. This work may not be
reproduced in whole or in part, by photocopy
or other means, without the permission of the author,
except for non-profit, scholarly use, for which
no further permission is required.

APPROVAL

Name: Andrew Hendriks

Degree: Master of Science

Title of thesis: A Parallel Evolutionary Algorithm for RNA Secondary Structure Prediction

Examining Committee:

Dr. John E. Bowes,
Chair

Dr. Kay C. Wiese, Assistant Professor,
Computing Science
Simon Fraser University
Senior Supervisor

Dr. Belgacem B. Youssef, Assistant Professor,
Interactive Arts and Technology
Simon Fraser University
Supervisor

Dr. Mohamed Hefeeda, Assistant Professor,
Computing Science
Simon Fraser University
SFU Examiner

Date Approved: Jul. 04/2005

SIMON FRASER UNIVERSITY

PARTIAL COPYRIGHT LICENCE

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the right to lend this thesis, project or extended essay to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users.

The author has further granted permission to Simon Fraser University to keep or make a digital copy for use in its circulating collection.

The author has further agreed that permission for multiple copying of this work for scholarly purposes may be granted by either the author or the Dean of Graduate Studies.

It is understood that copying or publication of this work for financial gain shall not be allowed without the author's written permission.\

Permission for public performance, or limited permission for private scholarly use, of any multimedia materials forming part of this work, may have been granted by the author. This information may be found on the separately catalogued multimedia material and in the signed Partial Copyright Licence.

The original Partial Copyright Licence attesting to these terms, and signed by this author, may be found in the original bound copy of this work, retained in the Simon Fraser University Archive.

W. A. C. Bennett Library
Simon Fraser University
Burnaby, BC, Canada

Abstract

RNA is central in several stages of protein synthesis, and also has structural and functional roles in the cell. The shape of organic molecules such as RNA largely determines their function within an organic system. Current physical methods for structure determination are time consuming and expensive, thus methods for the computational prediction of structure are sought after. Various algorithms that have been used for RNA structure prediction include dynamic programming and comparative methods. This thesis introduces P-RnaPredict, a fully parallel coarse-grained distributed genetic algorithm (GA) for RNA secondary structure prediction. The impact of three pseudorandom number generators (PRNGs) on P-RnaPredict's performance is evaluated. The parallel speedup of P-RnaPredict is analyzed. Finally, the prediction accuracy of P-RnaPredict is evaluated through comparison to ten known structures, and compared to structures predicted by a Nussinov DPA implementation and the mfold DPA. P-RnaPredict offers similar performance to mfold, and outperforms the Nussinov DPA.

To my folks for their ceaseless support of my foolhardy exploits.

"Don't think; feel! It is like a finger pointing away to the moon. Don't concentrate on the finger or you will miss all that heavenly glory."

— *Enter the Dragon*, BRUCE LEE, *1974*

Acknowledgments

There are a large number of persons and organizations without whose assistance this work would never have come to fruition.

I was fortunate enough to first encounter Dr. Kay C. Wiese in the last year of my undergraduate degree. It was this association that kindled my interest in Bioinformatics, and it was his subsequent guidance and support over the years of my graduate degree that made the bulk of this research possible. His meticulous management ensured not only my fiscal wellbeing, but also cemented my scholarly work in terms of publishing and presenting at conferences. Whatever progress I have made as a scholar first and foremost is the result of his mentorship.

Dr. Belgacem Ben Youssef provided invaluable expertise on parallel computing and random number generation; he also kindly served on my supervisory committee, for which I am very grateful. I am also grateful to Dr. Mohammed Hefeeda for agreeing to be the Examiner of this thesis, and for his insightful comments.

Several organizations offered me substantial financial support in the form of grants, scholarships, and fellowships. Of these, I wish to first thank the Natural Sciences and Engineering Research Council (NSERC) of Canada for awarding me a Postgraduate Scholarship (PGS-A), which supported me throughout my degree. I would also like to offer my thanks to the Advanced Systems Institute of BC (ASI) for awarding me a Graduate Recruitment Assistance Program scholarship. Simon Fraser University, the Faculty of Applied Sciences, and the School of Interactive Arts and Technology (SIAT) also provided me with several fellowships. Finally, the following organizations supported my scholarship through grants: the Institute of Electrical and Electronics Engineers (IEEE), the Canadian Conference on Artificial Intelligence, the Congress on Evolutionary Computation (CEC), and the Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB).

My colleagues in the Infonet Media Lab founded by Dr. Wiese all supported this work and my research in a myriad of ways. Alain Deschênes was a vital member of the developer team and generously provided his technical insight. Edward Glen was the other essential member of the development team, and also contributed technical insight and his excellent jViz.Rna RNA visualization application. Kirt Noël and I spent many fruitful hours in the study of bioinformatics; he bestowed upon me equal measures of biological insight and mordant satire. Finally, Herbert Tsang provided a great sounding board for ideas and an engineering perspective on the project. I am quite grateful for their support.

Our research group in turn rested on the fantastic technical support provided by Gordon Pritchard, Patrick Lougheed, and Robin Johnson. These gentlemen frequently went above and beyond the call of duty to provide resources and resolve tough problems, and I owe them all a debt of gratitude.

Next are what in my view are the pillars of any academic organization, the administration and the library. Without the tireless efforts of the SIAT administration, committees, and individuals such as Dr. Rob Woodbury, it is likely my graduate degree would never have existed. Allison Neil, Joyce Black, Lynne Jamieson, Liz Konyari, Desiree Nazareth, Susan Siddall, Heather Clendening, and numerous other individuals answered my queries and resolved my problems tirelessly and with remarkable efficiency. The SFU Library and its librarians Natalie Gick, Gordon Coleman, Mabel Tang, and Cecilia Roberts were a great aid in my research. Finally, thanks to Penny Simpson for helping ensure the correct formatting of my thesis. My heartfelt thanks to you all.

Last and certainly not least are my parents, Georgina and Johannes Hendriks. Early on in life they cultivated and nourished my curiosity and love of science. They never ceased in their efforts to provide me with whatever I required over the course of my studies. To them I owe a debt which can never be repaid.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Importance of RNA

The central dogma of molecular biology states that during protein synthesis information "flows" from DNA to RNA to proteins. This occurs first through transcription of DNA into RNA and then translation of RNA into proteins. There are several different types of RNA involved in this process: Heterogeneous nuclear RNA (hnRNA) acts as the transcriber of DNA in eukaryotes. Messenger RNA (mRNA) carries the coded message to the ribosomes for synthesis. Ribosomal RNA (rRNA) is a component of ribosomes. Finally, transfer RNA (tRNA) combines the amino acids. Each of these types of RNA is synthesized by RNA polymerase [46].

However, recent research indicates that RNA has much more of a role than a mere carrier of information. Interesting examples of RNA's importance include retroviruses such as HIV which challenge the central dogma, employing the enzyme reverse transcriptase to transcribe their RNA into DNA and integrate it into the host genome [70]. Another example is that a new class of RNA molecules called small RNAs was discovered to operate many controls within the cell; Science named this "Breakthrough of the Year" in their December 2002 issue [8].

1.2 RNA Defined

RNA [93], or ribonucleic acid, is a biopolymer which is chemically similar to DNA, or deoxyribonucleic acid. It has three components: a pentose sugar known as ribose, phosphoric

acid, and the nitrogenous bases adenine, guanine, cytosine, and uracil.

RNA differs from DNA in three ways. First, the sugar in RNA is ribose rather than the deoxyribose in DNA. Second, the uracil in RNA is replaced by thymine in DNA. Finally, RNA is typically single-stranded as opposed to DNA which is usually double-stranded.

The primary building block of RNA is the nucleotide. A nucleotide is formed by making a phosphoester bond between the phosphoric acid and the sugar, and a glycosidic bond between the sugar and the nitrogenous base. Figure 1.1 illustrates the three elements of a nucleotide.

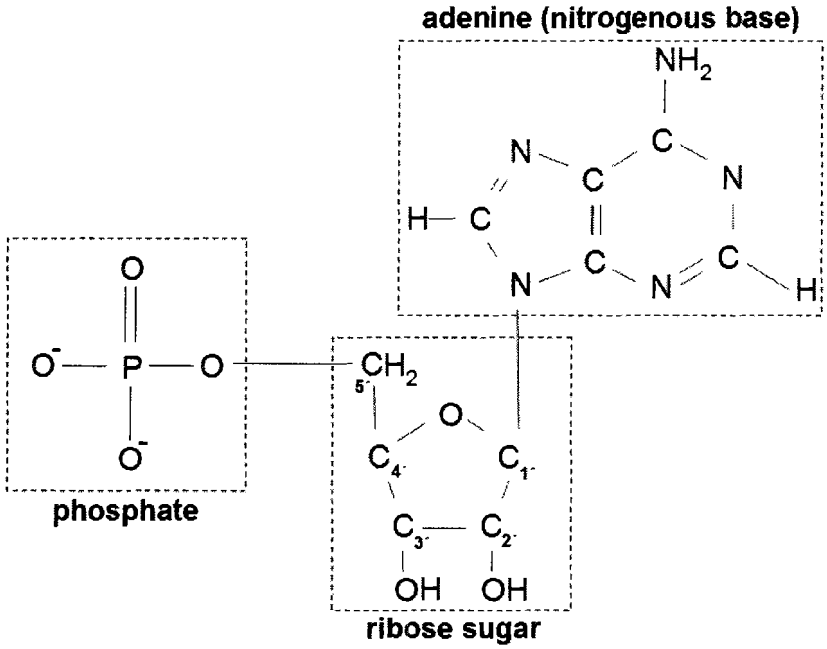

Figure 1.1: The elements of a nucleotide.

Individual nucleotides are polymerized through a phosphodiester linkage between the 3′ carbon of the ribose and an oxygen atom of the phosphate, forming a backbone of alternating sugars and phosphates. The resulting polynucleotide chain makes up a single strand of RNA, and is illustrated in Figure 1.2. Since the bases are independent of the nucleotide linkage, any combination of bases is possible. The phosphodiester linkages have a specific direction or polarity, which is conventionally read from 5′ to 3′, from left to right. The order of bases

in a chain are referred to as a sequence, and when written the bases are abbreviated to their first letter: adenine as A, guanine as G, cytosine as C, and uracil as U. An RNA sequence can then be written as GUCAAGU, or 5′-GUCAAGU-3′.

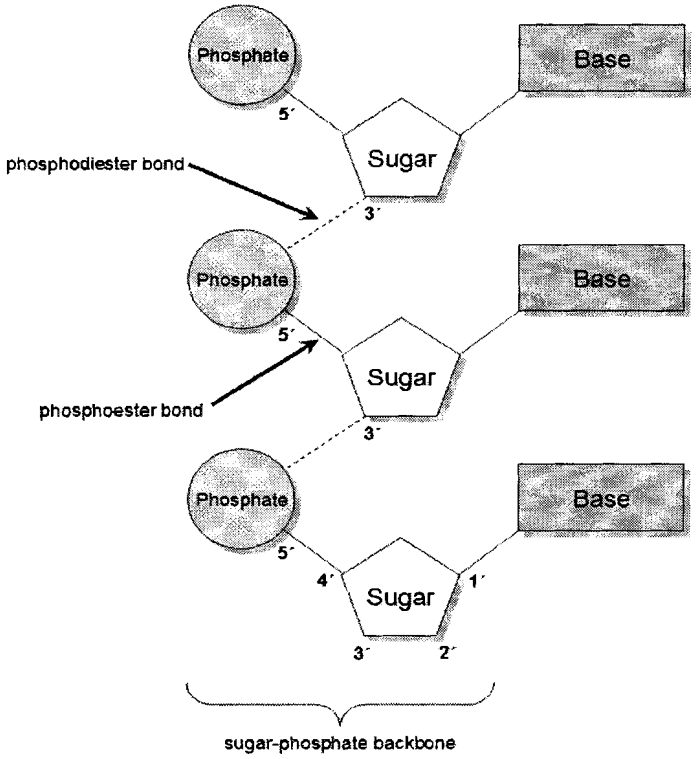

Figure 1.2: Polymerization of nucleotides into ribonucleic acid.

1.3 RNA Structure

The shape of organic molecules such as RNA largely determines their function within an organic system. The final three-dimensional structure forms when the sequence of nucleotides folds back onto itself. Structure is described hierarchically in biochemistry as primary, secondary, tertiary, and quaternary.

Primary structure refers to the linear sequence of bases which make up a biomolecule. As

mentioned in section 1.2, in RNA these are the nucleotide bases adenine, guanine, cytosine and uracil (A, G, C, and U).

Under the right thermodynamic conditions these four bases will form chemical bonds with each other and make base pairs. In typical RNAs over half of these are the Watson-Crick base pairs, AU and GC, and their mirrors. However, non-Watson-Crick base pairings also occur; the common ones are the sheared GA, GA imino, AU reverse Hoogsteen, and the GU and AC wobble pairs [65, 47]; the most common non-Watson-Crick base pair is the GU wobble pair.

Secondary structure refers to the structural elements which manifest as a result of these base pairings. Different structural elements will manifest themselves in the resulting secondary structure depending on which base pairs form bonds. These elements include *hairpin loops* which contain one base pair, *internal loops* which contain two base pairs, and *bulges* which contain 2 base pairs with 1 base from each of its pairs adjacent in the backbone of the molecule. There are also *multi-branched* loops, which contain more than two base pairs, and *external bases* which are not contained in any loop. Figure 1.3 illustrates examples of secondary structure elements.

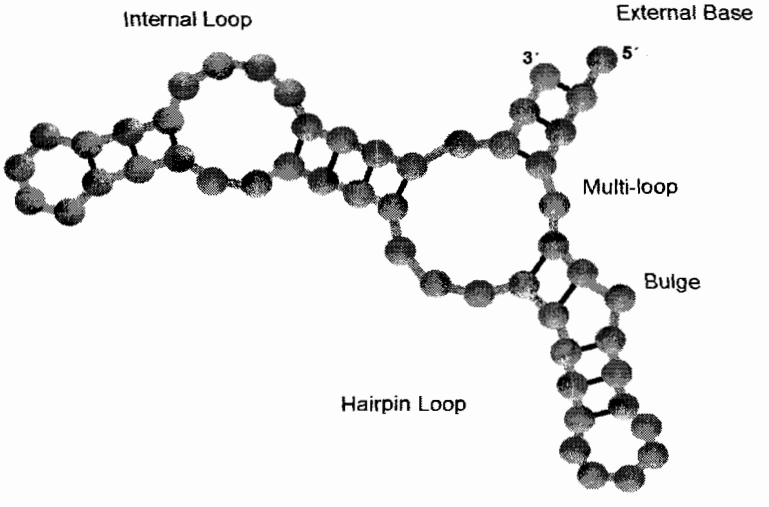

Figure 1.3: The elements of RNA secondary structure.

Stacked pairs, which form helices, provide stability in the secondary structure. A set

of stacked pairs is formed by two or more base pairs, such that the ends of the pairs are adjacent, forming a ladder type structure. These elements make up the secondary structure, which is typically thought of as two-dimensional.

The tertiary structure is the overall three-dimensional shape, and forms from the interactions between secondary structure elements. These tertiary interactions are formed through non-canonical base pairs, triple-stranded interactions such as base triples, and interactions between unpaired bases and the phosphodiester backbone of the RNA molecule [90]. Finally, quaternary structure refers to interactions between two or more RNA molecules.

When an RNA molecule folds, its overall free energy is reduced as bonds are formed between base pairs; this in turn increases the overall stability of the molecule. It should be noted that the energies involved through formation of secondary structure elements are significantly greater than those of tertiary elements; the set of secondary structure elements is also smaller [87].

One important RNA secondary structure element is the pseudoknot. A pseudoknot occurs when the nucleotides within a hairpin loop form base pairs with part of the RNA sequence outside of the loop. Figure 1.4 illustrates an example of a pseudoknot.

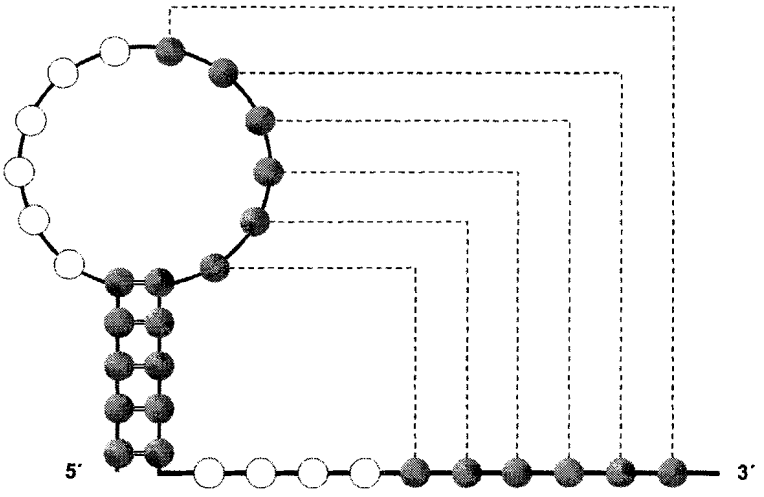

Figure 1.4: A secondary structure pseudoknot. Double lines and dotted lines indicate base pair bonds, solid lines indicate primary structure bonds.

1.3.1 Methods for RNA Structure Determination

There are two primary physical methods for determining RNA structure, X-ray crystallography [39] and Nuclear Magnetic Resonance [91]. In X-ray crystallography, structure is determined by crystallizing the molecule; analysis involves four steps [38]. First, an extremely pure, homogeneous sample of the RNA must be prepared. The RNA sample must then be crystallized through vapour diffusion combined with sparse matrix screening techniques. Next, the crystal is X-rayed, and data is collected on the resulting X-ray diffraction. Finally, a model of the electron density of the molecule is progressively built and refined against the data until the structure solution is resolved. Although molecules of practically any size may be analyzed using this method, it is effectively restricted to analyzing their crystalline form. This may not reflect the natural conformation of the RNA structure in its native environment.

Nuclear Magnetic Resonance employs a spectroscopic method in which the spin states of various atomic nuclei are probed via an electromagnetic field [69]. A minute sample of RNA must be purified, irradiated, and the resulting spectra analyzed by specialists. Unlike X-ray crystallography, the RNA structure can be analyzed in its native thermodynamic environment. However, the length of the sequences analyzed by NMR is limited to approximately 60 nucleotides.

Although promising, both methods of RNA structure determination are too time consuming and expensive to be effective. Thus, there is a keen interest in methods for predicting RNA structure computationally.

1.3.2 RNA Structure Prediction

With RNA it is possible to separate the process of folding from the primary structure to the tertiary structure into two hierarchical steps. First, the RNA sequence folds into its secondary structure through the formation of base pairs. Next, the three-dimensional tertiary structure forms from the secondary structure.

This claim can be substantiated with the knowledge that the energy involved in stabilizing secondary structure, base pairing, and base pair stacking is far greater than the energy involved in the interaction of secondary structure elements to form the tertiary structure. Thus, the secondary structure is much more stable than the tertiary structure, and tertiary

structure formation almost always follows secondary structure [87]. Evidence of the dominance of RNA secondary structures can also be found in nature, as secondary structure elements are conserved in evolution [29]. Thus, approaches to RNA structure prediction generally focus on secondary structure, which greatly simplifies the problem.

1.3.3 Methods for RNA Secondary Structure Prediction

The two most popular computational methods for RNA secondary structure prediction are comparative methods [23] and dynamic programming (DP) [57].

The basic principle on which comparative methods rely is that structure is conserved in RNA far more frequently than sequence. In other words, different RNA sequences can fold to form identical secondary and tertiary structures. Since function follows from structure, it can be seen that a biomolecule cannot undergo a total change in structure and retain what is likely an important function. If a mutation of a base occurs in the RNA during the course of evolution, a compensating mutation will often occur to maintain the original base pair position and the secondary structure of which it is a part. Otherwise, the RNA would suffer a loss of these secondary structure elements and likely the functionality they provide as well.

Comparative sequence analysis [74, 101] is performed on a set of homologous, phylogenetically related RNA sequences. First, multiple sequence alignment is performed on the set of sequences. Next, positions are found in the alignment that vary in a coordinated fashion, or covary, in order to maintain base pairing inside a potential secondary structure helix. This covariation analysis deduces base pair positions by locating identical or similar patterns of strong covariation in the multiple sequence alignment. There are two variants of comparative sequence analysis. The first is the Sankoff algorithm [74], which combines sequence alignment with a Nussinov [66] folding algorithm. The second is applied in the case where sequence conservation is very weak. Instead of performing sequence alignment, structures are predicted for each sequence and alignment is performed on the predicted structures. Current applications employing the basic comparative methods algorithm include *RNAalifold* [37], *Pfold* [43], and *ILM* [72]. Applications employing a restricted version of the Sankoff algorithm include *Foldalign* [26, 27], *Dynalign* [56], and *PMcomp* [36]. Finally, applications employing structure alignment include *RNA forester* [34] and *MARNA* [79]. A comprehensive comparison of comparative RNA structure prediction approaches is offered in [23]. One major limitation of comparative methods is that they require a set of RNA

sequences with minimal sequence diversity. If there is only one sequence available, or a set of RNAs with little diversity, comparative methods cannot be applied.

The DP approach to structure prediction proceeds with an *ab initio* method, where only the primary structure of the RNA molecule in question is known. DP is a technique typically applied to optimization problems which possess two properties [7]. The first is optimal substructure, where the optimal solution to the problem has optimal solutions to its subproblems. The second property is overlapping subproblems, in that a recursive algorithm for the problem solves the same subproblems repeatedly. The first application of DP to structure prediction was developed by Nussinov [66], and functioned by maximizing the number of base pairs in a predicted structure. In this case it was assumed that the formation of a given base pair was independent of all other base pairs. Thus, the prediction problem can be divided into subproblems, the subproblems solved and their solution tabulated, and the final structure computed from the tabulated subproblem solutions. The main problems with this first DP algorithm were as follows: First, there were no constraints on hairpin loop length, whereas actual RNA requires a minimum of about 3 nucleotides. Second, there were no size constraints on bulges and internal loops; this resulted in helices with shorter lengths than what would occur in actual RNA.

Later, Zuker [113, 107, 108, 112, 109, 110] developed an alternate approach which employed thermodynamic models to minimize the free energy of the predicted structure. Free energy can be thought of as a measure of stability in RNA secondary structures, and is discussed in greater detail in Section 2.1. Instead of assuming base pairs were completely independent as per Nussinov, the free energy of helices was based on the stacking contribution from the interaction between base pairs. The destabilizing energy contribution of loops is also considered. Thus, the free energy of a predicted structure is roughly the sum of the free energies of secondary structure elements such as hairpin loops, bulges, internal loops, and multi-branched loops. Recurrence relations capture the detail of each loop length and type, and the predicted structure with the global minimum free energy is returned.

A key problem with predicting structures based on minimum free energy is that the natural fold is frequently found to be in a suboptimal energy state [89]. In general, structures with 92% of their base pairs determined through phylogenetic analysis are found within 2% of the minimum free energy [53]. This problem was addressed through the modification of the DP algorithm to allow it to determine suboptimal RNA secondary structures within a specified range of the minimum free energy [107].

Current DP implementations include *mfold* [110] for UNIX, a Windows-based version of *mfold*, *RNAStructure* [54] developed by Mathews, and *Vienna* [35] developed by Hofacker. *Vienna* is a C code library and several applications for RNA structure prediction and comparison. The three DP structure prediction applications are a basic minimum free energy algorithm which generates a single optimal structure [113], a partition function algorithm [60, 15] which computes base pair probabilities, and an algorithm to generate all suboptimal structures within a specified range of the minimum free energy [103, 104].

1.4 Genetic Algorithms

Evolutionary computing [1] refers to methods by which evolution is simulated on a computing platform. The algorithms reliant on these methods are known as evolutionary algorithms, and include evolution strategies [71, 75], evolutionary programming [18], genetic programming [44], and genetic algorithms [40].

Genetic algorithms (GAs) are widely applicable search and optimization techniques. In general, they have five basic components [64]:

1. A genetic representation of solutions to the problem
2. A way to create an initial population of candidate solutions
3. An evaluation function rating solutions in terms of their fitness
4. Genetic operators that alter the genetic composition of children during reproduction
5. Values for the parameters of GAs

The GA maintains a population of individuals, usually bitstrings, in which each individual represents a potential solution to the given problem. For each cycle or "generation" each individual is evaluated to determine its relative fitness. Those individuals representing the current best solution to the problem are maintained. Two fundamental operators, mutation and crossover, may be applied to this population. Mutation creates new individuals by randomly altering individuals, while crossover combines elements from two separate individuals to make new individuals. These new individuals replace the least fit individuals in the population. This completes one generation, and the cycle repeats itself, now evaluating the new individuals or offspring. After a varying number of generations, the algorithm will

converge to the best individual, which represents a suboptimal or optimal solution to the given problem.

Two key considerations of the search strategies are exploitation of good solutions and exploration of the search space. GAs execute an essentially blind search in complex fitness landscapes; selection operators should direct the search towards the most favourable areas of the landscape. GAs must strike a balance between exploitation and exploration to be successful. [24]

1.5 GAs in RNA Structure Prediction

In the case of DP and GAs, structure prediction can be formulated as an energy minimization problem using thermodynamic models. While DP has been shown to accurately predict a structure with minimum energy [57], the natural fold has been shown to vary greatly from the predicted one [30]. However, van Batenburg et al. [89] implemented a simple binary GA which outperformed the DP algorithm when considering true-positive canonical base pairs in the structure. As the base pairs ultimately make up the secondary structure, this development is quite significant. Van Batenburg et al. attribute this result to how the GA emulates the natural folding pathway by adding and removing the helices which make up the secondary structure. This suggests that GAs may perform well in this domain. Other GA designs have been applied to the RNA structure prediction problem. These include massively parallel GAs [78], and serial GAs [88, 89].

1.6 Research Question

The underlying research question for this thesis is whether a coarse-grained distributed GA, *P-RnaPredict*, can successfully perform RNA secondary structure prediction. Facets of this thesis include an investigation into the effects of pseudorandom number generator quality, an analysis of the parallel speedup, and a brief overview of parameter optimization as applied to the parallel algorithm. The viability of this approach is demonstrated through the comparison of the structures predicted by *P-RnaPredict* to known structures, and against structures predicted by the Nussinov and *mfold* dynamic programming algorithms (DPAs).

My specific contributions in this thesis are as follows:

- modification of an existing single-population canonical GA into a coarse-grained distributed GA
- development of a serial simulation of the distributed GA to determine if parallelization was worthwhile
- modification of the serial simulation into a fully parallel coarse-grained distributed GA
- evaluation of the influence of parallel PRNGs on the quality of output of the distributed GA
- evaluation of the performance of the GA in terms of prediction accuracy through comparison to known structures, and the Nussinov [66] and *mfold* [110] DP algorithms

1.7 Thesis Overview

Free energy minimization and thermodynamic models as applied within *P-RnaPredict* are discussed in detail in Chapter 2. The representation of RNA structures in *P-RnaPredict* are reviewed in Chapter 3. In Chapter 4, the methods whereby the serial GA was modified to become a serial simulation of a distributed GA are presented. The impact of pseudorandom number generation on *P-RnaPredict* is investigated in Chapter 5. Chapter 6 presents the results of the parallel speedup testing. A comparison of the structures predicted by *P-RnaPredict* to known structures is offered in Chapter 7. Chapter 8 reviews a comparison of the Nussinov DPA and *P-RnaPredict*. A comparison between the *mfold* DPA and *P-RnaPredict* is presented in Chapter 9. Finally, conclusions and future work are offered in Chapter 10.

1.8 Summary

This chapter presented an overview of RNA secondary structure prediction and its importance. The biochemistry behind RNA and its structure were reviewed. A synopsis of the methods for secondary structure prediction, including GAs, was offered. Finally, the research question and an overview of the thesis were presented.

Chapter 2

Free Energy Minimization

As mentioned in Section 1.4, a GA requires a fitness function to evaluate candidate solutions. The fitness function employed to guide this structure prediction algorithm is free energy, and is discussed in detail below. The four thermodynamic models employed in *P-RnaPredict* are reviewed, and a general critique of free energy minimization is presented.

2.1 Free Energy

Gibbs free energy is a measure of the energy available in a system to do work under conditions of standard temperature and pressure. It can be expressed by the following equation,

$$\Delta G = \Delta H - T\Delta S \tag{2.1}$$

where ΔG is the change in free energy, ΔH is the change in enthalpy, a measure of the heat content of a chemical system, T is the temperature in degree Kelvin, and ΔS is the change in entropy, a measure of the disorder in a chemical system. In a chemical reaction, bonds between atoms are broken and replaced by other bonds, and a transfer of energy occurs. This transfer of energy also happens in RNA folding when base pairs form through chemical (hydrogen) bonds.

In a chemical reaction, or a change in configuration of RNA structure, ΔG quantifies the spontaneity of the reaction. If the ΔG of a given process is negative, the products of that process are favoured, and the process can proceed spontaneously. On the other hand, a positive ΔG favours the reactants, and the process cannot proceed spontaneously. When

equilibrium is reached, $\Delta G = 0$ and no further change in free energy occurs. As suggested by Equation 2.1, the free energy lost as equilibrium is reached is transformed into either heat or increases the amount of entropy.

In the case of RNA secondary structure prediction, ΔG is used to quantify the spontaneity of a RNA molecule in folding into specific secondary structure configurations. The details of how RNA structural configurations are related to a specific value of ΔG are captured within thermodynamic models, and are explained in the next section.

2.2 Thermodynamic Models

In our model, RNA secondary structure forms as a consequence of chemical (hydrogen) bonds that form between specific pairs of nucleotides. These are GC, AU, and GU, and their mirrors which are collectively known as the canonical base pairs. Searching a sequence of nucleotides for all possible base pairs is rapid and straightforward; the challenge comes from attempting to predict which specific canonical base pairs will form bonds in the real structure.

The change in free energy associated with RNA structural configurations is captured within the four thermodynamic models implemented within *P-RnaPredict*. These models can be divided into two main groups. The first are the hydrogen bond models proposed by Major [96] and Mathews et al. [55]. The second group are the stacking-energy thermodynamic models Individual Nearest Neighbour (INN) [3, 21, 76] and Individual Nearest Neighbour-Hydrogen Bond (INN-HB) [105].

2.3 Review of Base Pairing Models

Base pairing or hydrogen bond thermodynamic models are based on the idea that each base pair contributes individually to the free energy change of the entire structure, and there is no interaction between base pairs.

2.3.1 The Major Model

The main premise behind the Major thermodynamic model is that each base pair contributes to the free energy of a RNA secondary structure based on the relative strength of its chemical bonds. The bond strengths are in turn based on the following facts: GC base pairs have

three hydrogen bonds, the AU pair has two, and the wobble pair GU has much weaker bonding than the AU pair [97]. Thus, each pair makes the free energy contribution [96] shown in Table 2.1.

Table 2.1: Major Thermodynamic Model, all values at 37 °C

Base Pair	ΔG (kcal/mol)
GC	-3 kcal/mol
AU	-2 kcal/mol
GU	-1 kcal/mol

Each base pair forms completely independently of all other base pairs in a given RNA secondary structure, and the total free energy of a given RNA structure is computed via Equation 2.2 [97].

$$E(S) = \sum_{i,j \in S} e(r_i, r_j) \tag{2.2}$$

Here, $e(r_i, r_j)$ denotes the free energy ΔG contribution between the i^{th} and j^{th} nucleotide from the formation of a base pair.

2.3.2 The Mathews Model

The Mathews model is quite similar to the Major model discussed above. The difference is that instead of a proportional strength rating, the free energy contribution of the base pairs is based solely on the number of hydrogen bonds they form: GC with three hydrogen bonds, AU with two, and the GU also with two. Thus, each pair makes the free energy contribution shown in Table 2.2.

Table 2.2: Mathews Thermodynamic Model, all values at 37 °C

Base Pair	ΔG (kcal/mol)
GC	-3 kcal/mol
AU	-2 kcal/mol
GU	-2 kcal/mol

Again, the free energy contributions of the base pairs are summed to compute the free

energy of a given structure [55] as per Equation 2.2.

2.4 Review of Stacking Energy Models

The essential idea of stacking-energy models is that the stabilizing contribution each base pair makes to its helix depends on that base pair's nearest neighbours. For example, the free energy contribution of a GC base pair would vary depending on whether the adjacent base pair in the helix is an AU base pair, or its mirror a UA base pair. This differs from the simple base pairing models reviewed in Section 2.3 which only consider the individual base pair and disregard both its orientation and its neighbours.

2.4.1 Individual Nearest-Neighbour Model (INN)

There are two distinct components to computing the free energy of a helix using INN. The first is initiation, or the formation of the first base pair. Initiation brings the two strands together and entails hydrogen bonding. The second component is propagation, or the continued formation of subsequent base pairs. Propagation involves nearest-neighbour or stacking interactions as well as hydrogen bonding. The nearest-neighbour thermodynamic parameters used in the INN model were initially measured at 25°C [3], but were later re-measured and extended at 37°C [20, 21, 22, 31, 84, 102, 106].

The following example, taken from [76], is of a predicted free-energy change of helix formation of the symmetric duplex $\begin{matrix} 5'\text{GGAUCC}3' \\ 3'\text{CCUAGG}5' \end{matrix}$

$$\begin{aligned} \Delta G^\circ_{37} = {} & \Delta G^\circ_{37} \begin{matrix} \text{GG} \\ \text{CC} \end{matrix} + \Delta G^\circ_{37} \begin{matrix} \text{GA} \\ \text{CU} \end{matrix} + \Delta G^\circ_{37} \begin{matrix} \text{AU} \\ \text{UA} \end{matrix} + \\ & \Delta G^\circ_{37} \begin{matrix} \text{UC} \\ \text{AG} \end{matrix} + \Delta G^\circ_{37} \begin{matrix} \text{CC} \\ \text{GG} \end{matrix} + \\ & \Delta G^\circ_{37\,\text{init}} + \Delta G^\circ_{37\,\text{sym}} \end{aligned} \tag{2.3}$$

Here the set of nearest neighbour terms is formed by working through each base pair from left to right through the duplex and grouping it with the base pair on the right. The $\Delta G^\circ_{37\,\text{init}}$ term is a constant which accounts for the entropy loss when initiation occurs and the first base pair is formed. $\Delta G^\circ_{37\,\text{sym}}$ is a symmetry correction term for duplexes with complementary strands.

This next example, also taken from [76], illustrates how INN accounts for 3′ unpaired nucleotides in the duplex $\begin{matrix} 5'\ \text{GGAUCCA}3' \\ 3'\text{ACCUAGG}\ 5' \end{matrix}$

$$\Delta G^\circ_{37} = \Delta G^\circ_{37}(\text{Core duplex}) + 2\Delta G^\circ_{37} \begin{matrix} \text{CA} \\ \text{G} \end{matrix} \tag{2.4}$$

In the above equation, the term $\begin{matrix} \text{CA} \\ \text{G} \end{matrix}$ is considered to be the same as $\begin{matrix} \text{G} \\ \text{AC} \end{matrix}$, except that it has been rotated by 180°.

A similar case occurs with the 5′ unpaired nucleotides in the duplex $\begin{matrix} 5'\text{AGGAUCC}\ 3' \\ 3'\ \text{CCUAGGA}5' \end{matrix}$

$$\Delta G^\circ_{37} = \Delta G^\circ_{37}(\text{Core duplex}) + 2\Delta G^\circ_{37} \begin{matrix} \text{AG} \\ \text{C} \end{matrix} \tag{2.5}$$

Terminal mismatches, such as those in the duplex $\begin{matrix} 5'\text{AGGAUCCA}3' \\ 3'\text{ACCUAGGA}5' \end{matrix}$ are accounted for using specific mismatch terms as shown below:

$$\Delta G^\circ_{37} = \Delta G^\circ_{37}(\text{Core duplex}) + 2\Delta G^\circ_{37} \begin{matrix} \text{CA} \\ \text{GA} \end{matrix} \tag{2.6}$$

A thorough review of the INN model complete with thermodynamic parameters can be found in [76].

2.4.2 Individual Nearest-Neighbour Hydrogen Bond Model (INN-HB)

Later experimentation determined that duplexes with identical nearest neighbours but varying terminal ends also differed in their stabilities. Specifically, a duplex with one additional terminal GC pair and one or less terminal AU pairs is always more stable [105].

This difference is accounted for by a modification of the INN model, described via the following equation:

$$\begin{aligned} \Delta G^\circ(\text{duplex}) \;=\; & \Delta G^\circ_{\text{init}} + \sum_j n_j \Delta G^\circ_j(\text{NN}) + \\ & m_{\text{term-AU}} \Delta G^\circ_{\text{term-AU}} + \Delta G^\circ_{\text{sym}} \end{aligned} \tag{2.7}$$

Each $\Delta G^\circ_j(\text{NN})$ term accounts for the free energy contribution of the jth nearest neighbour with n_j occurrences in the sequence. The $m_{\text{term-AU}}$ term is the number of terminal

AU pairs, and the $\Delta G^{\circ}_{\text{term-AU}}$ term is the free energy contribution of a terminal AU pair. The only difference from the INN model is the inclusion of the $m_{\text{term-AU}}\Delta G^{\circ}_{\text{term-AU}}$ term to account for the free energy penalty attributed to terminal AU pairs. With the thermodynamic tables from [22], the computation of the stability of a given helix is straightforward.

The following examples are taken from [105]. The first example is the stability of a non-self-complementary duplex with one terminal AU pair, $\begin{matrix}5'\text{ACGAGC}3'\\ 3'\text{UGCUCG}5'\end{matrix}$:

$$\begin{aligned}\Delta G^{\circ}(\text{duplex}) &= \Delta G^{\circ}_{37\,\text{init}} + \Delta G^{\circ}_{37}\begin{matrix}\text{AC}\\ \text{UG}\end{matrix} + \\ &\quad \Delta G^{\circ}_{37}\begin{matrix}\text{CG}\\ \text{GC}\end{matrix} + \Delta G^{\circ}_{37}\begin{matrix}\text{GA}\\ \text{CU}\end{matrix} + \\ &\quad \Delta G^{\circ}_{37}\begin{matrix}\text{AG}\\ \text{UC}\end{matrix} + \Delta G^{\circ}_{37}\begin{matrix}\text{GC}\\ \text{CG}\end{matrix} + \\ &\quad 1\times\Delta G^{\circ}_{37}\begin{matrix}5'\text{A}\\ 3'\text{U}\end{matrix}\end{aligned} \tag{2.8}$$

The $1\times\Delta G^{\circ}_{37}\begin{matrix}5'\text{A}\\ 3'\text{U}\end{matrix}$ term accounts for the destabilizing effect of the terminal AU base pair.

The second example illustrates the stability computation for a self-complementary duplex with two terminal AU base pairs, $\begin{matrix}5'\text{UGGCCA}3'\\ 3'\text{ACCGGU}5'\end{matrix}$:

$$\begin{aligned}\Delta G^{\circ}(\text{duplex}) &= \Delta G^{\circ}_{37\,\text{init}} + 2\times\Delta G^{\circ}_{37}\begin{matrix}\text{CA}\\ \text{GU}\end{matrix} + \\ &\quad 2\times\Delta G^{\circ}_{37}\begin{matrix}\text{GG}\\ \text{CC}\end{matrix} + \Delta G^{\circ}_{37}\begin{matrix}\text{GC}\\ \text{CG}\end{matrix} + \\ &\quad 2\times\Delta G^{\circ}_{37}\begin{matrix}\text{A}3'\\ \text{U}5'\end{matrix} + \Delta G^{\circ}_{37\,\text{sym}}\end{aligned} \tag{2.9}$$

Note that the $\Delta G^{\circ}_{37}\begin{matrix}\text{A}\\ \text{U}\end{matrix}$ term is doubled to account for both terminal AU pairs.

Although the INN-HB model only specifies a penalty for terminal AU pairs, terminal GU pairs are given the same penalty as suggested by Mathews et al. in [55].

The INN-HB model as presented here is unable to account for higher-order structures such as loops, junctions, bulges, and pseudoknots. In fact, Zuker et al. [112] state that the energy rules break down when pseudoknots are included.

2.5 Critique of the Minimum Free Energy Thermodynamic Model

There are several important instances where the free energy model is inadequate in determining RNA structure [23]. Aside from the primary RNA structure, other cellular components such as chaperones, base modifications, and transcription itself have an impact on RNA folding. Chaperones are proteins which assist other biomolecules in proper folding. Base modification is the post-transcription alteration of RNA bases; one example of this is pseudouridine in tRNA [93]. Another important class of RNA structures is the riboswitch [50], which regulates gene expression. Riboswitches are RNA sequences which possess two or more functional structures. Finally, the prediction of pseudoknots (defined in Section 1.3) is not adequately handled due to their relative scarcity in RNA structures, and the computational complexity involved in detecting them. Other drawbacks to the minimum free energy model are that it is based on physical measurement, which potentially suffers from error through noisy data. There is also a lack of modeling of global interactions between secondary structure elements.

With a GA evaluation function for RNA structure prediction defined, the next consideration is a genetic representation of potential solutions.

2.6 Summary

An overview of the free energy minimization fitness metric has been offered in this chapter. A definition of Gibbs free energy was presented, and the four thermodynamic models implemented in *P-RnaPredict* were reviewed. Finally, a critique of the minimum free energy method was presented.

Chapter 3

RNA Representation in the GA

Representations of secondary structures present unique challenges in the implementation of GAs. Most implementations employ the pre-calculation of all secondary structure elements such as stems and loops which are possible within the given RNA sequence. The representation of the solution in this case is then a selection of the available elements from the pre-calculated set. However, it must be noted that certain helices in the set are likely to be mutually incompatible.

3.1 Helix Generation Algorithm

In our model, a helix is specified by three constraints. First, each helix must have at least three stacked canonical base pairs. Second, the sequence or loop connecting the two strands must be at least 3 nucleotides long. Third, each helix must not share bases with another. With this in mind, the pseudocode of the helix generation algorithm is shown in Figure 3.1.

A walkthrough of the helix generation pseudocode follows. Start with a pair (r_i, r_j) from the pre-generated set of base pairs. The first action is to determine if this base pair is already part of another helix. This is done by checking if the base pair (r_{i-1}, r_{j+1}) is a canonical base pair. If it is, it is assumed that this base pair is already part of a previously generated helix and it is discarded. If not, the next step is to attempt "growing" the potential helix h by stacking base pairs on top of it. The indices i and j are incremented and decremented respectively, and each successive pair is checked to determine if is it a canonical base pair. Thus, the base pairs $\{(r_{i+1}, r_{j-1}), (r_{i+2}, r_{j-2}), (r_{i+3}, r_{j-3}), ...\}$ are added to h until the first non-canonical pair is encountered. At this stage, two criteria are applied. If h has a

```
Generate set of possible base pairs (r_i, r_j) from given sequence;
initialize helix h; for each pair (r_i, r_j) do
    while (helix h is valid) and (helix h is incomplete) do
        if ((r_i, r_j) is a canonical base pair) and ((r_i, r_j) is not part of an existing
        helix) then
            add base pair (r_i, r_j) to helix h;
            increment index i;
            decrement index j;
        else
            if helix h contains less than 3 base pairs then
                helix h is invalid;
            else if helix h has less than 3 bases in between the last base pair then
                helix h is invalid;
            else
                helix h is complete;
            end
        end
    end
    if (helix h is valid) then
        insert helix h into set of all helices H;
    end
end
```

Figure 3.1: Helix generation pseudocode

minimum of three stacked base pairs, and at least three unpaired bases between the last base pair, it is considered valid and added to the set of all helices H. Otherwise, helix h is discarded and the process continues with the next base pair.

With this formulation for structure generation, the structure prediction problem becomes a combinatorial optimization problem; the final solution is a subset of the set H of all possible helices that contains only the helices which make up the final structure. The constraint that no helix share bases with another ensures that only chemically feasible structures are predicted. Since the helices determined by this model are always a contiguous set of base pairs, the conflict data is computed by looking for overlap between the $3'$ and $5'$ indices of the base pairs at the top and bottom of each helix. This conflict data is then captured within a lookup table for swift retrieval.

3.2 Size of Search Space

A general comment on the size of the search space resulting from the helix generation algorithm should be made. If we consider that each helix found in the structure can either be present or absent, then the total number of structures which can be generated from a given sequence is $2^{|H|}$, where H is the set of all helices.

Table 3.1: Total helices found by sequence. Details for each sequence can be found in the table associated with its name.

Organism Name	Sequence Length (nt)	Total Helices Found
S.cerevisiae (Table 4.5)	118	175
H.marismortui (Table 4.4)	122	198
A.lagunensis (Table A.17)	468	2324
H.rubra (Table 4.3)	543	3933
A.griffini (Table 4.2)	556	3637
C.elegans (Table 4.1)	697	6074
D.virilis (Table 7.6)	784	8491
X.laevis (Table 7.2)	945	8233
H.sapiens (Table A.6)	954	8481
S.acidocaldarius (Table A.23)	1493	32274

For example, the 1493 nt *S.acidocaldarius* sequence results in a set of 32274 helices, and a search space of 2^{32274} possible structures, an extremely large search space.

3.3 RNA Structure Representation

When a GA randomly assembles stems from the common set into individuals to populate the first generation, it may assemble an infeasible individual with incompatible stems. The stochastic operation of the mutation and crossover operators could also easily produce infeasible individuals; this was resolved by repairing individuals containing incompatible helices. Other improvements to this initial design were accomplished by adding a finer-grained fitness function, and favouring additions to removals of stems in the mutation operator as suggested by [89].

The existing serial GA incorporates two separate representations: binary-based and permutation-based. In the *binary-based representation*, each helix in H was represented by one bit in a bitstring; a set bit indicated the presence of a helix, and an unset bit indicated its absence. For example, consider a set of helices H, containing five helices. A candidate structure h might contain, assuming there are no conflicts, the second, fourth, and fifth helices from H. The binary representation of that would be the bitstring {01011}. As noted above, repair of each individual is necessary due to the potential for helix conflicts. The bitstring is iterated through from left to right. If a bit is set, the helix represented at that bit is checked for conflicts with helices to the left of it. If a conflict is found, that bit is unset.

In the *permutation-based representation* [95, 96], each helix in H is numbered by an integer ranging from 0 to $n-1$, n being the total number of generated helices. A given structure is represented by a permutation of that set of integers. Permutation-based individuals are not repaired; the task of producing only feasible solutions is left to the decoder. This is accomplished by iterating through the permutation from left to right. The helix specified by each integer is checked for conflicts with helices to its left. If no conflicts are found, the helix is retained; otherwise it is discarded. In the permutation-based representation, there are multiple possible encodings. For example, consider again the set H above. Since the decoding process proceeds from left to right and only removes helices if there is a conflict, it is assumed that the first helix conflicts with the fifth helix, and the third helix conflicts with the second and fourth helices. Starting the permutation numbering from zero, one possible encoding for a structure is {1,3,4,0,2}; another is {4,0,1,3,2}. It should be noted that both repair and decoding add substantially to the computational complexity of the fitness evaluation for the GA.

This defines the binary-based and permutation-based methods for creating a genetic representation of potential solutions. Crossover and mutation operators for these representations are well established; the specific types used in the GA are discussed in Section 3.4.1. At this point, all the necessary components for a canonical GA for RNA secondary structure prediction have been introduced. The next section focuses on the design of *RnaPredict*, the GA which is the foundation for the parallel GA, *P-RnaPredict*.

3.4 *RnaPredict*–a GA for RNA Folding

The foundation for *P-RnaPredict* is a serial GA for RNA secondary structure prediction developed initially by Wiese and Glen [97]. This GA employed permutations to encode RNA secondary structures, and was named *RnaPredict*. Deschênes and Wiese [11] extended *RnaPredict* by implementing the stacking-energy based thermodynamic models INN and INN-HB. These extensions were found to predict certain RNA secondary structures with very high accuracy and also outperform a DP algorithm [13].

The next section provides a brief overview of *RnaPredict*'s design.

3.4.1 Serial GA Design

The operators and parameters necessary to configure *RnaPredict* [11] are as follows:

- Generations
- Population Size
- Crossover Probability (P_c)
- Mutation Probability (P_m)
- Crossover Operator
- Selection Strategy (KBR or STDS)
- Elitism (On / Off)
- Thermodynamic Model (Major, Mathews, INN, or INN-HB)
- Pseudoknots (On / Off)

- Random Seed

"Generations" is the number of generations for which the GA will execute. "Population Size" is the total number of individuals in a single population. P_c is the probability of crossover occurring on a pair of parents. P_m is the probability of mutation occurring on a given child. For the binary-based representation, mutation is simply flipping a random bit to its alternate value. For the permutation-based representation, mutation is simply swapping the integers at two random indices.

"Crossover" is the type of crossover employed. Since *RnaPredict* supported both binary-based and permutation-based structure representation, crossovers of both types are supported. They include the binary crossovers 1-Point, N-Point and Uniform, and the permutation crossovers Cycle Crossover (CX) [67], Order Crossover (OX) [9], Order #2 Crossover (OX2) [85], Edge Recombination Crossover (SYMERC) [94], Partially Matched Crossover (PMX) [25], and Asymmetric Edge Recombination Crossover (ASERC) [98].

The "Selection Strategy" parameter offers two types of strategies to manage the output of crossover. Standard Selection (STDS) simply passes the two children of crossover directly through to the next generation, regardless of their fitness. Keep-Best Reproduction (KBR) selects the most fit parent and the most fit child and passes them on to the next generation. "Elitism," if applied, retains the best individual from the previous generation to the next generation. The "Thermodynamic Model" parameter determines which of the four models described above (Major, Mathews, INN, or INN-HB) is used in the GA. "Allow Pseudoknots" determines whether pseudoknots are permitted in the predicted structure. Finally, the "Random Seed" parameter lets the user choose a random seed to initialize random number generation for the GA. Pseudocode for the *RnaPredict* algorithm is shown in Figure 3.2.

Earlier research with *RnaPredict* determined that there were certain optimal combinations of parameters [10, 11, 12, 13]. These settings, based on the use of 1-Elitism, were a permutation-based representation, 700 generations, a population size of 700, a P_c of 0.7, a P_m of 0.8, the OX2 and CX crossovers, a selection strategy of STDS, the INN and INN-HB thermodynamic models, and the disallowing of pseudoknot prediction. *RnaPredict* is able to disable pseudoknots due to the conflict table generated by the helix generation algorithm.

At this stage it was suggested that *RnaPredict* should be parallelized to improve performance. Since there were a number of techniques applicable, further investigation was

```
Generate set of possible base pairs from sequence;
Generate set of possible helices using set of base pairs;
Generate initial random population;
for all generations do
    for population size / 2 do
        Select two parents;
        if random value < P_c then
            crossover parents to create two children;
        else
            pass parents through as children;
        end
        for each child do
            if random value < P_m then
                randomly mutate this child;
            end
        end
        if selection strategy is KBR then
            retain best parent and best child based on fitness;
        else if selection strategy is STDS then
            always retain children;
        end
        insert retained individuals into new population;
        apply 1-Elitism and retain the best individual from the previous generation;
    end
end
output best structure;
```

Figure 3.2: *RnaPredict* pseudocode

necessary to determine how to proceed. Chapter 4 discusses this process.

3.5 Summary

This chapter has presented the method whereby RNA secondary structure is represented in *P-RnaPredict*. The helix generation algorithm was reviewed in depth, complete with pseudocode. Commentary was made on the substantial size of the structure prediction search space. The method of RNA structure representation was detailed, including the differences between binary-based and permutation-based representations. Finally, the predecessor to *P-RnaPredict*, *RnaPredict*, was briefly discussed.

Chapter 4

P-RnaPredict – a Parallel GA for RNA Folding

This chapter reviews the process whereby the serial GA *RnaPredict* was parallelized to produce *P-RnaPredict*, a fully parallel distributed GA. There are two distinct stages in this process. The first was the creation of a serial simulation of the distributed GA, covered in Section 4.1. This section discusses the benefits of the various approaches for parallelization, ultimately making the case for a multi-population GA. The second was the parallelization of the serial simulation, covered in Sections 4.2 and 4.3. These two sections cover the justification for parallelization and the implementation details, respectively.

4.1 Parallelization of *RnaPredict*

4.1.1 A Classification of Parallel GAs

Cantú-Paz [5] identifies four major types of parallel GAs: single-population master-slave, multiple-population GAs, fine-grained GAs, and hierarchical hybrids.

Single-population master-slave GAs, as the name suggests, have a single population. The fitness evaluation is distributed among a set of slave processors, while the master node performs selection, crossover, and mutation. Here, the entire GA population is included in selection and crossover, thus this type is also known as a "global" parallel GA. Figure 4.1 illustrates a single-population master-slave GA.

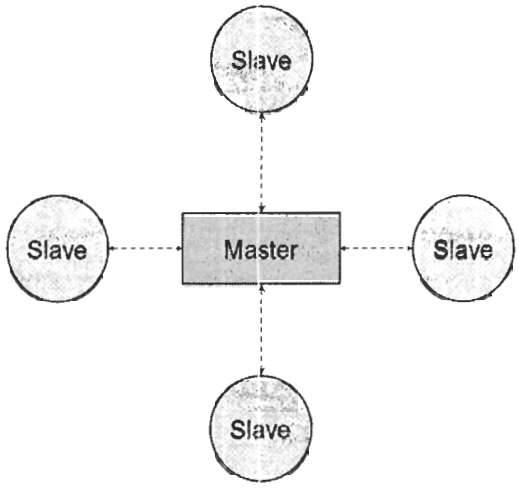

Figure 4.1: Illustration of a single-population master-slave GA with four nodes.

Multiple-population GAs, also known as multiple-deme GAs, consist of several subpopulations which occasionally exchange individuals through "migration." Migration is controlled by several parameters: interval, rate, policy, and topology. The migration interval determines when migration occurs. The migration rate determines how many individuals migrate. Migration policy determines which migrants are selected in the source deme and which individuals are replaced in the destination deme by them. Finally, the migration topology determines the source and destination demes of migrant individuals.

There is a variety of terminology for multiple-population GAs. When implemented on distributed-memory Multiple-Instruction Stream Multiple-Data (MIMD) computers, multiple-population GAs are known as distributed GAs. Since distributed GAs have a high ratio of computation to communication they are also referred to as coarse-grained. Coarse-grained distributed GAs fall into two classes: an island-model if demes are fully connected, or a stepping-stone model if migration is restricted between neighbouring populations. Figure 4.2 illustrates a distributed GA.

Fine-grained GAs typically distribute their population across a two-dimensional grid with one individual per grid point. Preferably one processor is allocated to each individual, which permits fitness evaluations to be performed simultaneously. Selection and crossover are constrained to the immediate neighbours surrounding an individual. They are also

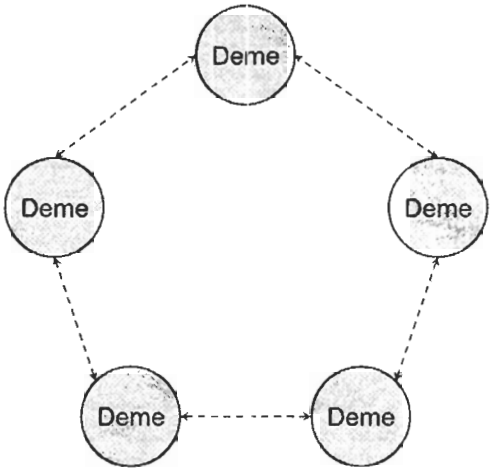

Figure 4.2: Illustration of a multiple-population GA with five nodes. The five nodes are connected in a stepping-stone model, in this case a bi-directional ring topology.

known as diffusion-model GAs and cellular GAs. One example includes the massively parallel GA for RNA secondary structure prediction developed by Shapiro et al. [77]. Figure 4.3 illustrates a fine-grained GA.

Finally, it is possible to combine GAs with multiple demes and master-slave or fine-grained GAs. Cantú-Paz [5] classifies this variety as hierarchical because they are multiple-deme algorithms at a high level with single-population parallel GAs at a low level. The notion is that hierarchical GAs combine the advantages of their components to produce potentially better performance than the individual components could produce separately.

4.1.2 Coarse-Grained Distributed GA

Coarse-grained distributed GAs [5] offer a number of advantages beyond the benefits of parallelization. These include the prevention of premature convergence by maintaining diversity, an increase of the selection pressure within the entire population, and also a reduction of the time to convergence. Another benefit of distributed GAs is that unlike single-population master-slave GAs, they need only communicate during migration. Thus, it was decided to investigate the multiple-deme or coarse-grained distributed GA. This development proceeded in two distinct steps. The first was to implement a serial simulation

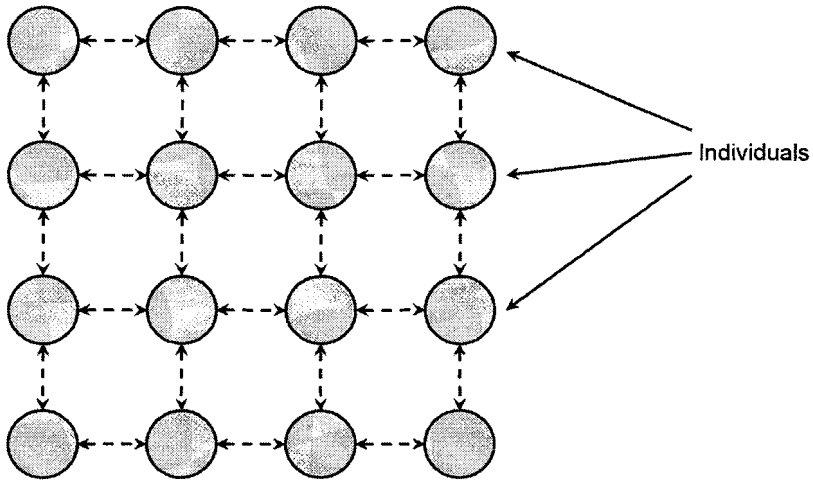

Figure 4.3: Illustration of a fine-grained GA with 16 nodes. The 16 nodes are connected in a mesh topology.

of the distributed GA [32, 33]. This was done to validate the potential benefits of the coarse-grained distributed GA in this problem domain without the overhead of implementing the potentially complex parallel communication code. Once the serial simulation was complete, the final stage was to implement a fully parallel version.

4.1.3 Distributed GA Configuration

The additional parameters necessary to configure the distributed GA are as follows:

- Deme Count
- Deme Size
- Migration Interval
- Migration Rate
- Migration Topology
- Migration Policy

"Deme Size" is the total number of individuals per deme. "Deme Count" is the total number of demes. "Migration Interval" is the number of generations between migration. "Migration Rate" is the percentage of a deme population selected for migration. "Migration Topology" determines the source and destination demes of migrant individuals; a typical topology is a fully connected set of demes. Finally, "Migration Policy" determines which individuals migrate from the source deme, and which are replaced in the destination deme.

With the parameter definitions established, we next review the pseudocode for the distributed GA in Figure 4.4. There are two significant differences between the *P-RnaPredict* pseudocode and the *RnaPredict* pseudocode in Figure 3.2. The first is the separation of the global population into individual demes. The second is in the additional logic to support migration at the end of the generational loop.

Through previous studies [33, 32] it was determined that overall the serial simulation of the distributed GA achieved comparable performance to the serial GA in terms of convergence behaviour and minimum free energy. With the viability of the serial simulation established, a fully parallel version was the next step.

4.2 The Case for Parallelization

At this stage, the viability of the coarse-grained distributed GA had been established through the serial simulation. Prior to parallelizing the serial simulation, it was necessary to determine whether a worthwhile speedup could be expected. In this section, the theoretical viability of the speedup is established, and important considerations such as available hardware and the software changes necessary to support parallelization are spelled out.

In simple terms, parallel computing is about simultaneously executing the same problem on multiple processors so that results may be obtained faster. This necessitates dividing up the original serial task and adapting it to execute concurrently on more than one processor. Assuming it were possible to perfectly divide the serial task requiring N time among p processors with no overhead for communication or pre-processing, the best time we could hope to obtain would be N/p.

Pancake [68] states that in considering whether to parallelize a given application one should consider three criteria: frequency of use, execution time, and resolution needs.

Frequency of use refers to how often the application is to be used before modifications are

```
Generate set of possible base pairs from sequence;
Generate set of possible helices using set of base pairs;
for each Deme do
| Initialize Random Population;
end
for all generations do
| for each Deme do
| | for deme size / 2 do
| | | Select two parents;
| | | if random value < P_c then
| | | | crossover parents to create two children;
| | | end
| | | for each child do
| | | | if random value < P_m then
| | | | | randomly mutate this child;
| | | | end
| | | end
| | | if selection strategy is KBR then
| | | | retain best parent and best child based on fitness;
| | | else if selection strategy is STDS then
| | | | always retain children;
| | | end
| | | insert retained individuals into new population;
| | | apply 1-Elitism and retain the best individual from the previous
| | | generation;
| | end
| end
| if generation falls on a migration interval then
| | for each Deme do
| | | select fraction (migration rate) of individuals based on policy;
| | | send selected individuals to other demes based on topology;
| | | replace individuals with incoming migrants based on policy;
| | end
| end
end
output single best structure from demes;
```

Figure 4.4: Distributed GA pseudocode

required. As this application is designed for research, it is likely that it will undergo substantial changes in the near future. However, coarse-grained distributed GAs are what is known colloquially as nearly "embarrassingly parallel" [100]. In this case, future modifications to the application will largely be independent of areas impacted by parallelization.

Execution time is simply the amount of time required to attain results. In the case of *P-RnaPredict*, this is directly related to the selected parameters and the length of RNA sequence considered.

Experiments were conducted with a serial simulation of the distributed GA on Nebula, the SFU Surrey 128 node Beowulf cluster. Each node's CPU is a 3 GHz Intel Pentium 4, and the nodes are interconnected with a Gigabit Ethernet network. The sequences tested were *Caenorhabditis elegans*, *Acanthamoeba griffini*, *Hildenbrandia rubra*, *Haloarcula marismortui*, and *Saccharomyces cerevisiae*; they are taken from the Comparative RNA Web Site [4], and their details are summarized in Tables 4.1, 4.2, 4.3, 4.4, and 4.5 respectively. The parameters for these runs are shown in Table 4.6. Table 4.7 shows runtimes from five sequences using identical parameters; each runtime is averaged over 30 randomly-seeded runs.

Table 4.1: *Caenorhabditis elegans* details, taken from the Comparative RNA Web Site [4]

Filename	d.16.m.C.elegans.bpseq
Organism	*Caenorhabditis elegans*
Accession Number	X54252
Class	16S rRNA
Length	697 nucleotides
# of BPs in known structure	189
# of non-canonical base pairs	23

Immediately it can be seen that the runtime tends to increases rapidly with the length of sequence under consideration. While it is inevitable that we wish to review longer sequences as research progresses, another important consideration is population size. Ideally, the population needs to be large enough to explore the solution space while small enough to avoid unnecessary computation. The solution space for *P-RnaPredict*'s structure prediction model is $2^{|H|}$, where $|H|$ is the cardinality of the set of all possible helices. For example, the *Sulfolobus acidocaldarius* (Table A.23) sequence has a length of 1494 nucleotides. Referring to Section 3.1, the helix generation algorithm finds 32274 possible helices resulting in a

Table 4.2: *Acanthamoeba griffini* details, taken from the Comparative RNA Web Site [4]

Filename	b.I1.e.A.griffini.1.C1.SSU.516.bpseq
Organism	*Acanthamoeba griffini*
Accession Number	U02540
Class	Group I intron, 16S rRNA
Length	556 nucleotides
# of BPs in known structure	131
# of non-canonical base pairs	1

Table 4.3: *Hildenbrandia rubra* details, taken from the Comparative RNA Web Site [4]

Filename	b.I1.e.H.rubra.1.C1.SSU.1506.bpseq
Organism	*Hildenbrandia rubra*
Accession Number	L19345
Class	Group I intron, 16S rRNA
Length	543 nucleotides
# of BPs in known structure	138
# of non-canonical base pairs	1

Table 4.4: *Haloarcula marismortui* details, taken from the Comparative RNA Web Site [4]

Filename	d.5.a.H.marismortui.bpseq
Organism	*Haloarcula marismortui*
Accession Number	AF034620
Class	5S rRNA
Length	122 nucleotides
# of BPs in known structure	38
# of non-canonical base pairs	4

Table 4.5: *Saccharomyces cerevisiae* details, taken from the Comparative RNA Web Site [4]

Filename	d.5.e.S.cerevisiae.bpseq
Organism	*Saccharomyces cerevisiae*
Accession Number	X67579
Class	5S rRNA
Length	118 nucleotides
# of BPs in known structure	37
# of non-canonical base pairs	2

Table 4.6: Serial simulation parameter settings from runtime experiments

Test Parameters	**Values**
Number of Generations	700
Crossover Probability (P_c)	0.7
Mutation Probability (P_m)	0.8
Selection Strategy	(Standard Roulette Wheel) STDS
Crossover Type	Cycle Crossover (CX)
Elitism	1
Thermodynamic Model	Individual Nearest Neighbour - Hydrogen Bond (INN-HB)
Deme Size	70
Deme Count	10
Migration Interval	20 generations
Migration Rate	10%
Migration Topology	Fully Connected, Island (FULL)
Migration Policy	Best Replace Worst

Table 4.7: Runtimes from serial simulation of distributed GA, averaged over 30 randomly-seeded runs

Organism	**Sequence Length**	**Serial Runtime (s)**
C. elegans	697	1264.4
A. griffini	556	754.0
H. rubra	543	871.9
H. marismortui	122	32.1
S. cerevisiae	118	27.5

search space of 2^{32274} possible structures. Thus, increasing the sequence lengths requires a corresponding increase in population size to properly explore the structure solution space, bringing with it a substantial increase in execution time. Aside from time complexity, another important issue is size complexity. Assuming that a permutation representation is employed with a population size of 500, the memory required to contain the individuals in the above example is:

32274 int (helices) $\times$ 4 bytes per int $\times$ 500 individuals = 64548000 bytes $\approx$ 61.6 MB

As population size and representation length increase, parallelization becomes one method of reducing the size complexity of the population.

The final criterion is resolution needs. For *P-RnaPredict*, this refers to the quality of the predictions that the GA is able to make. The results from all thermodynamic models, including INN and INN-HB [10, 11], indicate that there is no perfect correlation between matching true-positive base pairs in the known structure and minimum free energy. Two factors negatively impacting the accuracy of predicted structures include the fact that *P-RnaPredict*'s current thermodynamic models do not account for non-canonical base pairs, and are unable to account for pseudoknots. To improve the resolution of structure prediction will require a more advanced helix generation model and more detailed thermodynamic models, which in turn will increase the computational complexity.

Based on Pancake's criteria it appears that *P-RnaPredict* is a viable candidate for parallelization.

4.2.1 Potential Speedup

The potential for increased computational speed can be expressed by a measure known as the Speedup factor [99]. Speedup factor, $S(n)$, is a relative measure between the performance of a multiprocessor system and a single processor system:

$$S(n) = \frac{t_s}{t_p} \tag{4.1}$$

where t_s is the execution time on a single processor and t_p is the execution time on the multiprocessor. The absolute maximum speedup would be n with n processors, or linear speedup. This would be achieved by dividing the computation into processes running an identical duration of time, with no overhead and one process per processor:

$$S(n) = \frac{t_s}{t_p} = n \tag{4.2}$$

If $S(n)$ were to be greater than n, this would be what is known as superlinear speedup. It is known to occur occasionally, and can be attributed to employing a suboptimal sequential algorithm, or to having greater memory available to the processors. Although superlinear speedup is impossible for deterministic algorithms, it can occur for stochastic algorithms like parallel GAs. However, this is still controversial; one issue is that speedup and solution quality are considered separately in the literature. The important idea is that while migration in the distributed GA may lead to higher selection pressure and faster convergence, the GA must arrive at a comparable solution quality to a serial GA [5, 86].

In a theoretical analysis, the speedup factor could also be considered in terms of computational steps. It should be noted that this is intended to be a practical analysis; no GA-specific method for evaluating computational complexity independently of a representation has yet been developed [73]. If we simplify the computation by considering the time complexity of communications as 1 time unit per value sent, we can get a rough picture of the potential speedup:

d_c - deme count

d_s - deme size

G - total generations

l_{rep} - representation length

$t_{fit}(l_{rep})$ - fitness computation (dependent on representation length)

t_{sel} - selection time complexity (dependent on fitness computational complexity)

$t_{cross}(l_{rep})$ - crossover time complexity (dependent on representation length)

$t_{mut}(l_{rep})$ - mutation time complexity (dependent on representation length)

i_m - migration interval

r_m - migration rate

n - total number of processors

Reviewing the time complexity of the various phases of the GA:

$t_{pop\_init} = (d_c)(d_s)(t_{fit}(l_{rep}))$: covers the time complexity of population initialization. Since each individual created automatically computes its own fitness, $t_{fit}(l_{rep})$ accounts for the fact that the fitness computation's time complexity is both dominant and dependent on the representation length l_{rep}.

$t_{gen\_loop} = (G)(d_c)(d_s)(t_{sel}+t_{cross}(l_{rep})+t_{mut}(l_{rep}))$: covers the main GA generation loop. Selection, crossover, and mutation are executed once for each individual in the population, multiplied by the number of generations. Crossover and mutation are both dependent on the length of the representation l_{rep}.

$t_{comm\_overhead} = \frac{G}{i_m}(d_c)(t_{startup}+(r_m)(d_s)(l_{rep})(t_{data}))$: covers the cost of communication overhead. $t_{startup}$ is the startup time, or message latency; it is the cost of sending an empty message. t_{data} is the cost to send one data element, in this case one integer in the representation. Thus we have $r_m \cdot d_s \cdot l_{rep}$ data elements transmitted to d_c processors $\frac{G}{i_m}$ times over one GA run.

With these parameters established, it is possible to begin further analysis. Since the deme count (d_c) will be identical to the number of processors (n), those values will cancel in the parallel time complexity. Also, MPI's broadcast feature will be used instead of sending migrants serially to one deme process at a time. This reduces the time complexity of communications substantially.

$$S(n) = \frac{t_{pop\_init} + t_{gen\_loop}}{\frac{t_{pop\_init}}{n} + \frac{t_{gen\_loop}}{n} + t_{comm\_overhead}} \tag{4.3}$$

$$= \frac{t_{pop\_init} + t_{gen\_loop}}{(d_s)(t_{fit}(l_{rep})) + (G)(d_s)(t_{sel} + t_{cross}(l_{rep}) + t_{cross}(l_{rep})) + t_{comm\_overhead}} \tag{4.4}$$

The basic idea here is that for the parallelization of the distributed GA to be worthwhile certain conditions must be met. Specifically, the time consumed by migration through communication ($t_{comm\_overhead}$) must be substantially less than the processing time saved by subdividing the tasks of t_{pop_init} and t_{gen_loop} by d_c nodes. As typical migration rates used in *P-RnaPredict* are between 5 and 10 percent [32], the $t_{comm\_overhead}$ should be relatively small. Also, migration only occurs at comparatively rare intervals ($\frac{G}{i_m}$). This is a concrete example of why this type of GA is referred to as coarse-grained, with its high ratio of computation to communication.

Having established that parallelization of the distributed GA will theoretically provide a worthwhile speedup, the next issue is ensuring the computational resources are available to support *P-RnaPredict*'s parallel implementation.

4.2.2 Target Platform

Before a strategy for parallelization of the distributed GA can proceed, a major consideration is the hardware available on which the parallelized GA will be executed. Flynn [17] classifies computer hardware via four categories: single instruction stream-single data (SISD), multiple instruction stream-multiple data (MIMD), single instruction stream-multiple data (SIMD) and multiple instruction stream-single data (MISD). SISD refers to a single processor computer. MIMD refers to a multiprocessor system in which each processor executes a separate program and operates on different data. SIMD refers to hardware executing a single program upon multiple streams of data; the idea is to rapidly execute the same program on large arrays of data such as molecular simulations or image processing. Strictly speaking, MISD computers do not exist unless pipelined architectures are considered.

Given the available resources at SFU Surrey, the most likely candidate for an available platform is Nebula, a 128 node Beowulf Cluster with a Gigabit Ethernet network. This falls under the MIMD classification, with each separate workstation in the cluster having its own processor and own data. Within MIMD architectures it is possible to conceive of two different styles of programming structure: multiple program-multiple data (MPMD) and single program-multiple data (SPMD). Since the intent is to allocate one deme per processor, and each deme behaves as a serial GA between migrations, the implementation will be an identical deme program (single program) executing on a unique population (multiple data). Thus, the SPMD model was employed.

With the target platform and a parallel programming model established, the next consideration is the communications paradigm.

4.2.3 Message Passing

The current implementation of the serial simulation of the distributed GA has a 19,000 line GA C++ code base within more than 30 classes. This code in turn was based on a serial GA implemented by members of Dr. Kay Wiese's Bioinformatics lab; proper credit for this is detailed in section 4.4.2. A subset of these classes will be extended to construct the

parallelized prototype. The next major consideration is the message-passing standard to be employed; the choices are Parallel Virtual Machine (PVM) and Message-Passing Interface (MPI).

In PVM the problem is decomposed into separate programs written in C or Fortran and compiled for specific types of computers in the network. The set of computers to be utilized for a specific problem is defined in advance, forming the parallel machine. PVM permits any number of processes regardless of the number of processors available. Processes may be started dynamically from another process, and the standard message passing routines (send, receive, broadcast, scatter, gather, and reduce) are available and operate on predefined groups [99].

MPI offers routines for message passing similarly to PVM. Unlike PVM, there is no method of dynamically creating processes; they must be defined prior to execution and started together [99]. MPI applies the SPMD model; one program is written and executed by multiple processors. A key feature of MPI is its intent to provide a safe communications environment. To that end, MPI offers variations of basic send and receive, differentiated by whether they are locally or globally complete. Locally complete refers to whether the routine has completed its part of the communication operation; globally complete means that all routines must be locally complete. Also available are blocking send and receive routines, which return when they locally complete, and non-blocking routines, which return immediately.

MPI has four communication modes: standard, buffered, synchronous, and ready. In standard mode, the send routine does not assume that the corresponding receive routine has started. Buffered mode allows the send routine to start and return before the corresponding receive routine is reached. Synchronous mode requires that send and receive routines can only complete together. Finally, ready mode states that a send may only begin if the matching receive has already been reached.

Given that the demes of the distributed GA will start and remain in existence over the course of the GA execution, and the intent is to employ the SPMD parallel programming model, MPI is a logical standard to adopt. It is worth noting that MPICH [28] is a widely available implementation of the MPI standard. This is the implementation which will be used for the distributed GA.

With the message-passing technology established, we can focus on the anticipated implementation details in Section 4.3.

4.3 Implementation

Reviewing the pseudocode for the distributed GA, certain components may be immediately established under the SPMD model. First, each deme process must generate its own unique population. This presents an interesting challenge as the distributed GA will only have one random seed to initiate its pseudorandom number generator (PRNG). To succeed in creating a unique random population in each deme, a parallel PRNG method must be found and introduced to replace the original serial GA's PRNG. This is the first major modification.

After the initial deme population is created, the deme process will behave like an isolated serial GA, performing repeated iterations of selection, crossover, mutation, and replacement. The next area requiring redesign is the migration operator, which requires two major modifications to the current design. First, new control logic must be added to support migration within the SPMD model. Finally, a method must be developed to serialize the data contained in the migrants. This is necessary in order to utilize the MPI communication primitives for migration. The proposed modifications are described in detail below.

4.3.1 Random Number Generation

PRNGs are integral to the nature of GAs. In the original serial simulation implementation, a single random number generator could be used for every deme, since each deme was processed in succession on the same processor. However, this was no longer possible in the parallel implementation, whereby each unique processor required a unique stream of random numbers to process its deme and ensure statistical independence of the results.

Although originally thought to be a straightforward topic, there is a great deal of complexity in random number generation, especially for parallel applications. Thus, Chapter 5 in this thesis is devoted to parallel pseudorandom number generation and its impact on *P-RnaPredict*.

4.4 Control Logic

The first major modification is to rework the existing control logic to permit the separate demes to operate as individually as possible. The existing implementation already possesses *CDeme*, an object which contains all serial GA operations required in operating a single

deme. The first step in the serial GA implementation is to preprocess the given RNA sequence to determine all possible secondary structure elements (helices) which could form in feasible secondary structures. Since the Nebula Beowulf cluster permits concurrent access to data files stored on the master from the slaves, it is best to allow each deme to perform their own preprocessing as the helix generation is a deterministic algorithm. This is superior to making the slave nodes wait for the master to complete preprocessing and then transmit a comparatively large amount of data describing the helices and their conflicts. With initialization resolved, the next consideration is the migration logic. The pseudocode for this section is shown in Figure 4.5.

```
if this generation is a migration interval then
    if this node is the master node then
        generate a randomized list of all node IDs indicating the migration order;
        broadcast migration order to all slaves;
    else
        wait for migration order from master node;
    end
    for all nodes in migration node list do
        if current node in migration orderlist is this node then
            broadcast migrants to other nodes
        else
            receive migrants from the current node ID;
            insert new migrants into the population;
        end
    end
end
```

Figure 4.5: Control logic pseudocode

A walkthrough of the control logic pseudocode follows. Consider a *P-RnaPredict* run with 5 nodes, 4 slaves, and a master. Since each node keeps track of the current generation, all nodes are aware when a migration interval is about to occur. Slave nodes will halt on the migration generation, and wait for transmission of a randomized migration order from the master, node ID 0. The master node also halts on the migration generation, and creates a random permutation of the node IDs for all nodes; one example might be {4, 0, 3, 1, 2}. The list is randomized to ensure that certain demes are not given preference in migration simply due to numerical ordering of nodes. This permutation is then broadcast to all other nodes. Next all nodes, including the master, begin iterating through the list of node IDs. If the

node ID of a given node matches the current ID taken from the migration order, that node broadcasts its migrants to the other nodes. If the node ID of a given node does not match the current ID, it waits for the migration broadcast from the node matching the current ID. For example, given the list of {4, 0, 3, 1, 2}, the first node to broadcast its migrants would be node 4; all others would wait to receive node 4's migrants. The next node to broadcast is node 0, the master, and so on. This proceeds until all nodes have completed migration, whereupon normal generations are resumed.

Since the distributed GA is synchronous, it is necessary to exchange data in the form of migrants at the predetermined migration intervals. This necessitates the use of a barrier [99] to prevent the processes from proceeding until all migration is complete. In this case, rather than use an MPI barrier routine, the generation count can be used to determine when to halt each deme process for migration. Once migration is initiated, each process is able to act autonomously and is told when to broadcast or receive migrants based on the randomized node ID list broadcast by the master. This control structure permits the processors to act relatively independently and reduces the communication time required for migration. As soon as migration is complete, each deme process returns to individually computing the next generation for its deme.

4.4.1 Data Exchange for Migration

Once the control scheme was worked out for migration, another challenge was the exchange of the migrants themselves. In the original implementation, an object named *Individual* maintained all information about a given individual contained within the deme object. Fortunately, the actual "genetic" information stored within an *Individual* object is actually either a permutation of integer values or a string of bits. To make the exchange possible, a given processor which wishes to broadcast its migrants extracts the data from each *Individual* and packs it into a contiguous array of MPI_INT values for transmission. The receiving processor then unpacks the individuals, instantiates new Individual objects from each permutation, and inserts the new migrants into its population. The process of converting the object data to a data stream for transmission is referred as *data serialization.*

4.4.2 Development Process

The development of *P-RnaPredict* follows a steady evolution from the original research code base. As mentioned in Section 3.4, the initial implementation of *RnaPredict* included a serial simulation of the coarse-grained distributed GA. Work began on the foundation code, originally developed by Dr. Wiese and several research assistants. Several sets of C and C++ code captured the initial binary-based and permutation-based representation GAs; the code totalled roughly 4700 lines of code including comments.

It became clear that a unified development effort was necessary to ensure that the new code would meet the growing needs of the Bioinformatics research group. The three stakeholders involved in this effort were Alain Deschênes, Edward Glen, and myself. A new purely object-oriented design was laid out, emphasizing reusable components such as the C++ Standard Template Library (STL). The target platform was the Linux OS, and the development platform was the GNU development environment.

Of the roughly 18000 lines in the *RnaPredict* code-base, Alain Deschênes produced approximately 9500, Edward Glen added 3800, and I contributed 5039 lines of code. My main contributions were the container classes for the demes, the class which captured the RNA domain including sequence parsing, the helix generation algorithm, fitness computation, classes capturing the individual elements including base pairs and helices, and miscellaneous helper classes.

Once the serial simulation of the distributed GA had proven itself, the next stage was the final version of *P-RnaPredict*, the parallelization of the GA. This required a substantial revamp of the design, reusing some classes and combining the functionality of others to support the SPMD parallel paradigm. Aside from supporting parallel communication through MPI, an additional class to support the PRNG was also necessary. The end result was approximately 1000 lines of additional code to complete the implementation. Without the object-oriented design this would have required a great deal of time in rewriting code.

The source code was managed through the Concurrent Version System (CVS). This greatly facilitated the development process by ensuring developers could not accidently overwrite each other's code. The modular code supported easy testing for correctness, and also permitted profiling of modules to optimize the code-base.

In the initial stages of development, we made use of idle computing lab time during evenings and weekends on the SFU Surrey campus by converting the free machines into a

cluster. The tool we employed for this was the open source version of MOSIX [2], named OpenMosix. This was used to perform serial simulation runs on Nebula. Once the serial simulation gave way to the fully MPI-compliant implementation, I was fortunate to have the 128-node Nebula Beowulf cluster available; all *P-RnaPredict* runs were performed on it.

An important point is the series of shell scripts developed to manage, collate, and manipulate the large quantities of data generated by the experiments. The vast portion of development on these scripts was conducted by Alain Deschênes and Edward Glen, with minor modifications by myself to support the new parallel code-base.

4.5 Summary

This chapter has detailed the design and development of *P-RnaPredict*, a fully parallel coarse-grained distributed GA for RNA secondary structure prediction. An overview of parallel GA approaches has been provided, with the coarse-grained distributed approach selected as the most viable. A serial simulation of the distributed GA was developed, and runtime tests made the case for parallelization. Pancake's parallelization criteria were presented, and the distributed GA met all three criteria. A practical analysis of the expected parallel speedup was presented, and the target platform of the Nebula Beowulf cluster and MPI standard were established. The three primary implementation challenges of control logic, data serialization, and random number generation were summarized. Finally, the development process itself and the evolution from *RnaPredict* to *P-RnaPredict* were presented.

Chapter 5

Parallel Pseudorandom Number Generation

Random numbers are useful in many applications, and they are relied on extensively in *P-RnaPredict*. They can be used in simulations, sampling, numerical analysis, and decision making. In *P-RnaPredict*, random numbers are used to make coarse-grained decisions in population initialization, crossover, selection, mutation, and migration. This chapter covers a brief overview of PRNGs and their impact on GAs, methods of PRNG parallelization, *P-RnaPredict*'s requirements, and the results of a *P-RnaPredict* PRNG study.

5.1 PRNGs and Serial GAs

Although there appears to be very little in the literature regarding parallel GAs and PRNGs, several studies have been done on how serial GA performance is impacted by PRNGs. These are discussed below.

In 1997 [61], Meysenberg performed a thorough empirical study on the effect that twelve PRNGs of varying quality had on a simple GA using an eleven function real-value test suite. He found that PRNGs did not significantly impact this GA performance. In a later study [62], Meysenberg and Foster discovered isolated cases where poor PRNG quality resulted in slightly improved GA performance. Again, better PRNG quality failed to provide better GA performance.

In 1999, Meysenberg and Foster pursued what they referred to as their *granularity*

hypothesis [63]. In essence, a simple GA merely requires a PRNG to make choices between several options; this requires only that the PRNG produce a uniform distribution. Since even a poor quality PRNG can accomplish this, their theory was that its quality should not significantly impact GA performance. Their conclusions were that PRNG quality had no statistically significant effect on GA performance. However, this study was conducted on a generation by generation basis, and it revealed that GA performance could vary depending on the PRNG and test function chosen. The end result was that good PRNGs could actually result in poorer GA performance, and poor PRNGs could result in slightly better GA performance in isolated cases.

In 2002, Cantú-Paz performed an "ablation" study [6] where the individual GA components of initialization, selection, crossover, and mutation were separately tested. Both PRNGs and true random sources were tested. The results indicated that the PRNG used to initialize the random population is critical, whilst the other components were relatively unaffected. His conclusions were that the best PRNG available should be used to avoid misinterpretation of the results due to fortunate accidents.

With a basic notion of how PRNGS could impact GAs, the next step was determining the appropriate method for parallelizing the PRNG for the distributed GA.

5.2 Methods for Parallelization of PRNGs

Common methods of designing parallel random number generators include central server, cycle division, and parameterization.

The *central server* method establishes one process as a central random number server for all other processes in the parallel application. The immediate problem is the tremendous inter-process communications overhead, as each process must have exclusive access during its request to avoid conflicts. Another problem is that reproducibility becomes impossible to assure, as processes may make requests in different orders due to network traffic and the application implementation.

In *cycle division*, the period of a serial PRNG is subdivided amongst processors in one of three basic ways: *naïve* seed selection, cycle splitting, and "leap frog". In naïve seed selection, the user randomly chooses a different seed for each processor. The naïve hope here is that the portions of the PRNG period that each processor consumes are widely separated and do not overlap.

Another method, *cycle splitting*, involves the user carefully selecting the seeds to ensure they are widely separated. In this way, a contiguous block of random numbers from the serial PRNG can be assigned to each processor. However, if the processors consume too many random numbers, the period portions could again overlap.

Finally, there is the *leap frog* method. When given N processors, each processor gets numbers from the serial PRNG period which are N numbers apart. Here, the hazard is that long-range correlations in the serial PRNG become short-range correlations within each stream.

The problem with all these methods is that the resulting PRNG is non-scalable; each additional processor takes an equal share of the finite period of the original serial PRNG. Also, reproducibility becomes an issue as each additional processor results in a different serial PRNG period partition for all processors. Several studies have been done on serial PRNGs parallelized using cycle division [14, 51, 81, 82] which experimentally bear out these defects, especially for PRNGs such as linear congruential generators (LCGs) [45].

The parameterization method by contrast promises to provide independent and uncorrelated random number streams for each processor. There are two basic methods of parameterization [52]: seed parameterization and iterative function parameterization. Seed parameterization works on specific PRNGs for which each initial random seed automatically selects a smaller, separate, and independent period. A unique seed is assigned to each processor, ensuring each processor gets a unique period. The second method, iterative function parameterization, creates multiple independent random number streams by generating a different PRNG iteration function for each processor. The idea is that given a number i, the PRNG would generate a unique *ith* iteration function.

At the time of this writing, the best parallel PRNG available appears to be the parallel Mersenne Twister (MT), named "Dynamic Creation" (DC) [58]. DC implements iterative function parameterization, accepting a number of parameters including word size, period, working memory, and ID number. A small MT is then produced based on the submitted parameters. The key idea here is that the characteristic polynomial of the MT's linear recurrence encodes the specified ID number, ensuring a unique and highly independent PRNG for each ID.

5.3 PRNG Requirements of *P-RnaPredict*

As noted in Section 5.1, to date there appears to be little discussion in the literature on parallel GAs with regards to PRNGs. In this research, PRNGs became significant for a number of reasons. First, to gain an unbiased idea of the performance of *P-RnaPredict*, the results are averaged over 30 randomly-seeded runs. This implicitly assumes that the random numbers generated for each run are independent of each other. Second, the GA's consumption of random numbers has been rapidly increasing as structure prediction is performed on larger RNA sequences. This has reached a point where the periods of the PRNGs available in the standard C library are no longer adequate. Third, the parallelization of the initial serial simulation required a corresponding parallelization of whatever PRNG is used.

Cantú-Paz's ablation study underscores the importance of independent PRNGs during population initialization. An especially hazardous scenario occurs when parallel PRNG methods such as naïve seed selection or cycle splitting are used in a distributed GA. Consider an example of a distributed GA with two demes where the initial subpopulations are being generated. If each random chromosome in the initial population requires n random numbers, then the total amount of random numbers required to initialize each subpopulation is nm, where m is the population size. With a PRNG having a period of length p, each deme requires an independent section of that period of length nm to generate its initial population. The worst case scenario is if these sections overlap such that one section is offset to another by a multiple of the chromosome length n. Should this occur, identical chromosomes will be generated in the demes, greatly reducing the diversity within the parallel GA and possibly leading to diminished performance. This problem worsens with an increase in the number of demes. An example of this is shown in Figure 5.1. A segment of a PRNG period is shown with the initialization of four demes (A, B, C, and D). Note that the PRNG period sections of demes A and D overlap, and A is offset to D by a multiple of the chromosome length n. As a result, the last four individuals in deme D and the first four individuals in deme A are duplicates.

Based on these observations, two parallel PRNGs were selected for evaluation in the parallel GA implementation. The first was the DC PRNG described above. The second was a parallelized version of a MCG [45]; its parameters were $m = 2^{31} - 1$, $c = 0$, and $a = 6208991$ as suggested by [16]. This MCG was parallelized by the leap-frog method

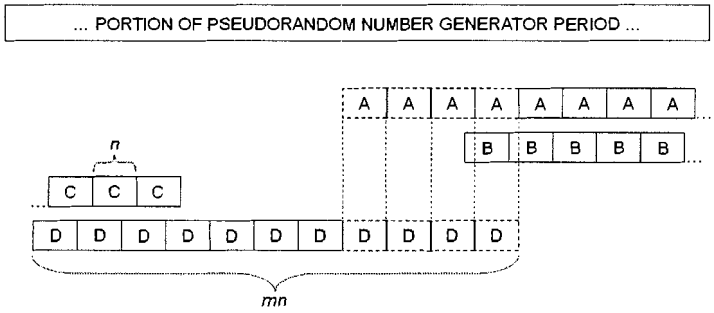

Figure 5.1: Example of demes generating duplicate individuals during initialization. Each box represents one individual, requiring n random numbers to initialize. Dashed boxes indicate duplicate individuals.

described in [19], and was deliberately chosen to have a lower quality and shorter period than the DC. Aside from the parallel PRNGs, it was also decided to check *P-RnaPredict*'s results against the original serial simulation of the distributed GA, which used the standard C library PRNG RAND [48].

5.4 PRNG GA Experiments

The purpose of these experiments was to determine the best PRNG to implement within *P-RnaPredict*. The evaluation criteria was the prediction accuracy of the final solutions both in terms of averaged performance over multiple randomly-seeded runs and the best results of individual runs. Given the enormous number of possible parameter combinations, the selection of parameter sets for these experiments was based on previous research [33, 95]. The global population was set to 700, with the crossover probability (P_c) set to 0.7. The mutation probability (P_m) varied as either 0.25 or 0.8. The selection strategy was set to STDS, and 1-Elitism [42] was also applied in all experiments. Finally, the INN-HB thermodynamic model and the CX [67] crossover were selected.

The parameters specifically relating to the distributed GA were chosen based on those which produced the best set of results in [33], and were set as follows: the global population was split into two separate sets of deme sizes and deme counts: 50 and 14, and 70 and 10 respectively. The migration interval and the migration rate were fixed at 20 generations

and 10 percent, respectively. Finally, the topology was fully connected, and the migration policy was set to "best replace worst." The parameter set for each experiment was repeated with 30 random seeds and the results averaged.

Table 5.1: Parameter settings for PRNG testing

Test Parameters	**Values**
Number of Generations	700
Crossover Probability (P_c)	0.7
Mutation Probability (P_m)	0.25, 0.8
Selection Strategy	(Standard Roulette Wheel) STDS
Crossover Type	Cycle Crossover (CX)
Elitism	1
Thermodynamic Model	Individual Nearest Neighbour - Hydrogen Bond (INN-HB)
Deme Count	10, 14
Deme Size	70, 50
Migration Interval	20 generations
Migration Rate	10%
Migration Topology	Fully Connected, Island (FULL)
Migration Policy	Best Replace Worst

Four RNA sequences were taken as test data from the Comparative RNA Web Site [4]; they were chosen to provide a good variety of sequence lengths and organisms. The sequences tested were *Acanthamoeba griffini*, *Hildenbrandia rubra*, *Haloarcula marismortui*, and *Saccharomyces cerevisiae*; their details are summarized in Tables 4.2, 4.3, 4.4, and 4.5 respectively.

Each set of parameters was tested with each of the three PRNGs. The first two were the DC and MCG generators detailed above. The third test was performed using the serial implementation of the distributed GA, which employed the GNU C standard library PRNG RAND. All runs were performed on the Nebula Beowulf cluster discussed in Section 4.2.2.

5.5 PRNG Experiment Results

The following sections present a summary of results for each RNA sequence. Parameters which do not vary between test runs have been omitted for brevity. "Deme Size" indicates the population of an individual deme. "P_m" indicates the probability of mutation. "Deme Count" indicates the number of demes used. "PRNG" indicates the type of PRNG used. "Avg. ΔG" is the free energy measured in kcal/mol, averaged over 30 randomly-seeded

runs. "Avg. Base Pair %" is the percent of base pairs which match the predicted structure, averaged over 30 randomly-seeded runs. Finally, "Best Base Pair %" is the percentage of matching base pairs from the run with the highest percentage out of the 30 randomly-seeded runs for that specified parameter set. Each table is sorted by Deme Size, P_m, and Avg. ΔG in order to group the results by parameter set. This is done to clearly delineate the performance differences between the three PRNGs in terms of the average final free energy reached. Rows containing best results are bolded.

5.5.1 *Acanthamoeba griffini* - 556 nt

Table 5.2 indicates that the MCG PRNG performed best in two of the parameter sets based on average free energy, with the DC and RAND PRNGs performing best in one parameter set each. Overall, the MCG PRNG reached the best average free energy at -190.79 kcal/mol with the following parameters: a Deme Size of 70, a Deme Count of 10, and a P_m of 0.8. The DC PRNG found the highest average percentage of base pairs matching the known structure at 32.34% with the following parameters: a Deme Size of 50, a Deme Count of 14, and a P_m of 0.25. The best overall structure was found with 64.88% matching base pairs with the following parameters: a MCG PRNG, a Deme Size of 70, a Deme Count of 10, and a P_m of 0.8.

5.5.2 *Hildenbrandia rubra* - 543 nt

In Table 5.3 we see that based on average free energy the DC PRNG performed best in two of the parameter sets, with the MCG and RAND PRNGs performing best in one parameter set each. Overall, the MCG PRNG reached the best average free energy at -207.08 kcal/mol with the following parameters: a Deme Size of 50, a Deme Count of 14, and a P_m of 0.8. The overall best structures matched 48.55% of the base pairs in the known structure, and were found in single runs from the following two parameter sets: the first was a MCG PRNG, Deme Size of 50, Deme Count of 14, and P_m of 0.8. The second was a RAND PRNG, with a Deme Size of 70, a Deme Count of 10, and a P_m of 0.25. Finally, the highest percentage of matching base pairs averaged over the 30 runs was 29.66%, and it occurred in a run set with the following parameters: a MCG PRNG, a Deme Size of 70, a Deme Count of 10, and a P_m of 0.8.

Table 5.2: Parallel GA results using three different PRNGs on the *A. griffini* sequence

Deme Size	P_m	Deme Count	PRNG	Avg. ΔG	Avg. Base Pair %	Best Base Pair %
70	0.25	10	DC	-187.58	28.39	58.77
70	0.25	10	MCG	-187.29	30.35	56.48
70	0.25	10	RAND	-186.35	27.04	46.56
70	**0.8**	**10**	**MCG**	**-190.79**	**29.79**	**64.88**
70	0.8	10	DC	-189.35	29.26	60.30
70	0.8	10	RAND	-187.80	28.39	60.30
50	0.25	14	RAND	-186.51	26.89	52.67
50	**0.25**	**14**	**DC**	**-184.74**	**32.34**	**58.01**
50	0.25	14	MCG	-184.43	28.04	48.09
50	0.8	14	MCG	-188.85	31.67	54.96
50	0.8	14	DC	-188.29	26.92	48.85
50	0.8	14	RAND	-185.27	27.17	47.32

Table 5.3: Parallel GA results using three different PRNGs on the *H. rubra* sequence

Deme Size	P_m	Deme Count	PRNG	Avg. ΔG	Avg. Base Pair %	Best Base Pair %
70	0.25	10	DC	-200.61	25.09	41.30
70	**0.25**	**10**	**RAND**	**-199.65**	**24.32**	**48.55**
70	0.25	10	MCG	-198.76	24.44	40.57
70	0.8	10	DC	-204.17	26.28	47.82
70	0.8	10	RAND	-203.66	27.58	46.37
70	**0.8**	**10**	**MCG**	**-203.14**	**29.66**	**44.92**
50	0.25	14	RAND	-200.64	27.19	44.20
50	0.25	14	DC	-199.05	24.42	41.30
50	0.25	14	MCG	-198.83	26.81	41.30
50	**0.8**	**14**	**MCG**	**-207.08**	**27.75**	**48.55**
50	0.8	14	RAND	-202.93	26.64	45.65
50	0.8	14	DC	-199.23	22.89	38.40

5.5.3 *Haloarcula marismortui* - 122 nt

In Table 5.4 we can see that based on average free energy the DC and RAND PRNG tied for best performance in three out of the four parameter sets, with the RAND PRNG performing best in the fourth parameter set. Overall, the DC and RAND PRNGs both reached the best average free energy at -54.94 kcal/mol with the following identical parameters: a Deme Size of 70, a Deme Count of 10, and a P_m of 0.8. However, the DC PRNG edged out the RAND with the best averaged base pair percentage of 42.10%. For this sequence, the overall best structure was found matching 71.05% of base pairs in the known structure with the following parameter set: a MCG PRNG, a Deme Size of 70, a Deme Count of 10, and a P_m of 0.25.

Table 5.4: Parallel GA results using three different PRNGs on the *H. marismortui* sequence

Deme Size	P_m	Deme Count	PRNG	Avg. ΔG	Avg. Base Pair %	Best Base Pair %
70	0.25	10	RAND	-54.93	38.59	42.10
70	0.25	10	DC	-54.93	39.47	42.10
70	**0.25**	**10**	**MCG**	**-54.88**	**39.56**	**71.05**
70	**0.8**	**10**	**RAND**	**-54.94**	**41.22**	**42.10**
70	**0.8**	**10**	**DC**	**-54.94**	**42.10**	**42.10**
70	0.8	10	MCG	-54.93	39.47	42.10
50	0.25	14	RAND	-54.92	37.71	42.10
50	0.25	14	DC	-54.92	36.84	42.10
50	0.25	14	MCG	-54.91	35.08	42.10
50	0.8	14	RAND	-54.93	40.35	42.10
50	0.8	14	MCG	-54.92	35.96	42.10
50	0.8	14	DC	-54.92	37.71	42.10

5.5.4 *Saccharomyces cerevisiae* - 118 nt

All runs for the *Saccharomyces cerevisiae* RNA sequence converged to identical free energy values and secondary structures, regardless of parameter settings or the chosen PRNG (Table 5.5). The overall results were as follows: The average fitness was -57.52 kcal/mol;

Table 5.5: Parallel GA results using three different PRNGs on the *S. cerevisiae* sequence

Deme Size	P_m	Deme Count	PRNG	Avg. ΔG	Avg. Base Pair %	Best Base Pair %
ALL	ALL	ALL	ALL	-57.52	89.18	89.18

the best average matching base pair and the best overall matching base pair percentages were 89.18%. The prediction accuracy for this sequence was very high.

5.6 Summary

After reviewing the results from the four sequences, it appears that the parallel implementation of the distributed GA (*P-RnaPredict*) performs comparably to the original serial simulation of the distributed GA (*RnaPredict*). Considering the PRNG studies reviewed [16, 59, 80], the PRNG quality of the generators tested, based on period size and spectral tests, should rank in order of DC, MCG, and RAND from best to worst respectively. Of note is that the MCG PRNG produced the highest percentage of matching base pairs in 2 out of the 4 sequences, also tying for the highest in 2 other sequences. It is interesting that the differences in performance between the two parallel PRNGs and the original serial GA PRNG do not appear to be significant. This reinforces the findings of the previous serial GA studies discussed in Section 5.1. However, the serial version of RAND cannot easily be parallelized.

Clearly, care needs to be taken when parallelizing PRNGs, especially in the case detailed in Section 5.3 where the period of the PRNG is insufficiently large. Based on these results and the PRNG literature review, it appears best to implement the best PRNG available at this time, the DC. Hence, all results presented in this thesis after this point use DC.

Chapter 6

Parallel Speedup Validation

Along with testing the PRNG performance, another series of experiments was performed to investigate speedup. In the following chapter the results of these experiments are presented.

6.1 Speedup Experiment Design

Two separate sets of experiments were performed. The first is a high-level time study comparing total runtimes with five different sequences. The second is a detailed analysis with the computation and communication times from one sequence.

Table 6.1 presents the parameters which were fixed for both experiment sets. These were determined experimentally to be among the best performing runs in the serial simulation of the distributed GA [33, 95].

6.2 High-Level Runtime Test Results

Five sequences of varying lengths were tested to verify speedup: *Caenorhabditis elegans*, *Acanthamoeba griffini*, *Hildenbrandia rubra*, *Haloarcula marismortui*, and *Saccharomyces cerevisiae*; they are taken from the Comparative RNA Web Site [4], and their details are summarized in Tables 4.1, 4.2, 4.3, 4.4, and 4.5 respectively.

The deme size and deme count were varied between two sets of values: (50, 14) and (70, 10). Each experiment was performed with 30 randomly-seeded runs. Table 6.2 compiles the results of the speedup test runs; its column descriptions are as follows:

Table 6.1: Fixed parameter settings in both sets of parallel speedup tests.

Test Fixed Parameters	Values
Number of Generations	700
Global Population Size	700
Crossover Probability (Pc)	0.7
Mutation Probability (Pm)	0.8
Selection Strategy	(Standard Roulette Wheel) STDS
Crossover Type	Cycle Crossover (CX)
Elitism	1
Thermodynamic Model	Individual Nearest Neighbour - Hydrogen Bond (INN-HB)
Migration Interval	20 generations
Migration Rate	10%
Migration Topology	Fully Connected, Island (FULL)
Migration Policy	Best Replace Worst

"Organism" is the name of the organism from which the sequence is taken. "Sequence Length" is the length of the specific sequence in nucleotides (nt). "Deme count" is the total number of demes in the GA; note that each deme is assigned to a single processor. "Serial Runtime (s)" is the total runtime of the serial simulation of the distributed GA in seconds, averaged over 30 randomly-seeded runs. "Parallel Runtime (s)" is the total runtime of the fully parallel distributed GA in seconds, averaged over 30 randomly-seeded runs. "Speedup $S(n)$" is the speedup factor S(n), defined in Equation 4.1.

Finally, "Efficiency" is the system efficiency, defined as: E = execution time using one processor divided by (execution time using a multiprocessor $\times$ number of processors) [99]:

$$E = \frac{S(n)}{n} \times 100\% \tag{6.1}$$

6.2.1 High-Level Time Test Discussion

The results in 6.2 are split between the deme counts of 10 and 14 for clarity, and sorted by sequence length. A clear improvement can be seen in Speedup and Efficiency as the sequence lengths increase. Note that the differing deme counts result in differing relationships with the Efficiency. For example, the *C.elegans* entry with 10 demes and a Speedup of 6.3 results in an Efficiency of 62.5%, while the corresponding 14 deme entry has a Speedup of 7.6 but results in an Efficiency of 54.4%.

The *H.rubra* sequence runtimes are notable; their deviation can be accounted for by

Table 6.2: Results from multiple sequence parallel speedup test runs

Organism	**Sequence Length**	**Deme Count**	**Serial Runtime (s)**	**Parallel Runtime (s)**	**Speedup** ($S(n)$)	**Efficiency** (%)
C.elegans	697	10	1264.4	202.4	6.3	62.5%
A.griffini	556	10	754.0	127.8	5.9	59.0%
H.rubra	543	10	871.9	145.0	6.0	60.1%
H.marismortui	122	10	32.1	11.5	2.8	27.9%
S.cerevisiae	118	10	27.5	10.9	2.5	25.2%
C.elegans	697	14	1255.8	165.0	7.6	54.4%
A.griffini	556	14	759.3	106.8	7.1	50.8%
H.rubra	543	14	870.9	203.1	4.3	30.6%
H.marismortui	122	14	31.6	11.3	2.8	20.0%
S.cerevisiae	118	14	27.2	10.9	2.5	17.8%

reviewing Table 3.1. Although *H.rubra* is a shorter sequence than *A.griffini*, the *H.rubra* sequence generates more helices under *P-RnaPredict*'s model, and thus has a longer representation. Based on the approximate time complexity analysis in section 4.2.1, we can see that it is the representation length which is influencing the runtime.

It is also possible to see that there is not a linear relationship between representation length and runtime. For shorter sequences, the Speedup and Efficiency appears low compared to the number of processors available (10 or 14). This is occurring because the time required to process one generation for these shorter sequences is significantly shorter than the time consumed through communication-related activities such as migration. As the sequences increase in length, there is a corresponding increase in the representation length. Thus, the amount of time spent in computation in an individual deme considerably exceeds the time spent in communication, and the Speedup factor and Efficiency improve dramatically.

6.3 Detailed Runtime Results

The parameter sets tested here were selected to illustrate reasonable usage of the distributed GA. The *Hildenbrandia rubra* sequence was chosen for testing. It was necessary to select deme counts which divided evenly into the global population of 700. Even-size deme populations were necessary as crossover in the GA works on a pairwise basis. A lower bound on the

deme population size of 50 was selected based on previous empirical results which indicated lower deme population sizes performed poorly. Although subdividing the global population is useful in illustrating the parallel speedup, it is important to note that a distributed GA is a different algorithm than a serial GA, and may require a total population larger than the original serial GA to be effective [5].

The parameters which varied were deme size and deme count, and were set to the values shown in Table 6.3.

Table 6.3: List of deme size and deme count settings in Parallel Speedup Test Runs

Deme Count	Deme Size
2	350
5	140
7	100
10	70
14	50

Each parameter set was tested with 5 random seeds, and the results averaged over the runs. Table 6.4 compiles the results of the speedup test runs; its field descriptions are as follows:

"Deme Count" is the total number of demes in the GA. "Averaged Communication Time (seconds)" is the total time spent in communications in the parallel implementation of the GA, averaged over 5 randomly-seeded runs. "Averaged Parallel Time (seconds)" is the total runtime of the parallel implementation of the GA, averaged over 5 randomly-seeded runs. "Averaged Serial Time (seconds)" is the total runtime of the serial implementation of the GA, averaged over 5 randomly-seeded runs. "Percent Communication" is the percentage of the parallel implementation runtime spent in communications. "Speedup $S(n)$" is the speedup factor S(n), defined in Equation 4.1. Finally, "Efficiency" is the system efficiency, defined in Equation 6.1.

6.3.1 Detailed Time Analysis

Figure 6.1 plots serial versus the parallel runtime. Interestingly, there is a drop in the serial implementation's runtime as more demes are added. One possible reason for this speedup is the faster convergence speed resulting from the increasingly smaller deme populations. When crossover operators such as Cycle Crossover (CX) operate on parents with the same

Table 6.4: Results from deme size and deme count parallel speedup test runs

Deme Count	Averaged Communication Time (s)	Averaged Parallel Time (s)	Averaged Serial Time (s)	Percent Communication	Speedup $S(n)$	Efficiency
2	30.5	649.4	1258.8	4.7%	1.9	96.9%
5	38.1	252.0	1060.6	15.1%	4.2	84.2%
7	38.1	182.2	1013.8	20.9%	5.6	79.5%
10	40.1	149.2	983.6	26.9%	6.6	65.9%
14	40.6	118.8	1008.6	34.2%	8.5	60.6%

chromosomes, they return children identical to the parents and their execution ends almost immediately. This means the runtime for each generation drops dramatically as the population converges to one chromosome, and this convergence acceleration is directly proportional to the number of demes.

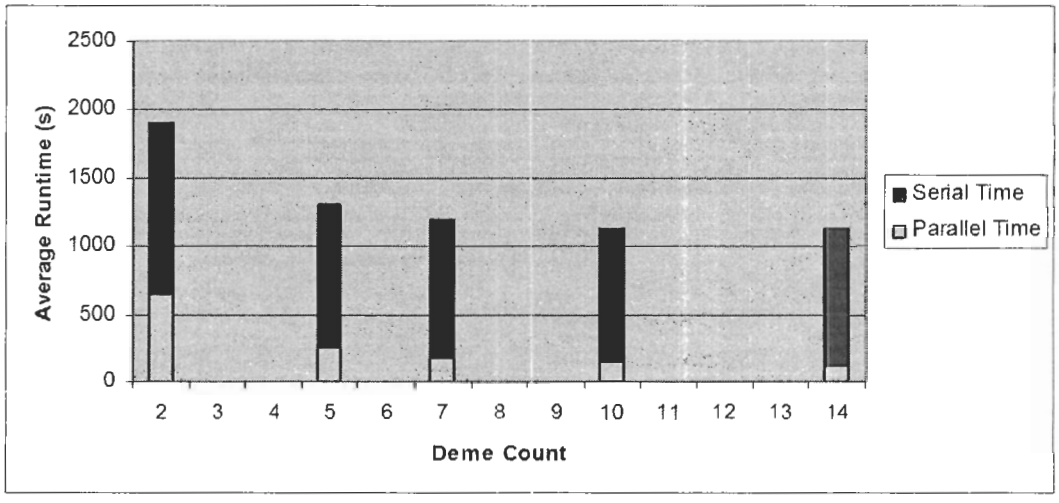

Figure 6.1: Plot of average runtimes of the serial simulation and parallel implementation of distributed GA against deme count.

P-RnaPredict's runtimes indicates the "pleasingly" or nearly embarrassingly parallel nature of a distributed GA. Each time the deme count is increased, the global population of 700 is subdivided into smaller populations which may be processed in parallel. The immediate impression is that parallelization greatly reduces the execution time.

Nevertheless, it is still necessary to both exchange migrants on each migration interval and send statistics to the master node after each generation. Consequently, a less than n-fold speedup is possible. This can clearly be seen in Figure 6.2, which plots the speedup factor, defined in Equation 4.1.

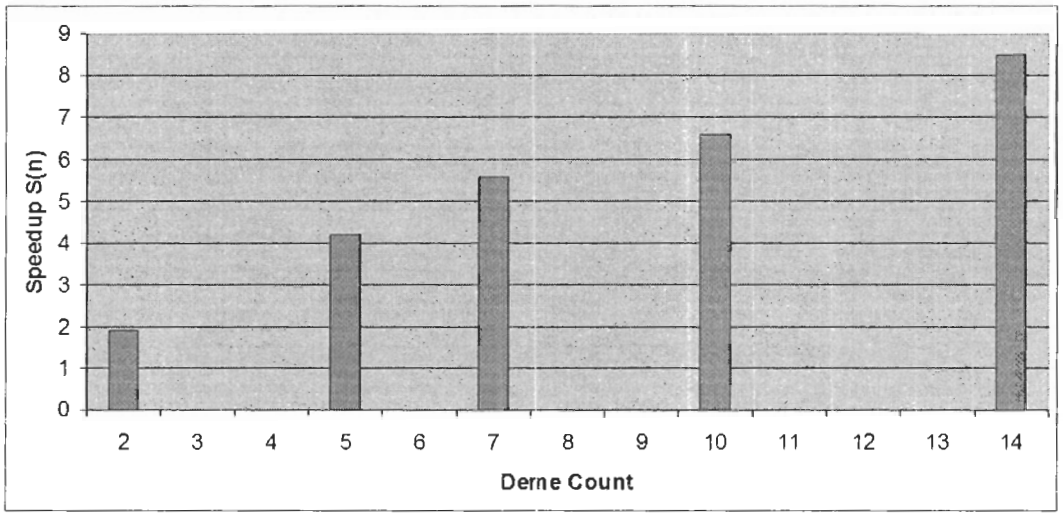

Figure 6.2: Plot of speedup factor $S(n)$ against deme count.

Figure 6.2 also shows the inevitable decline in speedup which results from the increased communications expense involving additional demes. This can be seen in even better detail in Figure 6.3, which plots the parallel runtime against the communication time. Communication is necessary for both migration and statistics reporting. Migration requires that for every migration interval, the chromosomes of the migrants from each deme are packed into a contiguous array and broadcast to every other deme. The statistics collected at this time are the global population's minimum and maximum free energy, and the mean and standard deviation. Since all individual fitness values are required for these statistics, all fitness values from each deme must be sent to the master every generation.

It is interesting to note that the communications overhead does not appear to be increasing significantly with the increase in deme count. One possible reason for this is the extensive use of MPI's broadcast functionality, which when used with non-blocking functions greatly reduces communications overhead. However, the Speedup and Efficiency continue to diminish with each additional node. This occurs because the communication time steadily

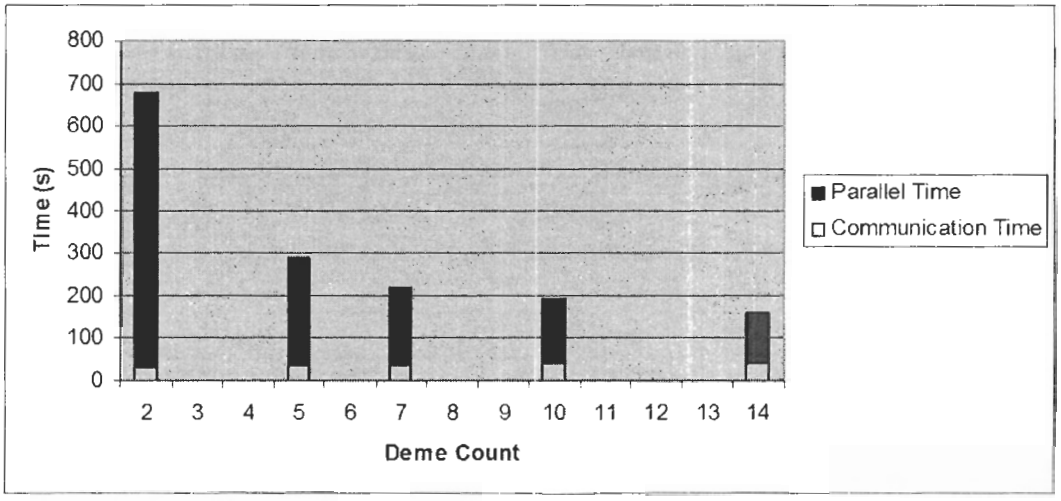

Figure 6.3: Plot of computational time and communications time versus deme count.

consumes a greater portion of the total parallel time with each node added, diminishing the overall returns.

One final point of interest was found when investigating the communication time on a node-by-node and function-by-function basis. There was a dramatic disparity between the master node and the other nodes in the distributed GA. Table 6.5 breaks down the various communications costs for the timing run with a deme count of 14; similar time ratios were noted for other deme counts. Table 6.5 field descriptions are as follows:

"Total Statistics Time(s)" is the total time in seconds for the node to either send or receive the statistics data. "Migration Order (s)" is the total time in seconds for the node to either send or receive the migration order. "Migrant Broadcast (s)" is the total time in seconds for the node to either send or receive the migrants during migration. "Total Migration Time (s)" is the total time required in seconds for the node to send or receive all data required for migration, and is the sum of the Migration Order and Migrant Broadcast times. "Total Communication Time (s)" is the total time required by a node for communications, and is the sum of the Total Statistics and Total Migration times.

Examining the Total Statistics Communication time, we see the master node requires much more time to receive the statistics data than the other nodes required to send it. One possible explanation for this is that the master node must perform multiple individual MPI receive calls to gather the fitness values from each of the nodes, while each node only needs

Table 6.5: Breakdown of averaged communication times by function and node type for a deme count of 14.

	Total Statistics Time (s)	**Migration Order (s)**	**Migrant Broadcast (s)**	**Total Migration Time (s)**	**Total Comm. Time (s)**
Master Node Average Time (s)	4.624	0.164	35.042	35.206	39.830
Other Nodes Average Time (s)	0.024	5.480	35.159	40.260	40.613

to send its data once with a non-blocking send and can immediately begin its next iteration. Thus, the master node could be stuck in a wait state, as each array of fitness values must be received from the node in order.

The second area for concern is migration. Specifically, the master node must randomize the node order for each migration, and broadcast that order to all other nodes. The block of data required for migration order is only an array of integers whose length is equal to the number of demes/nodes in the GA, in this case 14 integers. This seems dramatically out of proportion to the time required for statistical communication, which is significantly larger due to a greater volume of data transmitted. If MPI's broadcast functionality is non-blocking, why is the time required for the slave nodes to receive the migration order over an order of magnitude greater than the time required for the master to send it? The answer could be related to MPI caching behaviour, but more analysis is required before a definitive answer can be given.

6.4 Summary

In conclusion, the timing data suggests that at the very least that the parallelization is a viable method of drastically reducing the overall execution time for the distributed GA. The distributed GA is unique in that a single problem is not subdivided among a set of processors but rather each processor contributes one deme to the GA as a whole. Consequently, more work must be done to determine deme sizes and counts which contribute to

the highest solution quality. Especially interesting is the communication overhead disparity found between the master and the slave nodes. The high cost of transmitting the migration order from the master to the slaves is excessive and should be eliminated if possible.

Chapter 7

Comparison to Known Structures

This chapter presents a qualitative measure of *P-RnaPredict*'s performance through a comparison between the structures predicted by *P-RnaPredict* and known structures. The known structures are determined via comparative methods, and are taken from the Comparative RNA Web Site [4]. The primary criteria for comparison is true-positive matching base pairs; a higher matching base pair count indicates a higher solution quality. Out of the ten structures evaluated, five structure comparisons are reviewed in depth here; the remainder may be examined in the Appendix A.

The parameters for these experiments (see Table 7.1) were chosen based on prior research [33, 95] and were set as follows: The GA was iterated for 700 generations. The crossover probability (P_c) was fixed at 0.7, and the mutation probability (P_m) was fixed at 0.8. Prior experiments using 1-Elitism indicated the standard roulette wheel selection (STDS) worked best in this domain compared to KBR, so all runs presented here use STDS. The INN and INN-HB thermodynamic models, and the OX2 [83] and CX [67] crossovers were selected. Three separate sets of deme sizes and deme counts were employed: (50, 14), (70, 10), and (100, 10) respectively. For the final case, the total population of all demes ends up as 1000 individuals; this was employed to determine if premature convergence was a factor. If the GA found consistently better solutions with a larger population size, it would suggest that smaller deme sizes were converging too quickly. The migration interval and the migration rate were fixed at 20 generations and 10 percent, respectively. Finally, the topology was fully connected, and the migration policy was set to "best replace worst." The parameter set for each experiment was repeated with 30 random seeds and the results averaged.

Table 7.1: Parameter settings for PRNG testing

Test Parameters	Values
Number of Generations	700
Crossover Probability (P_c)	0.7
Mutation Probability (P_m)	0.8
Selection Strategy	STDS
Crossover Type	CX, OX2
Elitism	1
Thermodynamic Model	INN, INN-HB
Deme Count	10, 10, 14
Deme Size	100, 70, 50
Migration Interval	20 generations
Migration Rate	10%
Migration Topology	Fully Connected, Island (FULL)
Migration Policy	Best Replace Worst

7.1 Convergence Behaviour of *P-RnaPredict*

Figure 7.1 plots the free energy from a typical *P-RnaPredict* run for the *Hildenbrandia rubra* sequence. These results are from the entire population, averaged over 30 randomly-seeded runs. The parameters used were a deme size of 100, a deme count of 10, OX2 crossover, and the INN-HB thermodynamic model. The graph indicates the maximum and minimum free energies present as the lighter outer envelope, and the mean free energy of the population with standard deviation is shown as the darker, inner envelope.

The best free energy value reached after 700 generations was -217.03 kcal/mol. Note the rapid initial drop in minimum free energy up until approximately generation 120, where the minimum free energy begins to plateau. There is also the cyclic behaviour visible every 20 generations, where the maximum (worst) free energy drops dramatically. This was clearly reflected across the population, as the hump in the standard deviation indicates. These migration triggered fitness bumps are visible through all distributed GA runs employing the STDS selection strategy. The broad standard deviation indicates diversity was preserved through all generations, likely due to the high mutation rate and low selection pressure of STDS.

Table 7.2: *Xenopus laevis* details, taken from the Comparative RNA Web Site [4]

Filename	d.16.m.X.laevis.bpseq
Organism	*Xenopus laevis*
Accession Number	M27605
Class	16S rRNA
Length	945 nucleotides
# of BPs in known structure	251
# of non-canonical base pairs	22

Table 7.3: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Xenopus laevis* structure. Each row represents an experiment consisting of 30 averaged runs. Results are grouped by thermodynamic model. The known *Xenopus laevis* structure contains 251 base pairs.

ΔG (kcal / mol)	**Pred. BPs**	**Corr. BPs**	**Corr. BPs (%)**	**Cross.**	**Deme Size**	**Deme Count**	**Model**
-274.25	241.0	**65.7**	**26.2**	OX2	100	10	INNHB
-268.06	238.1	59.3	23.6	CX	100	10	INNHB
-267.26	237.4	56.5	22.5	OX2	70	10	INNHB
-263.45	236.2	53.7	21.4	OX2	50	14	INNHB
-259.23	236.0	47.4	18.9	CX	70	10	INNHB
-255.23	233.5	48.2	19.2	CX	50	14	INNHB
-268.4	243.0	**61.9**	**24.7**	OX2	100	10	INN
-258.7	240.6	52.9	21.1	OX2	70	10	INN
-254.9	238.2	54.6	21.8	OX2	50	14	INN
-254.1	236.1	51.3	20.4	CX	100	10	INN
-250.2	238.1	50.0	19.9	CX	70	10	INN
-248.5	236.9	49.2	19.6	CX	50	14	INN

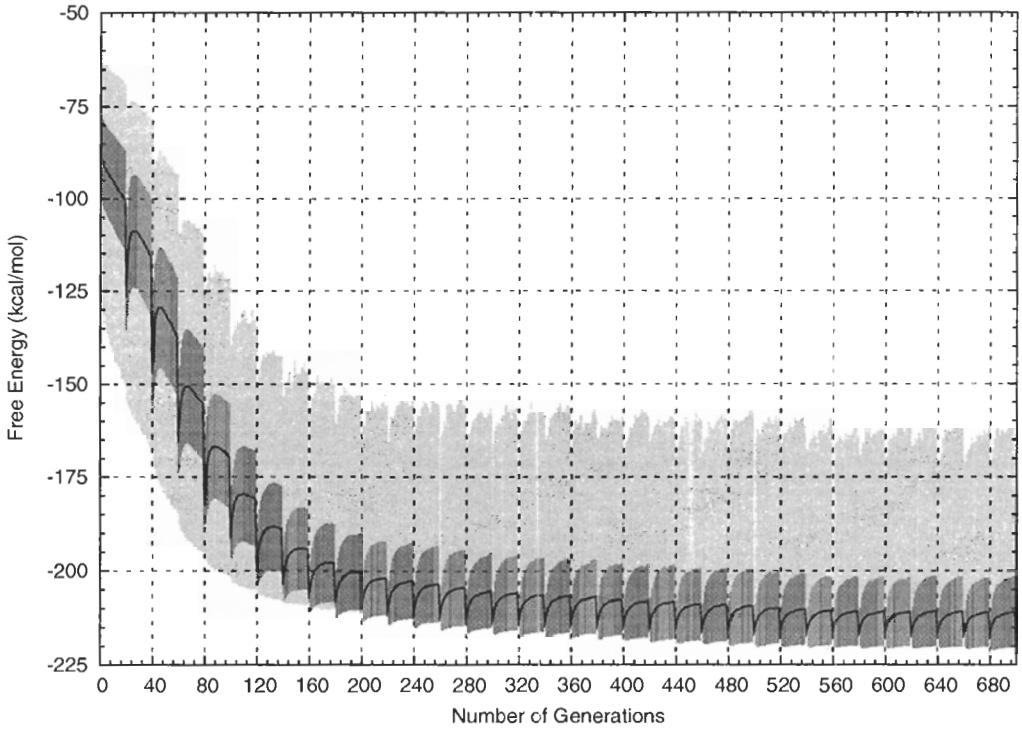

Figure 7.1: *Hildenbrandia rubra*, $P_m = 0.8$, $P_c = 0.7$, deme size = 100, deme count = 10, OX2, STDS, 1-elitism, and the INN-HB thermodynamic model. This plots the free energy of the entire population, averaged over 30 randomly-seeded runs. This parameter set predicted 158.0 base pairs, where 31.7% of the known structure was correctly predicted.

7.2 *Xenopus laevis* - 945 nt

Table 7.3 presents a comparison of the lowest average ΔG *P-RnaPredict* structures from each experiment with the known *Xenopus laevis* structure. Each experiment is conducted with 30 random seeds, and all results are averaged over these 30 runs. Table column headings are defined as follows: "ΔG (kcal / mol)" is the free energy of the average lowest energy structure for a given experiment. "Pred. BPs" is the average number of predicted base pairs. "Corr. BPs" is the average number of base pairs matching the known structure. "Corr. BPs (%)" is the average percentage of base pairs matching the known structure. "Cross." is the specific crossover operator used in the experiment. "Deme Size" is the total number of individuals per deme in the experiment. "Deme Count" is the total number of

demes in the experiment. Finally, "Model" is the thermodynamic model employed in the experiment.

Each experiment is organized by its parameters: crossover, deme size, deme count, and thermodynamic model. Parameters which do not vary between runs have been removed for clarity. Results are sorted by averaged ΔG value; since ΔG values are incompatible between INN and INN-HB, the results are split by thermodynamic model. The best values for "Corr. BPs" and "Corr. BPs (%)" are bolded. With 12 experiments, and 30 randomly-seeded runs per experiment, results are presented from 360 runs.

Table 7.3 indicates that for both INN and INN-HB the lowest ΔG structures were reached with the following parameter set: an OX2 crossover, a deme size of 100 and a deme count of 10. The best INN-HB experiment found structures with an average ΔG of -274.25 kcal/mol, 241.0 total base pairs, and matching 26.2% of base pairs in the known structure. The best INN experiment found structures with an average ΔG of -268.4 kcal/mol, 243.0 total base pairs, and matching 24.7% of base pairs in the known structure.

Overall, the INN-HB thermodynamic model performed best in terms of true-positive matching base pairs when considering the averaged results of the minimum free energy experiment. The larger deme sizes, such as 100, also performed significantly better than smaller ones.

Table 7.4 presents a comparison of the single lowest ΔG *P-RnaPredict* structure from each experiment with the known *Xenopus laevis* structure. Should multiple runs tie for lowest ΔG in a given experiment, the results for that experiment are averaged.

Table column headings are identical to those in Table 7.3, with two additions: "Freq." is the number of runs this ΔG value appears in a given experiment. "Gens" is the number of generations that the GA run required to reach the lowest ΔG value. Again, all data is split by thermodynamic model and sorted by ΔG.

Table 7.4 indicates that the best INN-HB run in terms of minimum ΔG reached a ΔG of -295.98 kcal/mol, 248 total base pairs, and matching 27.9% of base pairs in the known structure. The INN-HB run's parameters were a CX crossover, a deme size of 70, and a deme count of 10. However, the best INN-HB structure in term of correct base pairs matched 36.7% of base pairs in the known structure, with a ΔG of -293.97 kcal/mol and 243 total base pairs. This run employed an OX2 crossover, a deme size of 100, and a deme count of 10. For INN, the best run reached a ΔG of -290.7 kcal/mol, 100 total base pairs, and matching 39.8% of base pairs in the known structure. The run's parameters were an OX2

Table 7.4: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Xenopus laevis* structure. Results are grouped by thermodynamic model. The known *Xenopus laevis* structure contains 251 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-295.98	1	595	248	70	27.9	CX	70	10	INNHB
-293.97	1	562	243	**92**	**36.7**	OX2	100	10	INNHB
-289.02	1	590	238	76	30.3	OX2	50	14	INNHB
-286.23	1	590	239	65	25.9	OX2	70	10	INNHB
-282.86	1	630	240	70	27.9	CX	100	10	INNHB
-278.18	1	559	247	60	23.9	CX	50	14	INNHB
-290.7	1	670	258	**100**	**39.8**	OX2	100	10	INN
-279.7	1	669	248	75	29.9	CX	50	14	INN
-275.5	1	392	243	70	27.9	OX2	70	10	INN
-272.7	1	683	250	94	37.5	CX	100	10	INN
-270.0	1	662	241	80	31.9	CX	70	10	INN
-267.8	1	647	240	82	32.7	OX2	50	14	INN

crossover, a deme size of 100, and a deme count of 10; this was also the best INN structure in terms of base pair overlap.

In general, the INN thermodynamic model performed best in terms of true-positive matching base pairs when considering the minimum free energy of single runs. Again, larger deme sizes also performed significantly better than smaller ones.

Table 7.5 presents the best single structure in terms of correctly predicted base pairs from an experiment regardless of its ΔG value. Table column headings are identical to those in Table 7.4. Should multiple single runs reach an identical value for total correct base pairs, the values for that experiment are averaged.

Table 7.5: Single run with the highest number of correctly predicted base pairs of *Xenopus laevis*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 251 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-269.65	1	658	238	**94**	**37.5**	OX2	70	10	INNHB
-268.60	2	500.5	247.5	**94**	**37.5**	OX2	100	10	INNHB
-282.18	1	667	235	87	34.7	OX2	50	10	INNHB
-268.96	1	690	239	86	34.3	CX	100	10	INNHB
-255.18	1	681	233	78	31.1	CX	50	14	INNHB
-295.98	1	595	248	70	27.9	CX	70	10	INNHB
-290.7	1	670	258	**100**	**39.8**	OX2	100	10	INN
-272.7	1	683	250	94	37.5	CX	100	10	INN
-267.8	1	647	240	82	32.7	OX2	50	14	INN
-262.8	1	691	246	82	32.7	OX2	70	10	INN
-273.4	2	514.5	233.0	80	31.9	CX	50	14	INN
-270.0	1	662	241	80	31.9	CX	70	10	INN

The best overall structure as indicated by Table 7.5 was found in a single INN run, with 39.8% correct matching base pairs. This single run reached a ΔG of -290.7 kcal/mol, and had 258 total base pairs. The run's parameters were an OX2 crossover, a deme size of 100, and a deme count of 10. By comparison, two separate INN-HB experiments reached 37.5% matching base pairs. The first experiment run's structure had a ΔG of -269.65 kcal/mol and 238 total base pairs. Its parameters were an OX2 crossover, a deme size of 70, and a

deme count of 10.

For the second INN-HB experiment, two separate runs reached an identical matching base pair percentage of 37.5%. These two runs had an average ΔG of -268.60 kcal/mol, and averaged 247.5 total base pairs. The experiment parameters for these two runs were an OX2 crossover, a deme size of 100, and a deme count of 10.

When considering the best structure in terms of matching known base pairs regardless of minimum free energy, the INN thermodynamic model performed best when reviewing single runs. Here also, larger deme sizes also performed significantly better than smaller ones.

The results in Table 7.3 show that for the *Xenopus laevis* there was a consistent correlation between a lower ΔG and a higher matching base pair count. Also noteworthy is that the higher deme sizes produce consistently better results in terms of lowest ΔG. Finally, *P-RnaPredict* succeeded in predicting up to 39.8% of the known base pairs in the best instance. This suggests that *P-RnaPredict* is performing effectively in exploring the structure space given the constraints of its helix generation algorithm and thermodynamic models. One possible explanation for *P-RnaPredict* not determining a greater part of the *Xenopus laevis* structure is that the known structure contains 22 non-canonical base pairs which cannot be predicted by *P-RnaPredict*.

7.3 *Drosophila virilis* - 784 nt

Table 7.6: *Drosophila virilis* details, taken from the Comparative RNA Web Site [4]

Filename	d.16.m.D.virilis.bpseq
Organism	*Drosophila virilis*
Accession Number	X05914
Class	16S rRNA
Length	784 nucleotides
# of BPs in known structure	233
# of non-canonical base pairs	11

Table 7.7 indicates that for both INN and INN-HB the lowest ΔG structures were reached with the following parameter set: an OX2 crossover, a deme size of 100 and a deme count of 10. The best INN-HB experiment found structures with an average ΔG of -181.14

Table 7.7: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Drosophila virilis* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 233 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-181.14	239.4	**29.6**	**12.7**	OX2	100	10	INNHB
-175.35	239.2	28.6	12.3	OX2	70	10	INNHB
-172.02	236.3	28.1	12.1	OX2	50	14	INNHB
-171.67	236.1	26.9	11.5	CX	100	10	INNHB
-169.35	235.6	29.0	12.4	CX	70	10	INNHB
-163.99	233.0	27.1	11.6	CX	50	14	INNHB
-159.2	238.1	40.6	17.4	OX2	100	10	INN
-155.7	237.3	**41.3**	**17.7**	CX	100	10	INN
-154.1	238.7	34.8	14.9	OX2	70	10	INN
-153.2	238.6	35.3	15.2	OX2	50	14	INN
-147.1	234.0	33.0	14.1	CX	70	10	INN
-143.1	232.2	29.7	12.7	CX	50	14	INN

kcal/mol, 239.4 total base pairs, and matching 12.7% of base pairs in the known structure.

The best INN experiment found structures with an average ΔG of -159.2 kcal/mol, 238.1 total base pairs, and matching 17.4% of base pairs in the known structure. An INN experiment employing the CX crossover performed slightly better in terms of correct base pairs, finding 17.7% matching base pairs in the known structure, 237.3 total base pairs, but a slightly higher ΔG of -155.7 kcal/mol.

In general INN performed dramatically better than INN-HB, in terms of true-positive matching base pairs, when considering the averaged results of the minimum free energy experiment. Here again larger deme sizes, such as 100, performed significantly better than smaller ones.

Table 7.8 shows that for INN-HB, the best experiment reached a ΔG of -200.67 kcal/mol in a single run after 623 generations. This structure contained 252 base pairs, matched 21.3% of known base pairs, and was the best structure found by INN-HB. For INN, the best experiment reached a structure with a ΔG of -174.5 kcal/mol in a single run after 521 generations; it contained 248 base pairs. However, the best overall structure found with INN matched 22.7% of known base pairs and had a ΔG of -166.9 kcal/mol. INN performed best overall in terms of matching known base pairs when considering the minimum free

Table 7.8: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Drosophila virilis* structure. Results are grouped by thermodynamic model. The known structure contains 233 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-200.67	1	623	252	**49**	**21.3**	OX2	100	10	INNHB
-191.97	1	476	250	41	17.6	OX2	70	10	INNHB
-187.46	1	397	246	32	13.7	CX	100	10	INNHB
-186.02	1	486	237	30	12.9	CX	70	10	INNHB
-185.51	1	178	241	31	13.3	OX2	50	14	INNHB
-182.27	1	420	240	27	11.6	CX	50	14	INNHB
-174.5	1	521	248	21	9.0	OX2	100	10	INN
-168.2	1	436	242	45	19.3	OX2	50	14	INN
-166.9	1	447	245	**53**	**22.7**	OX2	70	10	INN
-166.0	1	429	239	41	17.6	CX	100	10	INN
-160.9	1	646	245	37	15.9	CX	50	14	INN
-160.3	1	602	248	39	16.7	CX	70	10	INN

energy of single runs; however, here the best structure does not coincide with the minimum free energy. Here again runs with larger deme sizes appear to result in significantly better results, both in terms of minimum free energy and matching known base pairs, than their smaller-sized counterparts.

Table 7.9: Single run with the highest number of correctly predicted base pairs of *Drosophila virilis*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 233 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-172.75	1	530	245	**63**	**27.0**	OX2	70	10	INNHB
-175.65	1	588	238	57	24.5	CX	100	10	INNHB
-163.06	1	558	226	55	23.6	CX	50	14	INNHB
-168.37	1	461	225	50	21.5	OX2	50	14	INNHB
-162.96	1	404	237	49	21.0	CX	70	10	INNHB
-200.67	1	623	252	49	21.0	OX2	100	10	INNHB
-152.5	1	554	234	**66**	**28.3**	CX	100	10	INN
-170.9	2	406.5	239.5	65	27.9	OX2	100	10	INN
-158.2	1	501	243	56	24.0	OX2	50	14	INN
-156.1	1	523	223	54	23.2	CX	70	10	INN
-166.9	2	382.0	243.0	53	22.7	OX2	70	10	INN
-141.5	1	698	231	49	21.0	CX	50	14	INN

The best overall structure as indicated by Table 7.9 was found in a single INN run, with 28.3% correct matching base pairs. This single run reached a ΔG of -152.5 kcal/mol, and had 234 total base pairs. By comparison, the best INN-HB run reached 27.0% correct matching base pairs, with a ΔG of -172.75 kcal/mol, and a total of 245 base pairs.

Here as with the previous sequence, the INN thermodynamic model produced the overall best structure in terms of matching known base pairs. Also, larger deme sizes again produced significantly superior results in terms of matching base pairs and minimum free energy.

In summary, for the *Drosophila virilis* structure, *P-RnaPredict* was able to predict 21.3% of the known structure with its single lowest free energy run. When considering the best matching structure independent of minimum free energy, 28.3% of the known structure was successfully predicted. Notably, there are 11 non-canonical base pairs in the *Drosophila*

virilis structure; these cannot be predicted by *P-RnaPredict* and contribute to the difficulties in determining its structure.

7.4 *Hildenbrandia rubra* - 543 nt

Table 7.10: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Hildenbrandia rubra* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 138 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-217.03	161.4	**43.7**	**31.7**	OX2	100	10	INNHB
-211.09	158.0	39.3	28.5	OX2	50	14	INNHB
-209.86	159.2	36.4	26.4	OX2	70	10	INNHB
-206.74	157.2	37.0	26.8	CX	100	10	INNHB
-204.17	155.7	36.3	26.3	CX	70	10	INNHB
-199.23	154.7	31.6	22.9	CX	50	14	INNHB
-198.2	161.7	37.9	27.5	OX2	100	10	INN
-193.6	159.0	38.0	27.5	OX2	70	10	INN
-190.4	158.3	32.8	23.8	OX2	50	14	INN
-188.3	157.6	**39.6**	**28.7**	CX	100	10	INN
-187.0	156.0	32.6	23.6	CX	50	14	INN
-186.6	157.2	37.5	27.2	CX	70	10	INN

Table 7.10 indicates that for both INN and INN-HB the lowest ΔG structures were reached with the following parameter set: an OX2 crossover, a deme size of 100 and a deme count of 10. The best INN-HB experiment found structures with an average ΔG of -217.03 kcal/mol, 161.4 total base pairs, and matching 31.7% of base pairs in the known structure.

The best INN experiment found structures with an average ΔG of -198.2 kcal/mol, 161.7 total base pairs, and matching 27.5% of base pairs in the known structure. However, an INN experiment employing the CX crossover performed slightly better, finding 28.7% matching base pairs in the known structure and 157.6 total base pairs, but a slightly higher ΔG of -188.3 kcal/mol.

For *Hildenbrandia rubra*, INN performed best in terms of true-positive matching base pairs when considering the averaged results of the minimum free energy experiment. However, neither thermodynamic model's best structure coincided with the lowest free energy.

Here as well, the experiments with larger deme sizes produced lower free energies and greater known structure agreement than smaller ones.

Table 7.11: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Hildenbrandia rubra* structure. Results are grouped by thermodynamic model. The known structure contains 138 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-225.43	1	417	166	53	38.4	CX	100	10	INNHB
-224.97	1	369	162	43	31.2	OX2	100	10	INNHB
-224.53	1	513	171	41	29.7	OX2	50	14	INNHB
-224.19	1	524	158	**64**	**46.4**	CX	70	10	INNHB
-220.36	1	418	154	53	38.4	OX2	70	10	INNHB
-212.77	1	532	164	33	23.9	CX	50	14	INNHB
-211.9	1	350	156	49	35.5	OX2	100	10	INN
-210.9	1	384	158	49	35.5	OX2	70	10	INN
-206.6	1	141	165	**55**	**39.9**	CX	70	10	INN
-203.4	1	241	164	39	28.3	CX	100	10	INN
-200.4	1	630	165	39	28.3	CX	50	14	INN
-199.4	1	628	161	26	18.8	OX2	50	14	INN

Table 7.11 indicates that for INN-HB the best experiment reached a ΔG of -225.43 kcal/mol in a single run after 417 generations. This structure contained 166 base pairs and matched 38.4% of known base pairs. However, the best overall INN-HB structure matched 46.4% of known base pairs and had a ΔG of -224.19 kcal/mol.

For INN, the best experiment reached a structure with a ΔG of -211.9 kcal/mol in a single run after 350 generations; it contained 156 base pairs. However, the best overall structure found with INN matched 39.9% known base pairs and had a ΔG of -206.6 kcal/mol.

In general, INN-HB performed best in terms of true-positive matching base pairs when considering the minimum free energy of single runs. Nevertheless, neither thermodynamic model's best structure coincided with the lowest free energy. As with previous experiments, larger deme sizes resulted in lower free energies and greater known structure agreement.

The best overall structure as indicated by Table 7.12 was found in a single INN-HB run, with 51.4% correct matching base pairs. This single run reached a ΔG of -215.72 kcal/mol, and had 159 total base pairs. By comparison, the best INN run reached 50.0% matching

Table 7.12: Single run with the highest number of correctly predicted base pairs of *Hildenbrandia rubra*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 138 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-215.72	1	668	159	**71**	**51.4**	OX2	100	10	INNHB
-211.46	1	533	166	67	48.6	OX2	50	14	INNHB
-204.86	1	229	159	66	47.8	CX	70	10	INNHB
-225.15	1	143	166	66	47.8	CX	100	10	INNHB
-214.18	1	538	164	59	42.8	OX2	70	10	INNHB
-212.04	1	679	160	53	38.4	CX	50	14	INNHB
-203.3	1	268	168	**69**	**50.0**	CX	100	10	INN
-201.5	1	292	166	68	49.3	OX2	100	10	INN
-199.1	1	433	164	64	46.4	CX	70	10	INN
-196.3	1	314	159	59	42.8	OX2	70	10	INN
-199.9	1	618	156	53	38.4	CX	50	14	INN
-197.8	1	476	161	48	34.8	OX2	50	14	INN

base pairs, with a ΔG of -203.3 kcal/mol, and 168 total base pairs.

When considering the highest number of known base pairs independent of minimum free energy, INN-HB performed best overall. Here again, larger deme sizes produce significantly better results in terms of minimum free energy and greater known structure agreement.

In summary, *P-RnaPredict* was able to predict 38.4% of the known structure with its single lowest free energy run. When considering the best matching structure independent of minimum free energy, 51.4% of the known structure was successfully predicted. There is only 1 non-canonical base pair in the known *Hildenbrandia rubra* structure; this makes structure prediction more straightforward and may partially account for the increased accuracy, along with the shorter sequence length.

7.5 *Haloarcula marismortui* - 122 nt

Table 7.13: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Haloarcula marismortui* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 38 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-54.94	33	16	42.1	OX2	50	14	INNHB
-54.94	33	16	42.1	CX	100	10	INNHB
-54.94	33	16	42.1	OX2	70	10	INNHB
-54.94	33	16	42.1	OX2	100	10	INNHB
-54.94	33	16	42.1	CX	70	10	INNHB
-54.92	33.3	14.3	37.7	CX	50	14	INNHB
-52.8	30	16	42.1	OX2	100	10	INN
-52.8	30	16	42.1	CX	50	14	INN
-52.8	30	16	42.1	OX2	70	10	INN
-52.8	30	16	42.1	OX2	50	14	INN
-52.8	30	16	42.1	CX	100	10	INN
-52.8	30.1	15.9	41.8	CX	70	10	INN

Table 7.13 indicates that nearly all experiments with this sequence converged to identical average ΔG values, depending on the thermodynamic model selected. With INN-HB, the ΔG reached was -54.94 kcal/mol, with a total of 33 base pairs predicted; it matched 42.1% of known base pairs. The one exception in INN-HB is an experiment which reached a ΔG

of -54.92 kcal/mol, with 33.3 total base pairs predicted; matching only 37.7% of known base pairs. Its parameters were a CX crossover, a deme size of 50, and a deme count of 14.

The INN experiments reached a ΔG of -52.8 kcal/mol, with 30 base pairs predicted, and 42.1% of known base pairs matched. Here, the one exception is an experiment with the same ΔG of -52.8 kcal/mol, but with 30.1 total base pairs predicted, and just 41.8% of known base pairs matched.

Table 7.14: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Haloarcula marismortui* structure. Results are grouped by thermodynamic model. The known structure contains 38 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-54.94	30	17.3	33.0	16.0	42.1	CX	100	10	INNHB
-54.94	30	18.4	33.0	16.0	42.1	OX2	100	10	INNHB
-54.94	30	40.5	33.0	16.0	42.1	OX2	70	10	INNHB
-54.94	30	40.5	33.0	16.0	42.1	OX2	50	14	INNHB
-54.94	30	49.7	33.0	16.0	42.1	CX	70	10	INNHB
-54.94	25	59.7	33.0	16.0	42.1	CX	50	14	INNHB
-52.8	30	24.0	30.0	16.0	42.1	OX2	100	10	INN
-52.8	29	25.7	30.0	16.0	42.1	CX	70	10	INN
-52.8	30	26.2	30.0	16.0	42.1	OX2	50	14	INN
-52.8	30	29.8	30.0	16.0	42.1	CX	100	10	INN
-52.8	30	30.6	30.0	16.0	42.1	OX2	70	10	INN
-52.8	30	51.1	30.0	16.0	42.1	CX	50	14	INN

Table 7.14 indicates that in each experiment nearly all of the 30 randomly-seeded runs converged to identical structures depending on thermodynamic model. Thus, the Gens. column values are decimals because they are averaged over those 30 runs.

With INN-HB the ΔG reached was -54.94 kcal/mol, with 33.0 total base pairs predicted, and matching 42.1% of known base pairs. The one INN-HB exception is an experiment which only reached the final structure in 25 out of its 30 runs; its parameters were a CX crossover, a deme size of 50, and a deme count of 14.

Similar behaviour occurred with the INN model, which reached a ΔG of -52.8 kcal/mol, with 30.0 base pairs predicted, and matching 42.1% of known base pairs. The one INN-HB

exception is an experiment which reached the final structure in 29 out of its 30 runs; its parameters were a CX crossover, a deme size of 70, and a deme count of 10.

Although all experiments converged to structures with identical percentages of known base pairs, the INN model performed best, predicting significantly fewer false-positive base pairs than INN-HB. If we sort the experiment results by the Gens. column in increasing order, it can be seen that on average the larger deme sizes result in faster convergence. Also, the two experiments which did not have all 30 runs converge to the same structure occurred in experiments with smaller deme sizes.

Table 7.15: Single run with the highest number of correctly predicted base pairs of *Haloarcula marismortui*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 38 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-54.94	30	17.3	33.0	16	42.1	CX	100	10	INNHB
-54.94	30	18.4	33.0	16	42.1	OX2	100	10	INNHB
-54.94	30	40.5	33.0	16	42.1	OX2	70	10	INNHB
-54.94	30	40.5	33.0	16	42.1	OX2	50	14	INNHB
-54.94	30	49.7	33.0	16	42.1	CX	70	10	INNHB
-54.94	25	59.7	33.0	16	42.1	CX	50	14	INNHB
-52.8	30	24.0	30.0	16	42.1	OX2	100	10	INN
-52.8	29	25.7	30.0	16	42.1	CX	70	10	INN
-52.8	30	26.2	30.0	16	42.1	OX2	50	14	INN
-52.8	30	29.8	30.0	16	42.1	CX	100	10	INN
-52.8	30	30.6	30.0	16	42.1	OX2	70	10	INN
-52.8	30	51.1	30.0	16	42.1	CX	50	14	INN

Noting that the results from Table 7.14 and Table 7.15 are in fact identical, it appears that *P-RnaPredict* has reached the best possible structures it could find given the current thermodynamic models.

Table 7.16: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Saccharomyces cerevisiae* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 37 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-57.52	39	33	89.2	OX2	50	14	INNHB
-57.52	39	33	89.2	CX	100	10	INNHB
-57.52	39	33	89.2	OX2	70	10	INNHB
-57.52	39	33	89.2	OX2	100	10	INNHB
-57.52	39	33	89.2	CX	70	10	INNHB
-57.52	39	33	89.2	CX	50	14	INNHB
-52.9	40	28	75.7	OX2	100	10	INN
-52.9	40	28	75.7	CX	50	14	INN
-52.9	40	28	75.7	OX2	70	10	INN
-52.9	40	28	75.7	CX	70	10	INN
-52.9	40	28	75.7	OX2	50	14	INN
-52.9	40	28	75.7	CX	100	10	INN

7.6 *Saccharomyces cerevisiae* - 118 nt

Table 7.16 indicates that all experiments with this sequence converged to identical average ΔG values, depending on the thermodynamic model selected. With INN-HB, the ΔG reached was -57.52 kcal/mol, with 39 base pairs predicted; matching 89.2% of known base pairs. The INN experiments reached a ΔG of -52.9 kcal/mol, with 40 base pairs predicted, and 75.7% of known base pairs matched.

With this particular sequence, Table 7.17 indicates that every one of the 30 randomly-seeded runs converged to identical structures depending on thermodynamic model. Thus, the Gens. column values are decimals because they are averaged over those 30 runs. Also, the minimum ΔG value was reached by all runs within an average of 10 generations. As above, with INN-HB the ΔG reached was -57.52 kcal/mol, with 39.0 base pairs predicted, and matching 89.2% of known base pairs. Again, with INN the ΔG reached was -52.9 kcal/mol, with 39.0 base pairs predicted, and matching 75.7% of known base pairs.

One final point of interest is that sorting the experiment results by the Gens. column in increasing order indicates that on average the larger deme sizes result in faster convergence. However, all values are within a few generations of each other.

Table 7.17: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Saccharomyces cerevisiae* structure. Results are grouped by thermodynamic model. The known structure contains 37 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-57.52	30	5.96	39.0	33.0	89.2	CX	100	10	INNHB
-57.52	30	6.43	39.0	33.0	89.2	OX2	100	10	INNHB
-57.52	30	7.06	39.0	33.0	89.2	OX2	70	10	INNHB
-57.52	30	7.80	39.0	33.0	89.2	CX	70	10	INNHB
-57.52	30	7.96	39.0	33.0	89.2	OX2	50	14	INNHB
-57.52	30	9.53	39.0	33.0	89.2	CX	50	14	INNHB
-52.9	30	5.90	40.0	28.0	75.7	CX	100	10	INN
-52.9	30	6.30	40.0	28.0	75.7	OX2	100	10	INN
-52.9	30	7.36	40.0	28.0	75.7	CX	70	10	INN
-52.9	30	8.10	40.0	28.0	75.7	OX2	70	10	INN
-52.9	30	8.50	40.0	28.0	75.7	OX2	50	14	INN
-52.9	30	9.20	40.0	28.0	75.7	CX	50	14	INN

Table 7.18: Single run with the highest number of correctly predicted base pairs of *Saccharomyces cerevisiae*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 37 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-57.52	30	5.96	39.0	33	89.2	CX	100	10	INNHB
-57.52	30	6.43	39.0	33	89.2	OX2	100	10	INNHB
-57.52	30	7.06	39.0	33	89.2	OX2	70	10	INNHB
-57.52	30	7.80	39.0	33	89.2	CX	70	10	INNHB
-57.52	30	7.96	39.0	33	89.2	OX2	50	14	INNHB
-57.52	30	9.53	39.0	33	89.2	CX	50	14	INNHB
-52.9	30	5.90	40.0	28	75.7	CX	100	10	INN
-52.9	30	6.30	40.0	28	75.7	OX2	100	10	INN
-52.9	30	7.36	40.0	28	75.7	CX	70	10	INN
-52.9	30	8.10	40.0	28	75.7	OX2	70	10	INN
-52.9	30	8.50	40.0	28	75.7	OX2	50	14	INN
-52.9	30	9.20	40.0	28	75.7	CX	50	14	INN

For the *Saccharomyces cerevisiae* structure, all *P-RnaPredict* experiments converged to structures with identical percentages of known base pairs, depending on thermodynamic model. Here the INN-HB model clearly performed best predicting a greater percentage of known base pairs and significantly fewer false-positive base pairs than INN.

Noting that the results from Table 7.18 and Table 7.17 are in fact identical, it is clear that *P-RnaPredict* has reached the best possible structures it could find given the current thermodynamic models.

7.6.1 Secondary Structure Comparison

Figure 7.2 shows the known secondary structure for the *Saccharomyces cerevisiae* RNA sequence. There are a total of 37 base pairs.

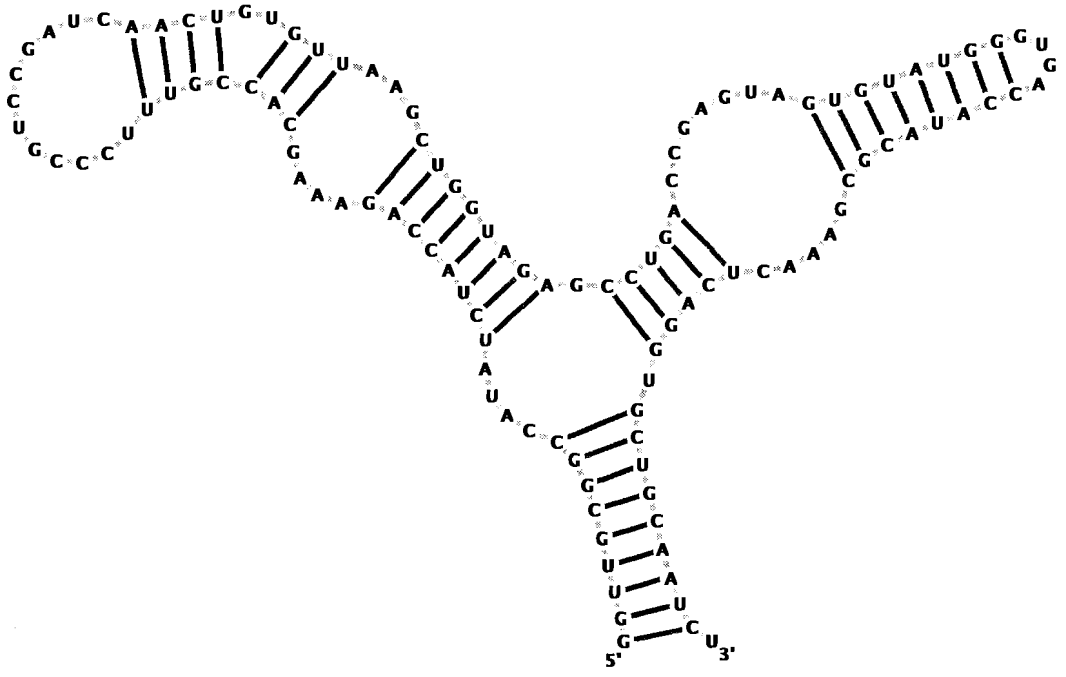

Figure 7.2: This plot shows the known secondary structure for the *Saccharomyces cerevisiae* RNA sequence. Black lines indicate base pairs in the known structure.

For the *S. cerevisiae* sequence, the highest number of correctly predicted base pairs *P-RnaPredict* found was 33 out of 37, or 89.2%. Figure 7.3 shows a comparison between the

known secondary structure for the *Saccharomyces cerevisiae* sequence, and the secondary structure predicted by *P-RnaPredict*. Light grey bonds indicate base pairs in the known structure not predicted by the GA. Dark grey bonds indicate base pairs predicted by the GA but not present in the known structure. Black bonds indicate base pairs present both in the known and predicted structure.

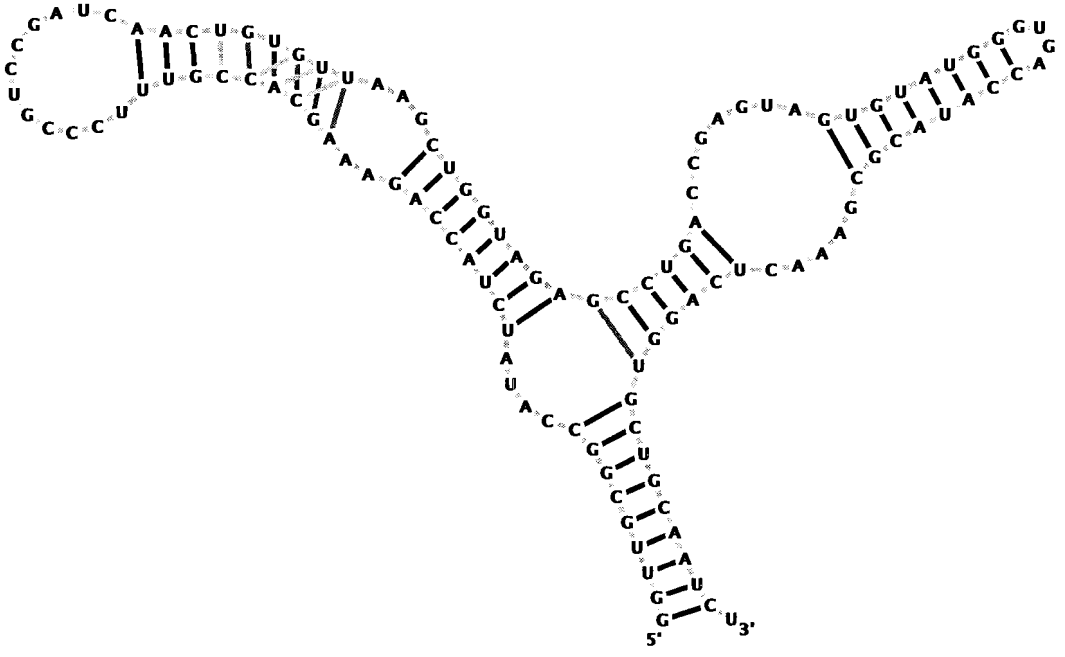

Figure 7.3: This plot shows a comparison between the known and predicted secondary structures for the *Saccharomyces cerevisiae* RNA sequence. Dark grey lines indicate predicted base pairs, light grey lines indicate base pairs in the known structure, and the black lines indicate the overlap between predicted and known base pairs. In this case, *P-RnaPredict* was able to predict 89.2% of the known base pairs.

It is interesting to note that *P-RnaPredict*'s current thermodynamic models do not account for non-canonical base pairs. However, they do exist in naturally occurring structures including *S. cerevisiae*. Note the two CU pairs in the structure in Figure 7.3, which could not have been predicted with the current thermodynamic models. This is why *P-RnaPredict* has predicted a different helix than what occurs naturally; this can be seen in the slight shift of the helix on the left of the figure. Within the limits of the underlying model, *P-RnaPredict* has found all correct base pairs it could possibly find. *H.marismortui* has a similar length to

S.cerevisiae; however, it contains many more non-canonical base pairs. This explains why the prediction accuracy is much lower for *H.marismortui* than for *S.cerevisiae*.

7.7 Summary

The results reviewed in this chapter indicate that *P-RnaPredict* demonstrates a noteworthy competency in determining low ΔG structures in a reasonable number of iterations. There is a clear correlation between a lower ΔG and the percentage of matching base pairs in the known structure.

In terms of overall results from these sequences, larger deme sizes had a significant impact on the results. In all cases, runs with a deme size of 100 and deme count of 10 either produced the best results overall or tied for best result. This suggests that *P-RnaPredict* can employ a larger overall population size to improve the quality of results while still retaining the benefits of parallel speedup.

P-RnaPredict successfully improved upon the initial randomly generated population, converging the population to a lower ΔG and increasing the percentage of matching known base pairs. However, non-canonical base pairs in naturally occurring structures cannot be modeled with current thermodynamic models. With the *Saccharomyces cerevisiae* sequence, *P-RnaPredict* successfully predicted 89.2% of the known base pairs. Overall, the prediction accuracy of *P-RnaPredict* is good, particularly so for shorter sequences.

Chapter 8

Comparison to Nussinov DPA

As mentioned in Section 1.3.3, the first application of DP to structure prediction was developed by Nussinov [66], and functioned by maximizing the number of base pairs in a predicted structure. The purpose behind running a comparison between *P-RnaPredict* and the Nussinov DPA is to provide a baseline by which the performance of *P-RnaPredict* may be judged. Specifically, this is to validate the correctness of the minimum free energy approach and thermodynamic models employed by *P-RnaPredict* against the pure base pair maximization approach utilized by the Nussinov DPA.

Originally, the Nussinov DPA applied equal scores of 1:1:1 to each GC:AU:GU base pair bond. In these experiments a basic implementation developed by Vingron [92] was modified to factor in the free energy of each type of base pair. Two variations of base pair scoring were included. The first set of scores were based on the relative free energy of the base pairs, and were designed to emulate the Major thermodynamic model outlined in Section 2.3.1. The selected free energy values were -3, -2, and -1 kcal/mol at 37°C for GC, AU, and GU base pairs respectively. Thus, the default Nussinov DPA GC:AU:GU scores of 1:1:1 are replaced by 3:2:1.

The second set of scores are 3:2:2, and are based on the relative number of hydrogen bonds in GC:AU:GU per base pair as per the Mathews model discussed in Section 2.3.2. The corresponding free energy values were -3, -2, and -2 kcal/mol at 37°C for GC, AU, and GU base pairs respectively. It is important to note that none of these base pair scores account for the destabilizing effect of loops or the stabilizing effect of stacked pairs. Thus, there are no constraints on the upper bounds of loop and helix length.

This chapter presents a comparison of the structures predicted by the Nussinov DPA

using these three weightings with those predicted by *P-RnaPredict*.

8.1 *Xenopus laevis* - 945 nt

The results from applying the Nussinov DPA to the *Xenopus laevis* sequence are shown in Table 8.1. Of the three weightings, 1:1:1 will always produce the maximum number of possible base pairs; in this case, it was 341. The 1:1:1 structure contained 12.0% of the known base pairs. A GC:AU:GU weighting of 3:2:2 produced a structure with 339 base pairs and 15.6% matching base pairs in the known structure. Finally, a weighting of 3:2:1 produced 333 base pairs, and correctly predicted 18.7% of the known structure.

Table 8.1: *Xenopus laevis*, Nussinov results. Number of known base pairs is 251.

DPA Weights	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
1:1:1	341	30	12.0
3:2:2	339	39	15.6
3:2:1	333	47	18.7

Table 8.2: Comparison of *P-RnaPredict* and Nussinov DPA on *Xenopus laevis* sequence.

Source	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
Nussinov Best BP	333	47	18.7
GA Average ΔG	241.0	65.7	26.2
GA Best ΔG	258	100	39.8
GA Best BP Overlap	258	100	39.8

Table 8.2 provides a direct comparison between the best Nussinov structure from Table 8.1 and the best *P-RnaPredict* structures from Tables 7.3, 7.4, and 7.5. "Source" indicates the origin of the structures, and includes the following four sources: "Nussinov Best BP" is the Nussinov structure with the highest percentage of correctly matching base pairs. "GA Average ΔG" is the averaged result from the lowest ΔG *P-RnaPredict* experiment. "GA Best ΔG" is the *P-RnaPredict* structure from the single GA run with the lowest ΔG.

"GA Best BP Overlap" is the *P-RnaPredict* structure from the single GA run with the highest percentage of correctly matching base pairs, regardless of ΔG.

"Predicted BP" is the total number of predicted base pairs. "Correctly Predicted BP" is the total number of correctly predicted base pairs. Finally, "Correctly Predicted %" is the percentage of correctly predicted base pairs.

Considering matching known base pairs for this sequence, Table 8.2 indicates that *P-RnaPredict* is a dramatic improvement over the Nussinov algorithm. For example, the *P-RnaPredict* average ΔG result finds 26.2% of the known structure, compared to 18.7% for the best Nussinov structure. Notably, the best *P-RnaPredict* prediction contains 39.8% of the known structure. When reviewing false-positive base pairs, the best Nussinov DPA structure contains 333 total base pairs, far more than the 258 predicted by *P-RnaPredict.* Thus, all *P-RnaPredict* results are far superior to the best Nussinov structure for this sequence.

8.2 *Drosophila virilis* - 784 nt

Table 8.3 shows the results when using the Nussinov DPA. First, the maximum number of possible base pairs is 320, given a corresponding GC:AU:GU weighting of 1:1:1. The number of base pairs in the known structure is 233, and 12.4% are correctly predicted base pairs. A change in weights to 3:2:2 results in a structure with 319 base pairs, with 9.9% correctly predicted base pairs. Finally, a weight of 3:2:1 results in a structure with 309 base pairs, of which 9.0% are correctly predicted base pairs.

Table 8.3: *Drosophila virilis*, Nussinov results. Number of known base pairs is 233.

DPA Weights	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
1:1:1	320	29	12.4
3:2:2	319	23	9.9
3:2:1	309	21	9.0

The best results from Tables 7.7, 7.8, and 7.9 are combined with the best Nussinov structure from Table 8.3. The best Nussinov structure contains a total of 320 base pairs, and matches 12.4% of the known structure. By contrast, the *P-RnaPredict* average ΔG

Table 8.4: Comparison of *P-RnaPredict* and Nussinov DPA on *Drosophila virilis* sequence.

Source	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
Nussinov Best BP	320	29	12.4
GA Average ΔG	238.1	40.6	17.4
GA Best ΔG	252	49	21.3
GA Best BP Overlap	234	66	28.3

prediction contains 238.1 total base pairs, and matches 17.4% of the known structure. The best ΔG result from *P-RnaPredict* contains 252 base pairs, and correctly predicts 21.3% of the known structure. Finally, the best overall *P-RnaPredict* structure contains 234 base pairs, 28.3% of which match the known structure. Again, all three *P-RnaPredict* results are dramatically better than the Nussinov DPA both in terms of matching known base pairs and fewer false-positives.

8.3 *Hildenbrandia rubra* - 543 nt

The Nussinov results are shown in Table 8.5. By simply maximizing the number of possible base pairs, the generated structure contains 213 base pairs, while the known structure contains 138 base pairs. The generated structure correctly predicts 5.0% of the base pairs existing in the known structure. A GC:AU:GU weight modification to 3:2:2, results in 211 predicted base pairs, of which 22.5% of the known structure is found. Finally, a weight modification to 3:2:1 reduced the number of predicted base pairs to 205, but this structure still correctly predicts 22.5% of the base pairs existing in the known structure and has fewer false-positives.

Table 8.6 summarizes the best *P-RnaPredict* results from Tables 7.10, 7.11, and 7.12, and compares them to the best Nussinov structure from Table 8.5. The *P-RnaPredict* average ΔG result contains 161.4 total base pairs, and matches 31.7% of the known structure. In comparison, the best Nussinov structure contains 205 total base pairs, and only matches 22.5% of the known structure. The lowest ΔG and best matching base pair *P-RnaPredict* structures are also significantly better at 38.4% and 51.4% matching base pairs, respectively. Again, in terms of a higher percentage of matching base pairs and less false-positive base pairs, *P-RnaPredict* has considerably outperformed the Nussinov DPA.

Table 8.5: *Hildenbrandia rubra* Nussinov results. Number of known base pairs is 138.

DPA Weights	**Predicted BP**	**Correctly Predicted BP**	**Correctly Predicted (%)**
1:1:1	213	7	5.0
3:2:2	211	31	22.5
3:2:1	205	31	22.5

Table 8.6: Comparison of *P-RnaPredict* and Nussinov DPA on *Hildenbrandia rubra* sequence.

Source	**Predicted BP**	**Correctly Predicted BP**	**Correctly Predicted (%)**
Nussinov Best BP	205	31	22.5
GA Average ΔG	161.4	43.7	31.7
GA Best ΔG	166	53	38.4
GA Best BP Overlap	159	71	51.4

8.4 *Haloarcula marismortui* - 122 nt

Table 8.7 shows the Nussinov DPA *Haloarcula marismortui* results. Maximizing the number of base pairs with a 1:1:1 weighting results in 45 base pairs, matching 21.1% of the known base pairs. Changing the GC:AU:GU weights to 3:2:2 produced a structure containing 44 base pairs with 10.5% correctly predicted base pairs. Finally, a 3:2:1 weighting produced identical results to the 3:2:2 weighting.

Table 8.7: *Haloarcula marismortui*, Nussinov results. Number of known base pairs is 38.

DPA Weights	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
1:1:1	45	8	21.1
3:2:2	44	4	10.5
3:2:1	44	4	10.5

Table 8.8: Comparison of *P-RnaPredict* and Nussinov DPA on *Haloarcula marismortui* sequence.

Source	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
Nussinov Best BP	45	8	21.1
GA Average ΔG	33.0	16.0	42.1
GA Best ΔG	33	16	42.1
GA Best BP Overlap	33	16	42.1

Table 8.8 reviews the best *P-RnaPredict* results from Tables 7.13, 7.14, and 7.15, and the best Nussinov structure from Table 8.7. Considering matching known base pairs for this sequence, Table 8.8 indicates that *P-RnaPredict* is a considerable improvement over the Nussinov algorithm. All *P-RnaPredict* results converged to identical structures with 33 total base pairs, 42.1% of which were correctly predicted. By contrast, the best Nussinov structure contains 45 base pairs, and matches 21.1% of known base pairs. Again, all *P-RnaPredict* results are a significant improvement over the best Nussinov structure for this sequence.

8.5 *Saccharomyces cerevisiae* - 118 nt

Results from the Nussinov DPA are shown in Table 8.9. Using a weighting of 1:1:1, the maximal number of base pairs predicted is 45, with 75.7% correctly predicted. With the weighting changed to 3:2:2, the algorithm predicts a different structure also containing 45 base pairs, resulting in 75.7% correctly predicted base pairs in the known structure. Finally, using a 3:2:1 weighting, the algorithm predicts 44 total base pairs, of which 24.3% are base pairs in the known structure.

Table 8.9: *Saccharomyces cerevisiae* Nussinov results. Number of known base pairs is 37.

DPA Weights	**Predicted BP**	**Correctly Predicted BP**	**Correctly Predicted (%)**
1:1:1	45	28	75.7
3:2:2	45	28	75.7
3:2:1	44	9	24.3

Table 8.10: Comparison of *P-RnaPredict* and Nussinov DPA on *Saccharomyces cerevisiae* sequence.

Source	**Predicted BP**	**Correctly Predicted BP**	**Correctly Predicted (%)**
Nussinov Best BP	35	28	75.7
GA Average ΔG	39.0	33.0	89.2
GA Best ΔG	39	33	89.2
GA Best BP Overlap	39	33	89.2

The best Nussinov structure from Table 8.9 and the best *P-RnaPredict* results from Tables 7.16, 7.17, and 7.18 are reviewed in Table 8.10. As with *Haloarcula marismortui*, all *P-RnaPredict* results converged to identical structures with 39 total base pairs, matching 89.2% of the known structure. By comparison, the best Nussinov structure contained 35, and matched 75.7% of the known structure. Here also, *P-RnaPredict* has considerably outperformed the Nussinov DPA in terms of a higher percentage of matching base pairs and less false-positive base pairs.

8.5.1 Secondary Structure Comparison

Figure 8.1 shows a comparison between the known secondary structure for the *Saccharomyces cerevisiae* sequence, and the best secondary structure predicted by the Nussinov DPA. Light grey bonds indicate predicted base pairs which are not present in the known structure. Black bonds indicate base pairs present both in the known and predicted structure. Bonds present only in the known structure are omitted for clarity.

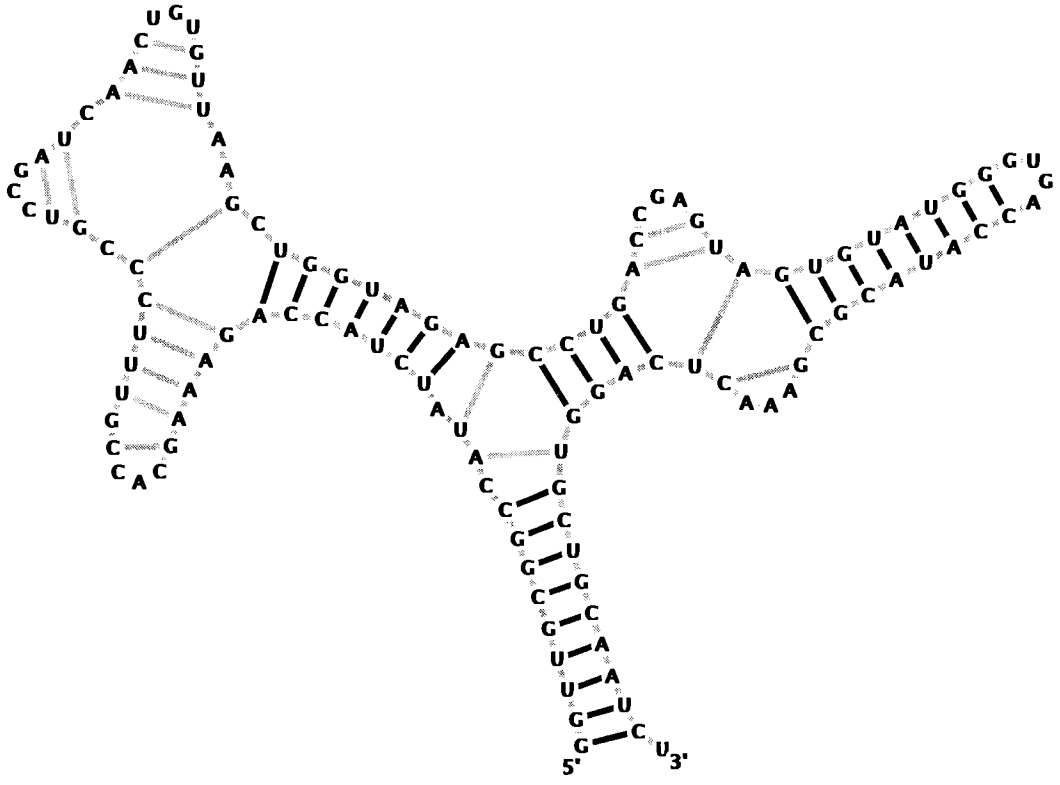

Figure 8.1: This plot shows a comparison between the known and predicted secondary structures for the *Saccharomyces cerevisiae* RNA sequence. Light grey lines indicate predicted base pairs, and black lines indicate the overlap between predicted and known base pairs. In this case, the best Nussinov DPA result was able to predict 75.7% of the known base pairs.

Although the percentage of correct base pairs, 75.7%, appears quite high in the Nussinov structure, there are striking differences between it and the known structure. The presence of two isolated base pairs dramatically skews the internal loops in the structure. Also, the

Nussinov algorithm predicts three hairpin loops on the left of the structure where only one should exist. Reviewing the known structure in Figure 7.2 and the structure predicted by *P-RnaPredict* in Figure 7.3, it can be seen that *P-RnaPredict* is able to predict a more accurate structure than the Nussinov DPA.

8.6 False-Positives: Over-prediction of base pairs

Table 8.11 presents a comparison of the total false-positive base pairs predicted by *P-RnaPredict* and the Nussinov DPA. Each *P-RnaPredict* entry was taken from the best experiment averaged over its 30 runs, and the Nussinov entries were taken from the best overall Nussinov run. A breakdown of the table column names follows.

"Sequence" is the name of the organism the sequence was taken from. "DPA Weights" are the base pair weights (GC:AU:GU) assigned to the Nussinov algorithm. "DPA over-pred." is the number of false-positive base pairs predicted by the Nussinov DPA. "GA over-pred." is the number of false-positive base pairs predicted by *P-RnaPredict*. "DPA Corr. BPs" is the number of true-positive base pairs predicted by the Nussinov DPA. "GA Corr. BPs" is the number of true-positive base pairs predicted by *P-RnaPredict*. Finally, "Cross.-Deme Size-Deme Count-Model" describes the parameters for the specific *P-RnaPredict* experiment, separated by dashes; these are the crossover, deme size, deme count, and thermodynamic model. An entry with "ALL" indicates that all settings for that parameter produced identical results.

For all sequences tested, *P-RnaPredict* correctly predicted more true-positive base pairs and less false-positives than the Nussinov DPA. The results demonstrate that with the sequences tested, *P-RnaPredict* performs better at predicting RNA secondary structures than the Nussinov DPA.

The results in Table 8.11 reviewed the average performance of *P-RnaPredict*. Next, we will discuss the best single runs of *P-RnaPredict*. A comparison between the structures determined by the individual *P-RnaPredict* experiment runs with the lowest free energy and those predicted by the best overall Nussinov run is shown in Table 8.12. In all cases, the results for these sequences indicate the superior performance of *P-RnaPredict* to Nussinov.

Table 8.13 presents a comparison between the structures predicted by *P-RnaPredict* containing the highest count of true-positive base pairs and the structures from the best overall Nussinov runs. In each case, *P-RnaPredict* is able to dramatically outperform the

Table 8.11: Comparison of false-positive totals between the best results, in terms of matching known base pairs, from the Nussinov DPA, and the best experiment in terms of minimum free energy from *P-RnaPredict.*

Sequence	**DPA Weights**	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
X. laevis	3:2:1	286	175.3	47	65.7	OX2-100-10-INNHB
D. virilis	1:1:1	291	197.5	29	40.6	OX2-100-10-INN
H. rubra	3:2:1	174	117.7	31	43.7	OX2-100-10-INNHB
H. maris-mortui	1:1:1	37	17	8	16	ALL-ALL-ALL-ALL
S. cere-visiae	1:1:1	17	6	28	33	ALL-ALL-ALL-INNHB

Table 8.12: Comparison of false-positive totals between the best results, in terms of matching known base pairs, from the Nussinov DPA and the single lowest energy runs with *P-RnaPredict*

Sequence	**DPA Weights**	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
X. laevis	3:2:1	286	158	47	100	OX2-100-10-INN
D. virilis	1:1:1	291	203	29	49	OX2-100-10-INNHB
H. rubra	3:2:1	174	113	31	53	CX-100-10-INNHB
H. marismortui	1:1:1	37	17	8	16	ALL-ALL-ALL-ALL
S. cerevisiae	1:1:1	17	6	28	33	ALL-ALL-ALL-INNHB

Nussinov DPA. Especially notable are the *Hildenbrandia rubra* and *Haloarcula marismortui* results.

Table 8.13: Comparison of false-positive totals between the best results, in terms of matching known base pairs, from the Nussinov DPA and the runs predicting the highest number of known base pairs with *P-RnaPredict*

Sequence	**DPA Weights**	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
X. laevis	3:2:1	286	158	47	100	OX2-100-10-INN
D. virilis	1:1:1	291	168	29	66	CX-100-10-INN
H. rubra	3:2:1	174	88	31	71	OX2-100-10-INNHB
H. marismortui	1:1:1	37	17	8	16	ALL-ALL-ALL-ALL
S. cerevisiae	1:1:1	17	6	28	33	ALL-ALL-ALL-INNHB

Overall, the results indicate that in terms of false-positive base pairs for the five sequences presented *P-RnaPredict* consistently predicts considerably fewer false-positives than the Nussinov DPA. It is especially notable that even the average *P-RnaPredict* result has significantly less false-positives than the best Nussinov structure. Overprediction data for other sequences can be found in Appendix A.

8.7 Summary

On the five sequences studied here, *P-RnaPredict* was able to drastically outperform the Nussinov DPA in all cases; both on average and single best runs. Considering true-positive base pairs, all *P-RnaPredict* results for the sequences presented are dramatically better than the best Nussinov result, even when considering average *P-RnaPredict* performance. This is also true when taking into account false-positive base pairs. All *P-RnaPredict* results

produced significantly less false-positives than the best Nussinov structure.

Data for five additional sequences may be found in Appendix A; a brief summary of these results follows. When considering the average performance of *P-RnaPredict* against the best Nussinov results in terms of matching base pair percentage, *P-RnaPredict* produced dramatically better structures for 3 sequences; Nussinov tied for one structure and outperformed *P-RnaPredict* for the fifth. Reviewing over-prediction, in all cases *P-RnaPredict* had dramatically less false-positive base pairs. Examining the best minimum free energy structures in terms of matching base pair percentage, *P-RnaPredict* significantly outperformed Nussinov in four out of the five structures. Also, in all cases *P-RnaPredict* again had noticeably less false-positive base pairs. Finally, when reviewing the best overall structures, *P-RnaPredict* again predicted a considerably higher proportion of known base pairs for four out of the five sequences. Once again, *P-RnaPredict* had dramatically fewer false-positives. One specific sequence, *S. acidocaldarius*, was particularly challenging for *P-RnaPredict* as compared to the Nussinov DPA. As yet, the reasons for this are unclear.

In conclusion, even when the Nussinov DPA is modified to support multiple base pair weightings, there is still a drastic performance difference between *P-RnaPredict* and the Nussinov algorithm. This is an important milestone which clearly demonstrates the benefits of the methods employed by *P-RnaPredict* against the pure base pair maximization approach utilized by the Nussinov DPA.

Chapter 9

Comparison to *mfold* DPA

As mentioned in Section 1.3.3, Zuker [113, 108, 110, 112, 107, 109] developed a DP approach to RNA secondary structure prediction which used thermodynamic models to minimize the free energy of the predicted structure. Unlike the Nussinov DPA, the free energy of helices is based on the stacking contribution from the interaction between base pairs. The DPA has also been modified to determine suboptimal RNA secondary structures within a specified range of the minimum free energy [41, 111, 53].

The *mfold* algorithm employs both the INN-HB thermodynamic model [105] and modeling of common RNA secondary structure element energy parameters. These elements include stacking energies, hairpin loop energies, interior loop energies, bulge loop energies, multi-branched loop energies, free base energies, 1×2 interior loop energies, tandem mismatch energies, loop destabilizing energies, tetra-loops, and other miscellaneous energies. Once *mfold* has predicted its structures, a helper application *efn2* with a more rigorous and computationally complex thermodynamic model re-evaluates the *mfold* structures.

This chapter presents a comparison of the structures predicted by *P-RnaPredict* with those predicted by *mfold*. The *mfold* results presented here were generated using *mfold* web server version 3.1 with default settings. One important setting was the "percent suboptimality number". This setting causes *mfold* to return a set of suboptimal structures whose free energy lies within the requested percentage of the minimum free energy. The setting in this case was 5%, which returns approximately 20 suboptimal structures on a 1000 nt sequence.

9.1 *Xenopus laevis* - 945 nt

The *mfold* results for *Xenopus laevis* are presented in Table 9.1. The table column headings are defined as follows: "*mfold* ΔG (kcal/mol)" is the ΔG of the predicted structure as optimized by *mfold*. "*efn2* ΔG (kcal/mol)" is the ΔG of the structure after re-evaluation with *efn2*. "Predicted BP" is the total number of base pairs in the predicted structure. "Correctly Predicted BP" is the total number of base pairs in the known structure. Finally, "% Correctly Predicted" is the percentage of correctly predicted base pairs in the predicted structure.

Table 9.1: *Xenopus laevis*, *mfold* results. Number of known base pairs is 251.

mfold ΔG **(kcal/mol)**	***efn2*** ΔG **(kcal/mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-250.6	-222.85	249	92	36.7
-249.6	-219.75	251	71	28.3
-248.8	-219.63	241	97	38.6
-248.6	-218.69	246	84	33.5
-248.0	-216.51	245	**113**	**45.0**
-248.0	-213.01	242	100	39.8
-247.8	-210.87	241	84	33.5
-247.4	-209.26	243	74	29.5
-247.2	-218.30	246	79	31.5
-247.1	-215.70	244	76	30.3
-246.7	-211.01	238	69	27.5
-246.5	-221.02	244	88	35.1
-246.5	-214.62	245	68	27.1
-246.3	-223.07	248	101	40.2
-245.3	-214.07	250	103	41.0
-245.0	-217.38	248	62	24.7
-244.7	-215.17	243	80	31.9
-244.3	**-223.49**	246	86	34.3
-243.7	-213.42	237	73	29.1
-243.6	-205.90	242	91	36.3
-242.5	-202.27	251	81	32.3

Table 9.1 indicates that the lowest ΔG structure found with *mfold* has a free energy of -250.6 kcal/mol and 249 total base pairs, 36.7% of which were present in the known structure. By contrast, *efn2* found a lowest energy structure with a ΔG of -223.49 kcal/mol

and 246 total base pairs, with 34.3% correctly predicted. Surprisingly, the *efn2* structure is less accurate than the *mfold* structure in terms of matching known base pairs.

Upon reviewing all results for the highest accuracy structure, one structure is found with 45.0% correctly predicted. *mfold* ranks this as fifth out of the 22 predicted structures. However, without comparison to the known structure there would have been no method of determining this structure's accuracy, and thus it would not have been found.

Table 9.2: Comparison of *P-RnaPredict* and *mfold* DPA on *Xenopus laevis* sequence.

	mfold Avg.	GA Avg. ΔG	*mfold* Best ΔG	*efn2* Best ΔG	GA Best ΔG	*mfold* Best BP %	GA Best BP %
Correct BP%	**33.6**	26.2	36.7	34.3	**39.8**	**45.0**	39.8
Over-prediction	**160.4**	175.3	**157**	160	158	**132**	158

Table 9.2 provides a direct comparison between the best *mfold* structures from Table 9.1 and the best *P-RnaPredict* structures from Tables 7.3, 7.4, and 7.5. Each result is identified by the following column headings: "*mfold* Avg." is the average over all *mfold* predicted structures. "GA Avg. ΔG" presents the best result from Table 7.3. "*mfold* Best ΔG" presents the best *mfold* structure as ranked by the *mfold* thermodynamic model. "*efn2* Best ΔG" presents the best *mfold* structure as ranked by the *efn2* thermodynamic model. "GA Best ΔG" presents the best result from Table 7.4. "*mfold* Best BP %" presents the best *mfold* structure in terms of percentage of matching known base pairs. Finally, "GA Best BP %" presents the best result from Table 7.5.

Each entry in the table is ranked by two criteria, identified by the following row headings: "Correct BP%" is the percentage of correctly predicted base pairs. "Over-prediction" is the number of false-positive base pairs predicted. Each set of related table entries are grouped for comparison, and the best results in the group are bolded for easy identification.

When considering average performance, Table 9.2 indicates that *mfold* produced a greater percentage of known base pairs, 33.6%, and a lower number of false-positives, 160.4, than *P-RnaPredict*. By contrast, the results ranked by minimum free energy show that the best *P-RnaPredict* structure matched 39.8% of the known structure, significantly better than both structures ranked by *mfold* and *efn2*. *P-RnaPredict* was also quite close with

respect to false-positives, narrowly outperformed by the best *mfold* structure. Finally, we review the best structures in terms of highest percentage of known base pairs regardless of their free energy. The best *mfold* structure has 45.0% matching base pairs, and 132 false-positive base pairs, significantly better than the best *P-RnaPredict* structure.

An important point should be kept in mind when regarding the comparison of highest matching base pair count structures. Since there is no perfect correlation between the lowest free energy and the highest matching base pair count, it is impossible to determine the structure with the best base pair overlap without having a known structure in advance. This is a general limitation from which all structure prediction algorithms based on free energy models suffer.

Overall for the *Xenopus laevis* structure, *P-RnaPredict* has a comparable performance to *mfold*. When considering minimum free energy, *P-RnaPredict* is able to predict a significantly higher percentage of matching base pairs than *mfold*, and a nearly identical number of false-positives.

9.2 *Drosophila virilis* - 784 nt

The *mfold* results for *Drosophila virilis* are shown in Table 9.3. The lowest free energy structure predicted by *mfold* contained 236 total base pairs, 15.9% of which were correctly predicted. The structure predicted by *efn2* had a free energy of -131.55 kcal/mol, and contained 254 base pairs, 14.2% of which were correctly predicted. Notably, the *efn2* predicted structure is less accurate in terms of correct base pairs than the lowest free energy *mfold* result.

Considering structures in terms of correct base pairs, there are two results which tie, containing 35.2% of the known base pairs. Since only free energy can realistically be considered as a ranking criteria, these structures could not have been found.

Table 9.4 summarizes the best *mfold* structures from Table 9.3 and the best *P-RnaPredict* structures from Tables 7.7, 7.8, and 7.9. Comparing the average results from the two algorithms, the performance of *P-RnaPredict* is quite close to that of *mfold*, with *P-RnaPredict* producing a nearly identical number of known base pairs and significantly less false-positives. Contrasting the minimum free energy results of *P-RnaPredict* and *mfold*, *P-RnaPredict* determined a substantially higher fraction of the known structure than *mfold*, and only

Table 9.3: *Drosophila virilis*, *mfold* results. Number of known base pairs is 233.

mfold ΔG **(kcal/mol)**	***efn2*** ΔG **(kcal/mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-146.3	-124.43	236	37	15.9
-146.3	-128.56	238	37	15.9
-146.2	-124.07	246	37	15.9
-146.1	-126.92	243	21	9.0
-145.8	-126.59	257	37	15.9
-145.5	-123.19	253	68	29.2
-145.4	-123.30	261	44	18.9
-145.1	-126.92	232	27	11.6
-145.0	-123.57	256	37	15.9
-144.7	-128.43	265	49	21.0
-144.4	-125.39	271	31	13.3
-144.3	-125.03	246	38	16.3
-144.2	-124.69	228	33	14.2
-144.2	-124.04	247	37	15.9
-143.9	-121.32	249	27	11.6
-143.7	-129.30	251	28	12.0
-143.5	-122.97	245	37	15.9
-142.9	-120.17	253	68	29.2
-142.8	-120.26	252	**82**	**35.2**
-142.5	-122.76	230	26	11.2
-142.4	-116.91	237	22	9.4
-142.4	-121.04	255	**82**	**35.2**
-142.3	-123.88	253	38	16.3
-142.1	-126.36	249	21	9.0
-141.8	-118.65	246	79	33.9
-141.4	-125.98	244	28	12.0
-141.2	**-131.55**	254	33	14.2
-141.1	-120.77	242	39	16.7
-140.1	-116.53	243	38	16.3
-140.0	-115.66	235	36	15.4
-139.9	-119.18	249	76	32.6
-139.7	-126.74	260	44	18.9
-139.3	-121.60	246	52	22.3
-139.0	-122.10	238	22	9.4

Table 9.4: Comparison of *P-RnaPredict* and *mfold* DPA on *Drosophila virilis* sequence.

	mfold Avg.	GA Avg. ΔG	*mfold* Best ΔG	*efn2* Best ΔG	GA Best ΔG	*mfold* Best BP %	GA Best BP %
Correct BP%	**17.9**	17.4	15.9	14.2	**21.3**	**35.2**	28.3
Over-prediction	206.1	**197.5**	**199**	221	203	170	**168**

slightly more false-positives than *mfold*'s best result. Finally, when examining the best results regardless of minimum free energy from Table 9.4, *mfold* is significantly better than *P-RnaPredict* in matching the known structure, while *P-RnaPredict* produced slightly less false-positives.

For the *Drosophila virilis* sequence, both algorithms appear to demonstrate a much weaker correlation between lowest free energy and structure accuracy than with the longer *Xenopus laevis* sequence. However, when considering average performance and minimum free energy structures, *P-RnaPredict* either performed comparably to or significantly outperformed *mfold*, especially in terms of matching known base pairs.

9.3 *Hildenbrandia rubra* - 543 nt

Table 9.5 indicates that the lowest ΔG structure found with *mfold* has a free energy of -204.9 kcal/mol and 176 total base pairs, 35.5% of which were present in the known structure. By contrast, *efn2* found a lowest energy structure with a ΔG of -199.63 kcal/mol and 171 total base pairs, with 27.5% correctly predicted. Interestingly, the *efn2* structure is less accurate than the *mfold* structure in terms of matching known base pairs. An alternate structure predicted by *mfold* contained 167 base pairs, 60.1% of which are present in the known structure. While this was a dramatic improvement, it also had a significantly higher free energy.

Table 9.6 gathers the best *mfold* structures from Table 9.5 and the best *P-RnaPredict* structures from Tables 7.10, 7.11, and 7.12. Reviewing the results from Table 9.6 in terms of average performance and best base pair overlap, it can be seen that *mfold* produced significantly better structures than *P-RnaPredict*. However, the minimum free energy results

Table 9.5: *Hildenbrandia rubra*, *mfold* results. Number of known base pairs is 138.

mfold ΔG **(kcal/mol)**	***efn2*** ΔG **(kcal/mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-204.9	-199.11	176	49	35.5
-204.6	**-199.63**	171	38	27.5
-203.9	-191.61	169	53	38.4
-203.4	-191.34	168	61	44.2
-203.3	-195.23	172	71	51.4
-202.6	-198.12	175	40	29.0
-202.3	-184.69	160	47	34.1
-202.0	-191.25	167	42	30.4
-201.7	-191.10	164	65	47.1
-201.5	-183.70	161	72	52.1
-201.5	-195.42	170	57	41.3
-201.1	-191.43	162	46	33.3
-201.0	-186.13	164	68	49.2
-200.8	-188.36	172	57	41.3
-200.8	-185.21	165	50	36.2
-200.6	-183.94	173	67	48.6
-200.3	-193.28	169	55	39.9
-200.2	-194.41	171	64	46.4
-199.9	-192.22	170	42	30.4
-199.9	-190.14	167	57	41.3
-198.7	-190.11	163	40	29.0
-198.5	-191.09	175	66	47.8
-197.7	-186.14	166	44	31.9
-197.0	-188.83	161	65	47.1
-196.6	-183.74	167	**83**	**60.1**
-195.9	-185.17	179	57	41.3
-195.9	-183.87	176	37	26.8
-195.8	-184.93	160	41	29.7
-195.2	-187.12	175	50	36.2

Table 9.6: Comparison of *P-RnaPredict* and *mfold* DPA on *Hildenbrandia rubra* sequence.

	mfold Avg.	GA Avg. ΔG	*mfold* Best ΔG	*efn2* Best ΔG	GA Best ΔG	*mfold* Best BP %	GA Best BP %
Correct BP%	**39.6**	31.7	35.5	27.5	**38.4**	**60.1**	51.4
Over-prediction	**113.9**	117.7	127	133.0	**113**	**84**	88

indicate that *P-RnaPredict* outperformed the best *mfold* structure both in terms of a higher matching base pair count and a lower number of false-positives.

Overall, *P-RnaPredict* performed comparably to *mfold* on the *Hildenbrandia rubra* sequence, and outperformed *mfold* when considering minimum free energy results.

9.4 *Haloarcula marismortui* - 122 nt

The single *mfold* result for *Haloarcula marismortui* is presented in Table 9.7. Note that this table contains dramatically fewer results. This is due to the short length of the *Haloarcula marismortui* sequence, and the corresponding reduction of the search space in terms of structures having free energies within 5% of the lowest energy structure. The one structure found by *mfold* contained 34 base pairs, 76.3% were correctly predicted.

Table 9.7: *Haloarcula marismortui*, *mfold* results. Number of known base pairs is 38.

mfold ΔG **(kcal/mol)**	***efn2*** ΔG **(kcal/mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-59.5	-56.44	34	29	76.3

Table 9.8 gathers the best *mfold* structure from Table 9.7 and the best *P-RnaPredict* structures from Tables 7.13, 7.14, and 7.15. Table 9.8 indicates that for all categories *P-RnaPredict* converged to identical structures. Overall, even considering individual experiment runs *P-RnaPredict* was significantly outperformed by *mfold*, especially in terms of matching the known structure.

Table 9.8: Comparison of *P-RnaPredict* and *mfold* DPA on *Haloarcula marismortui* sequence.

	mfold Avg.	GA Avg. ΔG	*mfold* Best ΔG	*efn2* Best ΔG	GA Best ΔG	*mfold* Best BP %	GA Best BP %
Correct BP%	**76.3**	42.1	**76.3**	**76.3**	42.1	**76.3**	42.1
Over-prediction	**5.0**	17	**5**	**5**	17	**5**	17

9.5 *Saccharomyces cerevisiae* - 118 nt

Table 9.9 presents the two *mfold* results for the *Saccharomyces cerevisiae* sequence. Here again only a few structures were predicted, due to the small search space. The lowest free energy structure predicted by *mfold* contained 41 total base pairs, 89.2% of which were correctly predicted. By contrast, *efn2* determined a structure containing 42 total base pairs, with 75.7% correctly predicted.

Table 9.9: *Saccharomyces cerevisiae*, *mfold* results. Number of known base pairs is 37.

mfold ΔG **(kcal/mol)**	***efn2*** ΔG **(kcal/mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-53.5	-50.70	41	**33**	**89.2**
-53.0	**-50.76**	42	28	75.7

Table 9.10 gathers the best *mfold* structures from Table 9.9 and the best *P-RnaPredict* structures from Tables 7.16, 7.17, and 7.18. Table 9.10 indicates that for all categories, as with *Haloarcula marismortui*, *P-RnaPredict* converged to identical structures. Interestingly, *mfold*'s more advanced *efn2* thermodynamic model produced a substantially inferior structure to all other results, both having a smaller known structure percentage and a larger number of false-positives. Overall, *P-RnaPredict* tied or surpassed the best *mfold* structure in average performance, minimum free energy, and best matching structure.

Table 9.10: Comparison of *P-RnaPredict* and *mfold* DPA on *Saccharomyces cerevisiae* sequence.

	mfold Avg.	GA Avg. ΔG	*mfold* Best ΔG	*efn2* Best ΔG	GA Best ΔG	*mfold* Best BP %	GA Best BP %
Correct BP%	82.4	**89.2**	**89.2**	75.7	**89.2**	**89.2**	**89.2**
Over-prediction	11.0	**6**	8	14	**6**	8	**6**

9.5.1 Secondary Structure Comparison

Figure 9.1 shows a comparison between the known secondary structure for the *Saccharomyces cerevisiae* sequence, and the best secondary structure predicted by the *mfold* DPA. Light grey bonds indicate predicted base pairs which are not present in the known structure. Black bonds indicate base pairs present both in the known and predicted structure. Base pairs only present in the known structure were omitted for clarity.

A comparison with Figure 7.2 and the structure predicted by *P-RnaPredict* in Figure 7.3 indicates that both *mfold* and *P-RnaPredict* were able to predict structures with 89.2% of the known base pairs. However, two additional false-positive base pairs were predicted by the *mfold* DPA.

Finally, *mfold* also predicted a 2 base pair long helix, visible on the top left of the plot, not present in the known structure. *P-RnaPredict* would not have predicted it as the minimum helix length required by its helix generation algorithm is three base pairs.

9.6 False-Positives: Over-prediction of base pairs

Table 9.11 presents a comparison of the total false-positive base pairs predicted by *P-RnaPredict* and the *mfold* DPA. Each *P-RnaPredict* entry was taken from the overall lowest energy structure, and the *mfold* entries were taken from the best overall *mfold* run in terms of lowest free energy. A breakdown of the table column names follows.

"Sequence" is the name of the organism the sequence was taken from. "DPA over-pred." is the number of false-positive base pairs predicted by the *mfold* DPA. "GA over-pred." is the number of false-positive base pairs predicted by *P-RnaPredict*. "DPA Corr. BPs" is

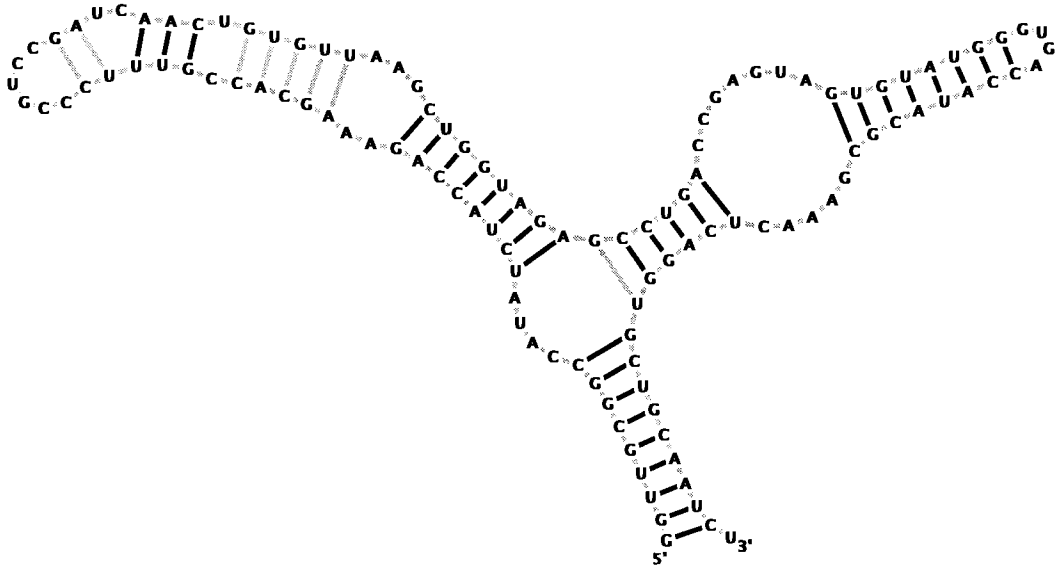

Figure 9.1: This plot shows a comparison between the known structure and the structure predicted by *mfold* for the *Saccharomyces cerevisiae* RNA sequence. Light grey lines indicate predicted base pairs, and black lines indicate the overlap between predicted and known base pairs. In this case, the *mfold* DPA was able to predict 89.2% of the known base pairs.

the number of true-positive base pairs predicted by the *mfold* DPA. "GA Corr. BPs" is the number of true-positive base pairs predicted by *P-RnaPredict*. Finally, "Cross.-Deme Size-Deme Count-Model" describes the parameters for the specific *P-RnaPredict* experiment, separated by dashes; these are the crossover, deme size, deme count, and thermodynamic model. An entry with "ALL" indicates that every setting for that parameter produced identical results.

The results indicate that *P-RnaPredict* was able to predict more true-positive base pairs than the *mfold* DPA for three sequences: *Xenopus laevis*, *Drosophila virilis* and *Hildenbrandia rubra*; *P-RnaPredict* tied for true-positive base pairs on *Saccharomyces cerevisiae*. Also, *P-RnaPredict* predicted less false-positive base pairs with two sequences, *Hildenbrandia rubra* and *Saccharomyces cerevisiae*.

Table 9.12 presents a comparison based on the overall highest number of correct base pairs predicted by both *mfold* and *P-RnaPredict*, regardless of energy. The column headings for Table 9.12 and Table 9.11 are identical. Table 9.12 indicates that *P-RnaPredict* predicted structures with less false-positive base pairs in two sequences (*Drosophila virilis* and

Table 9.11: Comparison of false-positive totals between the lowest *mfold* ΔG structure found with the *mfold* DPA and the overall lowest ΔG *P-RnaPredict* experiment

Sequence	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
X. laevis	157	158	92	100	OX2-100-10-INN
D. virilis	199	203	37	49	OX2-100-10-INNHB
H. rubra	127	113	49	53	CX-100-10-INNHB
H. marismortui	5	17	29	16	ALL-ALL-ALL-ALL
S. cerevisiae	8	6	33	33	ALL-ALL-ALL-INNHB

Saccharomyces cerevisiae). However, *mfold* was able to predict more true-positive base pairs in four sequences (*Xenopus laevis*, *Drosophila virilis*, *Hildenbrandia rubra*, and *Haloarcula marismortui*).

Overprediction data for other sequences can be found in Appendix A.

9.7 Summary

The *mfold* algorithm employs a DP approach to predict the secondary structure of RNA. It uses a complex thermodynamic model which includes all common RNA secondary structure elements to minimize the free energy of the predicted structure; this model is much more advanced than the Nussinov DPA discussed in Chapter 8. By contrast, *P-RnaPredict*'s thermodynamic model simply focuses on the free energy of nearest neighbour stacked base pairs in helices. *mfold* also possesses a much more mature implementation, compared with *P-RnaPredict*'s relatively short development lifespan.

Despite this, when comparing lowest energy structures *P-RnaPredict* was able to predict a higher percentage of known base pairs than *mfold* on the *Xenopus laevis*, *Drosophila virilis*, and *Hildenbrandia rubra* sequences. *P-RnaPredict* was also able to tie the percentage of

Table 9.12: Comparison of false-positive totals between the best structure with the *mfold* DPA and the overall best single structure found with*P-RnaPredict*

Sequence	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
X. laevis	132	158	113	100	OX2-100-10-INN
D. virilis	170	168	82	66	CX-100-10-INN
H. rubra	84	88	83	71	OX2-100-10-INNHB
H. marismortui	5	17	29	16	ALL-ALL-ALL-ALL
S. cerevisiae	8	6	33	33	ALL-ALL-ALL-INNHB

known base pairs with *mfold* on the *Saccharomyces cerevisiae* sequence. *P-RnaPredict* also predicted fewer false-positive base pairs on the *Hildenbrandia rubra* and *Saccharomyces cerevisiae* sequences, performing comparably to *mfold* on the other three.

For longer sequences, the prediction accuracy of *P-RnaPredict* drops; *mfold* also suffers from this problem. This can be attributed largely to limitations of the thermodynamic model regarding its ability to model global interactions as the structures grow larger, and is an issue from which all minimum free energy prediction algorithms suffer.

Data for five additional sequences may be found in Appendix A; a thorough comparison of *mfold* and *P-RnaPredict* performance in terms of known base pair agreement and false-positives for these sequences is provided there. A summary of the best overall *mfold* and *P-RnaPredict* structures in terms of known base pair agreement follows. For the *A. griffini* sequence, *P-RnaPredict* predicted 60.3% of the known structure, while *mfold* found 72.5%. The *H. sapiens* structure predicted by *P-RnaPredict* contained 34.6% correct base pairs; the *mfold* structure had 35.7%. The structure *P-RnaPredict* determined for *C. elegans* correctly predicted 29.6% of the known structure, whereas the *mfold* structure correctly predicted 21.2%. 64.6% of base pairs in the *A. lagunensis* structure determined by *P-RnaPredict* were present in the known structure; 65.5% of the *mfold* structure's base pairs were correctly

predicted. Finally, the *S. acidocaldarius* structure predicted by *P-RnaPredict* correctly predicted 34.0% of the known structure, while the *mfold* structure correctly predicted 57.9%.

It should be noted that one particular structure from the *S. acidocaldarius* sequence was particularly challenging for *P-RnaPredict* to predict; this was also the single structure on which the Nussinov DPA outperformed *P-RnaPredict.* The reasons for this as yet are unclear.

Given the extensive time and resources devoted to polishing *mfold*'s performance versus the relatively new *P-RnaPredict* implementation, the results are quite promising.

Chapter 10

Conclusion

In the course of this thesis I have presented the research conducted during the development of *P-RnaPredict*. *P-RnaPredict* is a complete object-oriented redesign and redevelopment of the original serial GA implementation developed in Dr. Wiese's lab. The primary objective here was the design and implementation of a fully parallel coarse-grained GA for RNA secondary structure prediction to run on a 128 node Beowulf cluster.

Several important milestones were reached while conducting this research. A series of potential parallel models were researched, and the coarse-grained distributed GA was selected as the model best suited for adoption. A serial simulation of the distributed GA was successfully developed to evaluate its benefits within the RNA structure prediction problem domain. Both a practical analysis and several runtime tests were conducted to substantiate the case for parallelization, and the results established the potential for a worthwhile speedup. The target platforms of the Nebula Beowulf cluster and MPI were established, and the three implementation challenges of control logic, data serialization, and random number generation were resolved. The importance of PRNGs in parallel GAs was established, and the impact of two distinct parallel PRNGs on the performance of *P-RnaPredict* were investigated. The DC, a parallelized MT, was consequently adopted for use in *P-RnaPredict*. The parallel speedup was investigated through empirical testing, and an analysis was performed on the communication timing. This confirmed that a worthwhile speedup had been produced through the parallelization of *P-RnaPredict*.

A comparison between 10 known structures and those predicted by *P-RnaPredict* was conducted with the following sequences: *Sulfolobus acidocaldarius* (1494 nt, Table A.23), *Homo sapiens* (954 nt, Table A.6), *Xenopus laevis* (945 nt, Table 7.2), *Drosophila virilis*

(784 nt, Table 7.6), *Caenorhabditis elegans* (697 nt, Table 4.1), *Acanthamoeba griffini* (556 nt, Table 4.2), *Hildenbrandia rubra* (543 nt, Table 4.3), *Aureoumbra lagunensis* (468 nt, Table A.17), *Haloarcula marismortui* (122 nt, Table 4.4), and *Saccharomyces cerevisiae* (118 nt, Table 4.5). The five sequences analyzed and discussed in detail were *Xenopus laevis*, *Drosophila virilis*, *Hildenbrandia rubra*, *Haloarcula marismortui*, and *Saccharomyces cerevisiae*. Data for the other five sequences can be found in Appendix A.

The results demonstrate that *P-RnaPredict* has a noteworthy competency in determining low ΔG structures. A clear correlation exists between a lower ΔG and the percentage of matching base pairs in the known structure. It was also determined that larger deme sizes had a significant impact on the results. This implies that *P-RnaPredict* can employ a larger overall population size to improve prediction quality and still retain a notable speedup. Lastly, non-canonical base pairs in naturally occurring structures cannot be modeled with current thermodynamic models. Overall, the prediction accuracy of *P-RnaPredict* is good, particularly so for shorter sequences.

A series of comparisons were performed with the Nussinov DPA. The best Nussinov structures were compared against the averaged, best minimum free energy, and best overall *P-RnaPredict* structures. When considering the percentage of matching base pairs in the known structure, the average *P-RnaPredict* performance was dramatically better on 8 of the 10 sequences. For both best minimum free energy and best overall structures, *P-RnaPredict* found a substantially higher percentage of matching base pairs than the Nussinov DPA on 9 of the 10 sequences. When considering over-prediction, in all cases *P-RnaPredict* found a lower amount of false-positive base pairs than Nussinov. Thus, even though the Nussinov DPA was modified to support multiple base pair weightings, *P-RnaPredict* dramatically outperformed it. This important milestone clearly establishes the benefits of the free energy minimization method employed by *P-RnaPredict* against the Nussinov base pair maximization approach.

When a similar comparison was performed with the *mfold* DPA, *P-RnaPredict* was found to perform comparably despite *mfold*'s much more sophisticated thermodynamic models and comparatively greater developmental maturity. When comparing lowest energy structures on the five sequences reviewed in depth, *P-RnaPredict* was able to predict a higher percentage of known base pairs than *mfold* on the *Xenopus laevis*, *Drosophila virilis*, and *Hildenbrandia rubra* sequences. *P-RnaPredict* tied the percentage of known base pairs with *mfold* on the *Saccharomyces cerevisiae* sequence. *P-RnaPredict* also predicted fewer false-positive

base pairs on the *Hildenbrandia rubra* and *Saccharomyces cerevisiae* sequences, performing comparably to *mfold* on the other three. Comparable performance was also seen on the five sequences reviewed in Appendix A. For longer sequences, the prediction accuracy of *P-RnaPredict* drops; *mfold* also suffers from this problem. This can be attributed largely to limitations of the thermodynamic model regarding its ability to model global interactions as the structures grow larger, and is an issue from which all minimum free energy prediction algorithms suffer. Given the extensive prior development time and refinement of *mfold*'s thermodynamic model, *P-RnaPredict* was still able to offer a comparable performance in terms of solution quality despite its relative novelty.

Overall, the prediction accuracy of *P-RnaPredict* is good, particularly so for shorter sequences. With the 118 nt *Saccharomyces cerevisiae* sequence, *P-RnaPredict* was able to predict up to 89.2% of the known base pairs. Given the rigid constraints of its helix generation algorithm and thermodynamic models, significant achievements were made in comparison to other established RNA structure prediction algorithms.

10.1 Future Work

Future work may involve four important subproblems as suggested by my prior research. They are as follows:

First, *P-RnaPredict*'s method of helix generation results in predicted structures with reduced accuracy when compared with known structures determined by comparative methods. The GA employs a nearest-neighbour thermodynamic model in its fitness function, INN-HB [105] which accounts for varying terminal base-pairs in a given helix. Currently, *P-RnaPredict* will always generate a helix which includes all possible stacked complementary base pairs. However, the helices in known structures examined to date do not always include all complementary base pairs. Modifying the helix generation algorithm to produce partially complete helices would enable the fitness function to discriminate between partially complete helices and thus would greatly improve the GA's accuracy. One caveat is that generating every single possible partially complete helix would make the search space far too large; a reasonable threshold must be found.

Another problem lies in the thermodynamic models themselves. As mentioned, the nearest-neighbour models the GA employs result in a much higher matching base-pair accuracy than the original simple hydrogen bond weighting which was first used. However,

when the correlation between the free energy of a given structure and the matching base-pair percentage is examined, it varies considerably depending on the tested RNA sequence. Also, quite often a given sequence may produce many structures possessing dissimilar base pairs but identical free energy values. One strategy to employ to resolve this issue is to use more detailed thermodynamic models to differentiate these structures in the fitness function. To reduce computational complexity these advanced models can be employed on a subset of the population, only when the free energy of the GA population has converged to a pre-specified degree. A similar technique is employed in the DP application *mfold* [55], whereby predicted structures are re-evaluated with a more complex thermodynamic model (*efn2*).

A significant challenge is the modeling of non-canonical base pairs. These are an important part of many RNA structures, and incorporating them into the helix generation algorithm would improve *P-RnaPredict*'s results by a large margin. An important caveat is that this would also dramatically increase the size of the search space generated by *P-RnaPredict*.

The last problem I will examine is that of predicting pseudoknots. Pseudoknots are a type of RNA substructure which occurs when bases inside a hairpin loop pair with bases outside the loop, and they are important for a number of RNA functions. Although the GA has the ability to predict pseudoknots, it currently overpredicts them by a wide margin because there is no thermodynamic penalty in the fitness function for scoring them. I intend to analyze data on known pseudoknotted structures in order to explicitly model them [49] and thus to incorporate their prediction into the GA.

Appendix A

Data for other sequences

A.1 *Acanthamoeba griffini* - 556 nt

Details for this sequence may be found in Table 4.2.

Table A.1: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Acanthamoeba griffini* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 131 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-200.24	164.3	**48.0**	**36.6**	OX2	100	10	INNHB
-194.21	161.6	40.5	30.9	CX	100	10	INNHB
-193.77	159.9	40.6	31.0	OX2	70	10	INNHB
-192.99	159.9	41.6	31.7	OX2	50	14	INNHB
-189.35	158.6	38.3	29.3	CX	70	10	INNHB
-188.29	157.6	35.3	26.9	CX	50	14	INNHB
-182.2	164.2	38.7	29.5	OX2	100	10	INN
-179.5	162.4	36.8	28.1	OX2	70	10	INN
-177.4	161.0	**42.2**	**32.2**	CX	100	10	INN
-177.4	163.7	38.7	29.5	OX2	50	14	INN
-173.2	161.3	32.6	24.9	CX	50	14	INN
-172.8	160.2	36.4	27.8	CX	70	10	INN

Table A.2: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Acanthamoeba griffini* structure. Results are grouped by thermodynamic model. The known structure contains 131 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-208.36	1	677	167	54	41.2	OX2	100	10	INNHB
-204.54	1	671	167	46	35.1	CX	100	10	INNHB
-204.39	1	677	173	50	38.2	CX	50	14	INNHB
-202.41	1	392	165	**60**	**45.8**	OX2	70	10	INNHB
-202.64	1	377	158	34	26.0	OX2	50	14	INNHB
-199.84	1	452	161	37	28.2	CX	70	10	INNHB
-193.5	1	291	173	58	44.3	OX2	100	10	INN
-191.5	1	632	163	42	32.1	CX	100	10	INN
-190.5	1	641	169	30	22.9	OX2	70	10	INN
-190.7	1	562	172	**76**	**58.0**	OX2	50	14	INN
-189.3	1	481	169	45	34.4	CX	50	14	INN
-187.4	1	396	166	50	38.2	CX	70	10	INN

Table A.3: Single run with the highest number of correctly predicted base pairs of *Acanthamoeba griffini*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 131 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-202.41	2	506.5	166.5	60	45.8	OX2	70	10	INNHB
-202.20	1	357	176	76	58.0	CX	100	10	INNHB
-198.18	1	555	164	79	60.3	OX2	100	10	INNHB
-194.05	1	659	157	64	48.9	CX	50	14	INNHB
-192.77	1	459	157	**79**	**60.3**	CX	70	10	INNHB
-192.99	1	673	167	61	46.6	OX2	50	14	INNHB
-187.8	1	557	170	**78**	**59.5**	OX2	50	14	INN
-183.9	1	405	164	65	49.6	OX2	100	10	INN
-181.5	1	689	169	53	40.5	CX	50	14	INN
-181.2	1	477	162	51	38.9	OX2	70	10	INN
-179.1	1	355	169	66	50.4	CX	70	10	INN
-176.3	1	528	160	64	48.9	CX	100	10	INN

Table A.4: *Acanthamoeba griffini*, Nussinov results. Number of known base pairs is 131.

DPA Weights	**Predicted BP**	**Correctly Predicted BP**	**Correctly Predicted (%)**
1:1:1	215	40	30.5
3:2:1	208	37	28.2
3:2:2	214	48	36.6

Table A.5: *Acanthamoeba griffini*, *mfold* results. Number of known base pairs is 131.

mfold ΔG **(kcal / mol)**	***efn2*** ΔG **(kcal / mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-193.0	-179.03	172	67	51.1
-192.7	-182.60	173	62	47.3
-191.8	-177.46	175	56	42.7
-190.3	-177.40	172	56	42.7
-190.3	-171.34	170	59	45.0
-189.9	-181.63	175	63	48.1
-189.6	-178.83	172	69	52.7
-188.7	**-182.67**	171	64	48.9
-188.3	-174.90	174	**95**	**72.5**
-188.2	-174.92	173	53	40.5
-187.8	-181.91	177	63	48.1
-187.4	-170.18	168	52	39.7
-187.2	-180.14	173	90	68.7
-187.0	-173.47	169	89	67.9
-186.6	-170.97	173	67	51.1
-186.3	-167.52	165	44	33.6
-184.1	-173.14	177	63	48.1

A.2 *Homo sapiens* - 954 nt

Table A.6: *Homo sapiens* details, taken from the Comparative RNA Web Site [4]

Filename	d.16.m.H.sapiens.bpseq
Organism	*Homo sapiens*
Accession Number	J01415
Class	16S rRNA
Length	954 nucleotides
# of BPs in known structure	266
# of non-canonical base pairs	30

Table A.7: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Homo sapiens* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 266 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-275.72	239.3	**48.9**	**18.4**	OX2	100	10	INNHB
-269.40	237.5	44.5	16.7	OX2	70	10	INNHB
-267.95	237.5	42.8	16.1	OX2	50	14	INNHB
-264.82	234.3	39.9	15.0	CX	100	10	INNHB
-258.78	232.1	39.4	14.8	CX	70	10	INNHB
-254.71	230.5	32.7	12.3	CX	50	14	INNHB
-265.9	243.7	40.4	15.2	OX2	100	10	INN
-259.4	240.5	40.7	15.3	CX	100	10	INN
-258.4	238.8	**42.5**	**16.0**	OX2	70	10	INN
-256.5	238.4	40.0	15.0	OX2	50	14	INN
-251.4	237.7	34.4	12.9	CX	70	10	INN
-249.8	236.3	36.0	13.5	CX	50	14	INN

Table A.8: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Homo sapiens* structure. Results are grouped by thermodynamic model. The known structure contains 266 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-293.56	1	649	253	66	24.8	OX2	100	10	INNHB
-293.37	1	699	240	41	15.4	OX2	70	10	INNHB
-287.73	1	519	241	**79**	**29.7**	OX2	50	14	INNHB
-284.21	1	617	248	32	12.0	CX	50	14	INNHB
-278.41	1	661	240	18	6.8	CX	70	10	INNHB
-276.85	1	680	235	43	16.2	CX	100	10	INNHB
-282.8	1	683	251	52	19.5	OX2	100	10	INN
-277.5	1	689	246	46	17.3	OX2	50	14	INN
-276.4	1	687	244	**69**	**25.9**	OX2	70	10	INN
-274.9	2	457.0	240.5	46.5	17.5	CX	100	10	INN
-271.8	1	526	248	31	11.7	CX	70	10	INN
-270.5	1	671	244	44	16.5	CX	50	14	INN

Table A.9: Single run with the highest number of correctly predicted base pairs of *Homo sapiens*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 266 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-287.73	1	519	241	79	29.7	OX2	50	14	INNHB
-278.01	1	651	239	71	26.7	OX2	100	10	INNHB
-273.36	1	607	227	**92**	**34.6**	OX2	70	10	INNHB
-270.91	1	423	240	72	27.1	CX	100	10	INNHB
-267.19	1	698	241	72	27.1	CX	70	10	INNHB
-251.05	1	440	228	53	19.9	CX	50	14	INNHB
-271.7	1	668	251	59	22.2	OX2	100	10	INN
-270.7	1	664	244	73	27.4	OX2	50	14	INN
-269.5	1	670	238	**73**	**27.4**	CX	70	10	INN
-268.0	1	546	242	65	24.4	CX	100	10	INN
-264.8	1	508	243	71	26.7	OX2	70	10	INN
-246.8	1	690	238	67	25.2	CX	50	14	INN

Table A.10: *Homo sapiens*, Nussinov results. Number of known base pairs is 266.

DPA Weights	**Predicted BP**	**Correctly Predicted BP**	**Correctly Predicted (%)**
1:1:1	342	33	12.4
3:2:1	333	22	8.2
3:2:2	339	32	12.0

Table A.11: *Homo sapiens*, *mfold* results. Number of known base pairs is 266.

mfold ΔG **(kcal / mol)**	***efn2*** ΔG **(kcal / mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-250.9	-217.20	258	**95**	**35.7**
-250.5	**-222.51**	255	91	34.2
-248.7	-213.42	251	84	31.6
-247.5	-213.14	262	44	16.5
-247.2	-219.97	251	51	19.2
-247.2	-214.50	256	82	30.8
-247.0	-207.98	260	52	19.5
-246.7	-210.39	255	82	30.8
-246.3	-207.85	257	81	30.5
-246.0	-206.53	256	37	13.9
-244.8	-217.55	255	57	21.4
-243.7	-205.51	262	43	16.2
-243.4	-198.70	259	50	18.8
-243.2	-211.12	258	67	25.2
-243.1	-198.17	258	44	16.5
-241.6	-202.39	245	51	19.2
-241.5	-210.26	257	59	22.2
-240.9	-219.87	259	50	18.8
-240.8	-204.14	266	57	21.4

A.3 *Caenorhabditis elegans* - 697 nt

Details for this sequence may be found in Table 4.1.

Table A.12: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Caenorhabditis elegans* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 189 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-167.38	208.4	25.4	13.5	OX2	100	10	INNHB
-161.50	205.6	**26.8**	**14.2**	OX2	70	10	INNHB
-160.24	203.3	25.0	13.2	CX	100	10	INNHB
-159.65	203.4	22.4	11.9	OX2	50	14	INNHB
-156.37	202.1	25.6	13.6	CX	50	14	INNHB
-153.21	199.4	22.3	11.8	CX	70	10	INNHB
-149.4	203.4	30.5	16.1	OX2	100	10	INN
-145.5	203.9	**30.7**	**16.3**	OX2	70	10	INN
-143.7	199.1	26.8	14.2	CX	100	10	INN
-141.4	202.1	26.5	14.0	OX2	50	14	INN
-140.1	199.2	25.2	13.4	CX	70	10	INN
-137.9	197.9	28.8	15.2	CX	50	14	INN

Table A.13: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Caenorhabditis elegans* structure. Results are grouped by thermodynamic model. The known structure contains 189 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-176.96	1	442	208	35	18.5	OX2	100	10	INNHB
-176.9	1	488	204	30	15.9	OX2	70	10	INNHB
-173.32	1	307	212	27	14.3	CX	100	10	INNHB
-172.35	1	593	204	28	14.8	CX	50	14	INNHB
-170.67	1	347	203	**38**	**20.1**	OX2	50	14	INNHB
-169.94	1	661	197	36	19.0	CX	70	10	INNHB
-161.0	1	457	211	35	18.5	OX2	70	10	INN
-160.2	1	680	208	34	18.0	OX2	100	10	INN
-156.7	1	666	192	29	15.3	CX	100	10	INN
-154.2	1	459	206	24	12.7	OX2	50	14	INN
-150.7	1	610	206	**54**	**28.6**	CX	50	14	INN
-150.1	1	495	201	35	18.5	CX	70	10	INN

Table A.14: Single run with the highest number of correctly predicted base pairs of *Caenorhabditis elegans*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 189 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-171.50	1	201	211	44	23.3	OX2	100	10	INNHB
-168.29	1	651	208	38	20.1	OX2	70	10	INNHB
-164.05	1	336	205	**46**	**24.3**	CX	50	14	INNHB
-163.42	1	513	204	41	21.7	CX	100	10	INNHB
-163.14	1	390	203	40	21.2	OX2	50	14	INNHB
-152.03	2	419.00	203.50	38	20.1	CX	70	10	INNHB
-157.7	1	632	207	46	24.3	OX2	100	10	INN
-150.7	1	610	206	54	28.6	CX	50	14	INN
-141.4	1	542	204	38	20.1	OX2	50	14	INN
-138.9	2	313.0	191.5	40	21.2	CX	100	10	INN
-137.9	1	484	193	36	19.0	CX	70	10	INN
-134.1	1	507	202	**56**	**29.6**	OX2	70	10	INN

Table A.15: *Caenorhabditis elegans*, Nussinov results. Number of known base pairs is 189.

DPA Weights	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
1:1:1	284	10	5.2
3:2:1	275	20	10.5
3:2:2	281	26	13.7

Table A.16: *Caenorhabditis elegans*, *mfold* results. Number of known base pairs is 189.

mfold ΔG **(kcal / mol)**	***efn2*** ΔG **(kcal / mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-142.1	-125.22	217	**40**	**21.2**
-141.3	-123.20	219	32	16.9
-141.2	-124.04	216	**40**	**21.2**
-140.6	-124.20	211	**40**	**21.2**
-137.9	-126.18	221	25	13.2
-137.6	-121.46	219	25	13.2
-137.5	-123.11	216	**40**	**21.2**
-137.3	-121.59	213	**40**	**21.2**
-137.0	-122.10	211	20	10.6
-136.8	-123.68	212	37	19.6
-136.4	-118.90	211	27	14.3
-136.2	-126.56	221	25	13.2
-136.2	**-128.97**	216	20	10.6
-136.1	-115.94	200	27	14.3
-135.9	-117.46	206	32	16.9
-135.7	-120.06	208	35	18.5
-135.5	-118.87	206	27	14.3
-135.5	-120.81	216	37	19.6
-135.4	-122.56	218	35	18.5
-135.1	-125.99	213	20	10.6

A.4 *Aureoumbra lagunensis* - 468 nt

Table A.17: *Aureoumbra lagunensis* details, taken from the Comparative RNA Web Site [4]

Filename	b.I1.e.A.lagunensis.C1.SSU.516.bpseq
Organism	*Aureoumbra lagunensis*
Accession Number	U40258
Class	Group I intron, 16S rRNA
Length	468 nucleotides
# of BPs in known structure	113
# of non-canonical base pairs	4

Table A.18: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Aureoumbra lagunensis* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 113 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-175.35	129.8	**47.6**	**42.2**	OX2	100	10	INNHB
-173.36	128.3	43.7	38.6	OX2	70	10	INNHB
-171.66	129.6	44.5	39.4	CX	100	10	INNHB
-170.36	127.1	42.0	37.2	OX2	50	14	INNHB
-166.29	127.5	40.1	35.5	CX	50	14	INNHB
-165.80	126.9	38.3	33.9	CX	70	10	INNHB
-170.0	129.9	**42.5**	**37.6**	OX2	100	10	INN
-167.1	128.4	40.5	35.8	OX2	70	10	INN
-165.8	129.2	39.3	34.7	CX	100	10	INN
-165.5	127.3	37.5	33.2	OX2	50	14	INN
-162.7	127.4	38.4	34.0	CX	50	14	INN
-161.3	125.7	34.2	30.3	CX	70	10	INN

Table A.19: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Aureoumbra lagunensis* structure. Results are grouped by thermodynamic model. The known structure contains 113 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-182.06	1	348	137	51	45.1	CX	100	10	INNHB
-182.06	1	145	137	51	45.1	OX2	70	10	INNHB
-182.06	4	367.8	137.0	51.0	45.1	OX2	100	10	INNHB
-182.06	1	509	137	51	45.1	CX	70	10	INNHB
-181.81	1	141	141	51	45.1	CX	50	14	INNHB
-181.40	1	578	132	**57**	**50.4**	OX2	50	14	INNHB
-179.4	1	528	134	53	46.9	OX2	70	10	INN
-179.4	2	560.0	134.0	53.0	46.9	OX2	100	10	INN
-179.4	2	408.0	134.0	53.0	46.9	CX	100	10	INN
-177.8	1	387	138	**56**	**49.6**	CX	50	14	INN
-177.6	1	122	135	47	41.6	OX2	50	14	INN
-177.6	1	261	135	47	41.6	CX	70	10	INN

Table A.20: Single run with the highest number of correctly predicted base pairs of *Aureoumbra lagunensis*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 113 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-180.47	1	103	127	64	56.6	OX2	70	10	INNHB
-177.10	1	699	131	72	63.7	OX2	100	10	INNHB
-177.06	1	618	131	70	61.9	CX	100	10	INNHB
-176.83	1	650	132	**73**	**64.6**	CX	70	10	INNHB
-175.82	1	632	130	68	60.2	OX2	50	14	INNHB
-169.24	1	698	121	70	61.9	CX	50	14	INNHB
-177.1	1	419	133	71	62.8	OX2	70	10	INN
-174.4	1	299	134	62	54.9	CX	100	10	INN
-172.0	1	371	131	66	58.4	CX	70	10	INN
-169.2	1	606	130	71	62.8	OX2	100	10	INN
-168.7	1	341	123	58	51.3	OX2	50	14	INN
-163.4	1	377	126	**71**	**62.8**	CX	50	14	INN

Table A.21: *Aureoumbra lagunensis*, Nussinov results. Number of known base pairs is 113.

DPA Weights	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
1:1:1	173	27	23.8
3:2:1	168	9	7.9
3:2:2	172	30	26.5

Table A.22: *Aureoumbra lagunensis*, *mfold* results. Number of known base pairs is 113.

mfold ΔG **(kcal / mol)**	***efn2*** ΔG **(kcal / mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-160.1	-142.35	128	60	53.1
-159.7	**-143.71**	136	60	53.1
-158.1	-141.78	134	60	53.1
-156.6	-143.17	134	61	54.0
-156.4	-138.52	133	63	55.8
-156.2	-140.50	132	60	53.1
-155.7	-143.49	137	72	63.7
-154.5	-141.88	131	72	63.7
-154.5	-138.76	130	72	63.7
-153.9	-136.16	133	48	42.5
-153.8	-136.47	140	60	53.1
-153.8	-140.57	133	**74**	**65.5**
-153.4	-134.89	125	51	45.1
-153.3	-140.79	131	60	53.1

A.5 *Sulfolobus acidocaldarius* - 1494 nt

Table A.23: *Sulfolobus acidocaldarius* details, taken from the Comparative RNA Web Site [4]

Filename	d.16.a.S.acidocaldarius.bpseq
Organism	*Sulfolobus acidocaldarius*
Accession Number	D14876
Class	16S rRNA
Length	1494 nucleotides
# of BPs in known structure	468
# of non-canonical base pairs	22

Table A.24: Comparison of average lowest ΔG *P-RnaPredict* structures with the known *Sulfolobus acidocaldarius* structure. Results are grouped by thermodynamic model. Each row represents an experiment consisting of 30 averaged runs. The known structure contains 468 base pairs.

ΔG (kcal / mol)	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-663.15	426.9	**98.5**	**21.0**	OX2	100	10	INNHB
-639.60	423.8	83.2	17.8	OX2	70	10	INNHB
-631.53	417.5	83.1	17.7	CX	100	10	INNHB
-626.80	416.0	74.8	16.0	OX2	50	14	INNHB
-602.46	409.0	66.5	14.2	CX	70	10	INNHB
-593.32	408.5	56.2	12.0	CX	50	14	INNHB
-621.6	431.2	**91.5**	**19.6**	OX2	100	10	INN
-602.8	421.5	76.9	16.4	OX2	70	10	INN
-591.9	417.9	73.7	15.8	OX2	50	14	INN
-587.1	417.2	76.8	16.4	CX	100	10	INN
-569.5	410.0	67.0	14.3	CX	70	10	INN
-554.5	404.9	58.9	12.6	CX	50	14	INN

Table A.25: Comparison of the best single run's lowest ΔG *P-RnaPredict* structure with the known *Sulfolobus acidocaldarius* structure. Results are grouped by thermodynamic model. The known structure contains 468 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-720.52	1	684	447	**159**	**33.9**	OX2	100	10	INNHB
-691.91	1	643	433	114	24.4	CX	100	10	INNHB
-691.64	1	684	439	130	27.8	OX2	70	10	INNHB
-678.59	1	681	434	75	16.0	OX2	50	14	INNHB
-645.60	1	677	419	91	19.4	CX	70	10	INNHB
-640.60	1	684	436	96	20.5	CX	50	14	INNHB
-666.7	1	636	442	90	19.2	OX2	100	10	INN
-648.6	1	681	435	**136**	**29.1**	OX2	50	14	INN
-641.7	1	682	419	94	20.1	CX	100	10	INN
-637.3	1	581	434	119	25.4	OX2	70	10	INN
-613.6	1	626	435	96	20.5	CX	70	10	INN
-595.0	1	697	419	95	20.3	CX	50	14	INN

Table A.26: Single run with the highest number of correctly predicted base pairs of *Sulfolobus acidocaldarius*, regardless of free energy. Results are grouped by thermodynamic model. The known structure contains 468 base pairs.

ΔG (kcal / mol)	Freq.	Gens	Pred. BPs	Corr. BPs	Corr. BPs (%)	Cross.	Deme Size	Deme Count	Model
-720.52	1	684	447	**159**	**34.0**	OX2	100	10	INNHB
-691.91	1	643	433	114	24.4	CX	100	10	INNHB
-691.64	1	684	439	130	27.8	OX2	70	10	INNHB
-675.10	1	695	430	111	23.7	OX2	50	14	INNHB
-644.97	1	688	417	115	24.6	CX	70	10	INNHB
-640.60	1	684	436	96	20.5	CX	50	14	INNHB
-648.6	1	681	435	**136**	**29.1**	OX2	50	14	INN
-647.6	1	681	432	124	26.5	OX2	100	10	INN
-637.3	1	581	434	119	25.4	OX2	70	10	INN
-636.7	1	689	445	134	28.6	CX	100	10	INN
-602.5	1	664	412	108	23.1	CX	70	10	INN
-581.7	1	671	418	104	22.2	CX	50	14	INN

Table A.27: *Sulfolobus acidocaldarius*, Nussinov results. Number of known base pairs is 468.

DPA Weights	Predicted BP	Correctly Predicted BP	Correctly Predicted (%)
1:1:1	584	187	39.9
3:2:1	570	143	30.5
3:2:2	582	187	39.9

Table A.28: *Sulfolobus acidocaldarius*, *mfold* results. Number of known base pairs is 468.

mfold ΔG **(kcal / mol)**	***efn2*** ΔG **(kcal / mol)**	**Predicted BP**	**Correctly Predicted BP**	**% Correctly Predicted**
-822.9	-781.20	494	261	55.8
-821.8	-773.67	493	243	51.9
-821.3	**-787.66**	496	266	56.8
-820.6	-777.39	496	**271**	**57.9**
-817.5	-766.00	493	240	51.3
-816.7	-779.52	487	270	57.7
-816.6	-766.23	495	285	60.9
-816.1	-774.22	485	247	52.8
-815.7	-779.32	494	243	51.9
-815.6	-779.82	492	237	50.6
-815.2	-776.33	489	249	53.2
-814.8	-761.41	491	230	49.1
-814.5	-768.46	495	243	51.9
-813.9	-762.55	491	229	48.9
-813.5	-772.38	490	254	54.3
-813.0	-783.78	489	241	51.5

A.6 Over-prediction of base pairs

This section will compare structures predicted by *P-RnaPredict*, Nussinov, and *mfold* for all sequences in terms of total false-positive base pairs.

Table A.29: Comparison between the number of false predictions between best results, in terms of correctly predicted base pairs, from the Nussinov DPA and the best experiments, in terms of minimum free energy, from *P-RnaPredict*

Sequence	DPA Weights	DPA over-pred.	GA over-pred.	DPA Corr. BPs	GA Corr. BPs	Cross.-Deme Size-Deme Count-Model
A. griffini	3:2:2	166	116.3	48	48.0	OX2-100-10-INNHB
A. lagunensis	3:2:2	142	82.2	30	47.6	OX2-100-10-INNHB
C. elegans	3:2:2	281	172.9	26	30.5	OX2-100-10-INN
D. virilis	1:1:1	291	197.5	29	40.6	OX2-100-10-INN
H. marismortui	1:1:1	37	17.0	8	16.0	ALL-ALL-ALL-ALL
H. rubra	3:2:1	174	117.7	31	43.7	OX2-100-10-INNHB
H. sapiens	1:1:1	309	190.4	33	48.9	OX2-100-10-INNHB
S. cerevisiae	1:1:1	17	6.0	28	33.0	ALL-ALL-ALL-INNHB
S. acidocaldarius	3:2:2	395	328.4	187	98.5	OX2-100-10-INNHB
X. laevis	3:2:1	286	175.3	47	65.7	OX2-100-10-INNHB

Table A.30: Comparison between the number of false predictions between best results, in terms of correctly predicted base pairs, from the Nussinov DPA and the single lowest energy runs with *P-RnaPredict*

Sequence	DPA Weights	DPA over-pred.	GA over-pred.	DPA Corr. BPs	GA Corr. BPs	Cross.-Deme Size-Deme Count-Model
A. griffini	3:2:2	166	115	48	58	OX2-100-10-INN
A. lagunensis	3:2:2	142	81	30	53	OX2-70-10-INN
C. elegans	3:2:2	281	173	26	35	OX2-100-10-INNHB
D. virilis	1:1:1	291	203	29	49	OX2-100-10-INNHB
H. marismortui	1:1:1	37	17	8	16	ALL-ALL-ALL-ALL
H. rubra	3:2:1	174	113	31	53	CX-100-10-INNHB
H. sapiens	1:1:1	309	187	33	66	OX2-100-10-INNHB
S. cerevisiae	1:1:1	17	6	28	33	ALL-ALL-ALL-INNHB
S. acidocaldarius	3:2:2	395	288	187	159	OX2-100-10-INNHB
X. laevis	3:2:1	286	158	47	100	OX2-100-10-INN

Table A.31: Comparison between the number of false predictions between best results, in terms of correctly predicted base pairs, from the Nussinov DPA and the runs predicting the highest number of known base pairs with *P-RnaPredict*

Sequence	**DPA Weights**	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
A. griffini	3:2:2	166	78	48	79	CX-70-10-INNHB
A. lagunensis	3:2:2	142	59	30	73	CX-70-10-INNHB
C. elegans	3:2:2	281	146	26	56	OX2-70-10-INN
D. virilis	1:1:1	291	168	29	66	CX-100-10-INN
H. marismortui	1:1:1	37	17	8	16	ALL-ALL-ALL-ALL
H. rubra	3:2:1	174	88	31	71	OX2-100-10-INNHB
H. sapiens	1:1:1	309	135	33	92	OX2-70-10-INNHB
S. cerevisiae	1:1:1	17	6	28	33	ALL-ALL-ALL-INNHB
S. acidocaldarius	3:2:2	395	288	187	159	OX2-100-10-INNHB
X. laevis	3:2:1	286	158	47	100	OX2-100-10-INN

Table A.32: Comparison between the number of false predictions between lowest energy structure found with the *mfold* DPA and the overall lowest energy single *P-RnaPredict* runs

Sequence	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
A. griffini	105	115	67	58	OX2-100-10-INN
A. lagunensis	68	81	60	53	OX2-70-10-INN
C. elegans	177	173	40	35	OX2-100-10-INNHB
D. virilis	199	203	37	49	OX2-100-10-INNHB
H. marismortui	5	17	29	16	ALL-ALL-ALL-ALL
H. rubra	127	113	49	53	CX-100-10-INNHB
H. sapiens	163	187	95	66	OX2-100-10-INNHB
S. acidocaldarius	233	288	261	159	OX2-100-10-INNHB
S. cerevisiae	8	6	33	33	ALL-ALL-ALL-INNHB
X. laevis	157	158	92	100	OX2-100-10-INN

Table A.33: Comparison between the number of false predictions between best structure with the *mfold* DPA and the overall best single structure found with *P-RnaPredict*

Sequence	**DPA over-pred.**	**GA over-pred.**	**DPA Corr. BPs**	**GA Corr. BPs**	**Cross.-Deme Size-Deme Count-Model**
A. griffini	79	78	95	79	CX-70-10-INNHB
A. lagunensis	59	59	74	73	CX-70-10-INNHB
C. elegans	177	146	40	56	OX2-70-10-INN
D. virilis	170	168	82	66	CX-100-10-INN
H. marismortui	5	17	29	16	ALL-ALL-ALL-ALL
H. rubra	84	88	83	71	OX2-100-10-INNHB
H. sapiens	163	135	95	92	OX2-70-10-INNHB
S. acidocaldarius	225	288	271	159	OX2-100-10-INNHB
S. cerevisiae	8	6.00	33	33	ALL-ALL-ALL-INNHB
X. laevis	132	158	113	100	OX2-100-10-INN

Bibliography

[1] Thomas Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, 1996.

[2] Amnon Barak, Oren La'adan, and Amnon Shiloh. Scalable cluster computing with mosix for linux. For the University of Jerusalem, 1999.

[3] Philip N. Borer, Barbara Dengler, Ignacio Tinoco Jr., and Olke C. Uhlenbeck. Stability of ribonucleic acid double-stranded helices. *Journal of Molecular Biology*, 86:843–853, 1974.

[4] Jamie J. Cannone, Sankar Subramanian, Murray N. Schnare, James R. Collett, Lisa M. D'Souza, Yushi Du, Brian Feng, Nan Lin, Lakshmi V. Madabusi, Kirsten M. Müller, Nupur Pande, Zhidi Shang, Nan Yu, and Robin R. Gutell. The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics*, 3, 2002.

[5] Erick Cantú-Paz. *Efficient and Accurate Parallel Genetic Algorithms.* Kluwer Academic Publishers, 2000.

[6] Erick Cantú-Paz. On random numbers and the performance of genetic algorithms. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages p. 311–318. Morgan Kaufmann Publishers, 2002.

[7] Thomans H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms.* MIT Press, second edition, 2001.

[8] Jennifer Couzin. Breakthrough of the year: Small RNAs make big splash. *Science*, 298(5602):2296–2297, 2002.

[9] Lawrence Davis. Job shop scheduling with genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pages 136–140. Lawrence Erlbaum Associates, Inc., 1985.

[10] Alain Deschênes. A genetic algorithm for RNA secondary structure prediction using stacking energy thermodynamic models. Master's thesis, Simon Fraser University, Burnaby, British Columbia, Canada, 2005.

[11] Alain Deschênes and Kay C. Wiese. Using stacking-energies (INN and INN-HB) for improving the accuracy of RNA secondary structure prediction with an evolutionary algorithm - a comparison to known structures. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, volume 1, pages 598–606, Portland, Oregon, Jun 2004. IEEE Press.

[12] Alain Deschênes, Kay C. Wiese, and Edward Glen. Comparison of permutation-based and binary representation in a genetic algorithm for RNA secondary structure prediction. In A. Y. Tawfik and S. D. Goodwin, editors, *Advances in Artificial Intelligence, 17th Conference of the Canadian Society for Computational Studies of Intelligence*, volume 3060 of *LNAI*, pages 549–550, London, Ontario, Canada, May 2004. Canadian AI, Springer.

[13] Alain Deschênes, Kay C. Wiese, and Jagdeep Poonian. Comparison of dynamic programming and evolutionary algorithms for RNA secondary structure prediction. In *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'04)*, pages 214–222. IEEE Press, Oct 2004.

[14] Karl Entacher, Andreas Uhl, and Stefan Wegenkittl. Parallel random number generation: long-range correlations among multiple processors. In P. Zinterhof, M. Vajtersic, and A. Uhl, editors, *Proceedings of the 4th International Conference of the ACPC (ACPC99)*, volume 1557 of *Lecture Notes in Computer Science*, pages 107–116. Springer-Verlag, 1999.

[15] Martin Fekete, Ivo L. Hofacker, and Peter F. Stadler. Prediction of RNA base pairing probabilities on massively parallel computers. *Journal of Computational Biology*, 7(1/2):171–182, 2000.

[16] George A. Fishman and III Louis R. Moore. An exhaustive analysis of multiplicative congruential random number generators with modulus 2^{31}-1. *SIAM J. Sci. Stat. Comput.*, 7(1):24–45, 1986.

[17] Michael J. Flynn. Very high speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909, 1966.

[18] Lawrence J. Fogel, Alvin J. Owens, and Michael J. Walsh. *Artificial Intelligence Through Simulated Evolution.* John Wiley & Sons, Inc., New York, 1966.

[19] Geoffrey C. Fox, Mark A. Johnson, Gregory A. Lyzenga, Steve W. Otto, John K. Salmon, and David W. Walker. *Solving Problems On Concurrent Processors, vol. 1 - General Techniques And Regular Problems.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[20] Susan M. Freier, Ryszard Kierzek, Marvin H. Caruthers, Thomas Neilson, and Douglas H. Turner. Free energy contributions of GU and other terminal mismatches to helix stability. *Biochemistry*, 25:3209–3213, 1986.

[21] Susan M. Freier, Ryszard Kierzek, John A. Jaeger, Naoki Sugimoto, Marvin H. Caruthers, Thomas Neilson, and Douglas H. Turner. Improved free-energy parameters for predictions of RNA duplex stability. *Proceedings of the National Academy of Sciences of the United States of America*, 83:9373–9377, 1986.

[22] Susan M. Freier, Naoki Sugimoto, Alison Sinclair, Dirk Alkema, THomas Neilson, and Ryszard Kierzek adn Marvin H. Caruthers adn Douglas H. Turner. Stability of xgcgcp, gcgcyp, and xgcgcyp helixes: An empirical estimate of the energetics of hydrogen bonds in nucleic acids. *Biochemistry*, 25:3214–3219, 1986.

[23] Paul P. Gardner and Robert Giegerich. A comprehensive comparison of comparative RNA structure prediction approaches. *BMC Bioinformatics*, 5(140), 2004.

[24] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms & Engineering Optimization.* John Wiley & Sons, Inc., New York, 2000.

[25] David E. Goldberg and Robert Lingle Jr. Alleles, loci and the travelling salesman problem. In J.J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 154–159. Lawrence Erlbaum Associates, 1985.

[26] Jan Gorodkin, Laurie J. Heyer, and Gary D. Stormo. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Research*, 25(18):3724–3732, 1997.

[27] Jan Gorodkin, Shawn L. Stricklin, and Gary D. Stormo. Discovering common stemloop motifs in unaligned RNA sequences. *Nucleic Acids Research*, 29(10):2135–2144, 2001.

[28] William Gropp, Ewing Lusk, Nathan Doss, and Anthony Skjellum. A high-performance, portable implementation of the mpi message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[29] Walter Gruner, Robert Giegerich, Dirk Strothmann, Christian Reidys, Jacqueline Weber, Ivo L. Hofacker, Peter F. Stadler, and Peter Schuster. Analysis of RNA sequence structure maps by exhaustive enumeration. Technical report, Santa Fe Institute, October 1995.

[30] Robin R. Gutell, Jung C. Lee, and Jamie J. Cannone. The accuracy of ribosomal RNA comparative structure models. *Current Opinion in Structural Biology*, 12:301–310, June 2002.

[31] Liyan He, Ryszard Kierzek, John SantaLucia Jr., Amy E. Walter, and Douglas H. Runer. Nearest-neighbor parameters for GU mismatches: GU/UG is destabilizing in

the contexts CGUG/GUGC, UGUA/AUGU but stabillizing in GGUC/CUGG. *Biochemistry*, 30:11124–11132, 1991.

[32] Andrew Hendriks, Alain Deschênes, and Kay C. Wiese. A parallel evolutionary algorithm for RNA secondary structure prediction using stacking-energies (INN and INN-HB). In *Proceedings of the 2004 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB'04)*, pages 223–230. IEEE Press, Oct 2004.

[33] Andrew Hendriks, Kay C. Wiese, Edward Glen, and Alain Deschênes. A distributed genetic algorithm for RNA secondary structure prediction. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003)*, pages 343–350. IEEE Press, Dec 2003.

[34] Matthias Höchsman, Thomas Töller, Robert Giegerich, and Stefan Kurtz. Local similarity in RNA secondary structure. In *Proceedings of IEEE Bioinformatics Conference 2003*, pages 159–168, 2003.

[35] Ivo L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Research*, 31(13):3429–3431, 2003.

[36] Ivo L. Hofacker, Stephan H. F. Bernhart, and Peter F. Stadler. Alignment of RNA base pairing probability matrices. *Bioinformatics*, 20(14):2222–2227, 2004.

[37] Ivo L. Hofacker, Martin Fekete, and Peter F. Stadler. Secondary structure prediction for aligned rna sequence. *Journal of Molecular Biology*, 319(5):1059–1066, 2002.

[38] Stephen R. Holbrook. Crystallographic analysis of RNA structure. In *RNA Structure and Function*, pages 147–174. Cold Spring Harbor Laboratory Press, 1998.

[39] Stephen R. Holbrook and Sung-Hou Kim. RNA crystallography. *Biopolymers*, 44(1):3–21, 1997.

[40] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, 1992.

[41] John A. Jaeger, Douglas H. Turner, and Michael Zuker. Improved predictions of secondary structures for RNA. *Biochemistry*, 86:7706–7710, October 1989.

[42] Kenneth De Jong. *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* PhD thesis, University of Michigan, Ann Arbor, MI, 1975.

[43] Bjarne Knudsen and Jotun Hein. Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Research*, 31(13):3423–3428, 2003.

[44] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, MA, USA, 1992.

[45] Derrick H. Lehmer. Mathematical methods in large-scale computing units. In *Proceedings of the 2nd Symposium on Large-Scale Digital Computing Machinery*, pages 141–146. Harvard University Press, 1951.

[46] Albert L. Lehninger, David L. Nelson, and Michael M. Cox. *Lehninger Principles of Biochemistry.* W.H. Freeman & Company, 4^{th} edition, 2004.

[47] Neocles Leontis and Eric Westhof. Geometric nomenclature and classification of RNA base pairs. *RNA*, 7(4):499–512, 2001.

[48] Linux. *RAND Manual Page*, 1995.

[49] Rune B. Lyngso and Christian N. S. Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology*, 7(3/4):409–427, 2000.

[50] Maumita Mandal and Ronald R. Breaker. Gene regulation by riboswitches. *Nature Reviews: Molecular Cell Biology*, 5(6):451–463, 2004.

[51] Michael Mascagni and Ashok Srinivasan. Algorithm 806: SPRNG: a scalable library for pseudorandom number generation. *ACM Transactions on Mathematical Software*, 26(3):436–461, September 2000.

[52] Michael Mascagni and Ashok Srinivasan. Parameterizing parallel multiplicative lagged-fibonacci generators. *Parallel Computing*, 30(5-6):899–916, 2004.

[53] David H. Mathews, Troy C. Andre, James Kim, Douglas H. Turner, and Michael Zuker. An updated recursive algorithm for RNA secondary structure prediction with improved free energy parameters. In N. B. Leontis and J. SantaLucia Jr., editors, *American Chemical Society*, 682, chapter 15, pages 246–257. American Chemical Society, Washington, DC, 1998.

[54] David H. Mathews, Matthew D. Disney, Jessica L. Childs, Susan J. Schroeder, Michael Zuker, and Douglas H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences, USA*, 101:7287–7292, 2004.

[55] David H. Mathews, Jeffrey Sabina, Michael Zuker, and Douglas H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288:911–940, 1999.

[56] David H. Mathews and Douglas H. Turner. Dynalign: An algorithm for finding the secondary structure common to two RNA sequences. *Journal of Molecular Biology*, 317:191–203, 2002.

[57] David H. Mathews, Douglas H. Turner, and Michael Zuker. RNA secondary structure prediction. *Current Protocols in Nucleic Acid Chemistry*, pages 11.2.1– 11.2.10, 2000.

[58] Makoto Matsumoto and Takuji Nishimura. Dynamic creation of pseudorandom number generators. In *Monte Carlo and Quasi-Monte Carlo Methods 1998*, pages 56–69. Springer, 1998.

[59] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[60] John S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6–7):1105–1119, 1990.

[61] Mark M. Meysenburg. The effect of the quality of pseudo-random number generators on the performance of a simple genetic algorithm. Master's thesis, University of Idaho, Moscow, Idaho, USA, 1997.

[62] Mark M. Meysenburg and James A. Foster. The quality of pseudo-random number generators and simple genetic algorithm performance. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 276–282, San Francisco, CA, 1997. Morgan Kaufmann.

[63] Mark M. Meysenburg and James A. Foster. Randomness and GA performance, revisited. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Fourth Genetic and Evolutionary Computation Conference*, volume 1, pages 425–432, San Francisco, CA, 1999. Morgan Kaufmann.

[64] Zbigniew Michalewicz. *Genetic algorithms + data structures = evolution programs (3rd ed.)*. Springer-Verlag, 1996.

[65] Uma Nagaswamy, Maia Larios-Sanz, James Hury, Shakaala Collins, Zhengdong Zhang, Qin Zhao, and George E. Fox. NCIR: a database of non-canonical interactions in known RNA stuctures. *Nucleic Acids Research*, 30(1):395–397, 2002.

[66] Ruth Nussinov, George Pieczenik, Jerrold R. Griggs, and Daniel J. Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied Mathematics*, 35:68–82, 1978.

[67] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms (ICGA-87)*, pages 224–230. Lawrence Erlbaum Associates, Inc., 1987.

[68] Cherri M. Pancake. Is parallelism for you? *IEEE Computational Science and Engineering*, 3(2):18–37, 1996.

[69] Elisabetta Viani Puglisi and Jospeh Daniel Puglisi. Nuclear magnetic resonance spectroscopy of RNA. In *RNA Structure and Function*, pages 117–146. Cold Spring Harbor Laboratory Press, 1998.

[70] Patricia J. Pukkila. Molecular biology: The central dogma. In *Encyclopedia of Life Sciences*. John Wiley and Sons, Ltd., 2000.

[71] Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, 1973.

[72] Jianhua Ruan, Gary D. Stormo, and Weixiong Zhang. An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. *Bioinformatics*, 20(1):58–66, 2004.

[73] Bart Rylander. *Computational Complexity and the Genetic Algorithm*. PhD thesis, University of Idaho, 2001.

[74] David Sankoff. Simultaneous solution of the RNA folding, alignment, and protosequence problems. *SIAM Journal on Applied Mathematics*, 45:810–825, 1985.

[75] Hans-Paul Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology. Wiley Interscience, New York, 1995.

[76] Martin J. Serra and Douglas H. Turner. Predicting thermodynamic properties of RNA. *Methods in Enzymology*, 259:242–261, 1995.

[77] B. A. Shapiro, J. C. Wu, D. Bengali, and M. J. Potts. The massively parallel genetic algorithm for RNA folding: Mimd implementation and population variation. *Bioinformatics*, 17:137–148, 2001.

[78] Bruce A. Shapiro and Joseph Navetta. A massively-parallel genetic algorithm for RNA secondary structure prediction. *Journal of Supercomputing*, 8:195–207, 1994.

[79] Sven Siebert and Rolf Backofen. MARNA: A server for multiple alignment of RNAs. In *Proceedings of the German Conference on Bioinformatics 2003*, pages 135–140, 2003.

[80] Bart Sinclair. How random is random()? URL: http://www.owlnet.rice.edu/ elec428/ rng/test.html, 2004.

[81] Ashok Srinivasan, David Ceperley, and Michael Mascagni. Random number generators for parallel applications. *Monte Carlo Methods in Chemical Physics*, 105:923–936, 1999.

[82] Ashok Srinivasan, Michael Mascagni, and David Ceperley. Testing parallel random number generators. *Parallel Computing*, 29(1):69–94, 2003.

[83] Timothy Starkweather, S. McDaniel, Keith E. Mathias, L. Darrell Whitley, and C. Whitley. A comparison of genetic sequencing operators. In Rick Belew and Lashon Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 69–76, San Mateo, CA, 1991. Morgan Kaufman.

[84] Naoki Sugimoto, Ryszard Kierzek, Susan M. Freier, and Douglas H. Turner. Energetics of internal GU mismatches in ribooligonucleotide helixes. *Biochemistry*, 25(19):5755 – 5759, 1986.

[85] Gilbert Syswerda. Handbook of genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, chapter Schedule optimization using genetic algorithms. Van Nostrand Reinhold, New York, 1991.

[86] Andrea Tettamanzi and Marco Tomassini. *Soft Computing: Integrating evolutionary, neural, and fuzzy systems.* Springer-Verlag, Berlin, 2001.

[87] Ignacio Tinoco Jr. and Carlos Bustamante. How RNA folds. *Journal of Molecular Biology*, 293:271–281, 1999.

[88] Igor I. Titov, Denis G. Vorobiev, Vladimir A. Ivanisenko, and Nikolay A. Kolchanov. A fast genetic algorithm for RNA secondary structure analysis. *Russ. Chem. Bull.*, 51:1135–1144, 2002.

[89] F. H. D. van Batenburg, Alexander P. Gultyaev, and Cornelis W. A. Pleij. An APL-programmed genetic algorithm for the prediction of RNA secondary structure. *Journal of Theoretical Biology*, 174:269–280, 1995.

[90] Gabriele Varani. RNA structure. In *Nature Encyclopedia of Life Sciences*, London, 2000. Nature Publishing Group. http://www.els.net.proxy.lib.sfu.ca/.

[91] Gabriele Varani, Fareed Aboul-ela, and Frederic H. T. Allain. NMR investigation of RNA structure. *Progress in Nuclear Magnetic Resonance Spectroscopy*, 29(1-2):51–127, June 1996.

[92] Martin Vingron. Algorithmische bioinformatik. C source code for Nussinov algorithm, 2003.

[93] James D. Watson, Tania A. Baker, Stephen P. Bell, Alexander Gann, Michael Levine, and Richard Losick. *Molecular Biology of the Gene.* The Benjamin/Cummings Publishing Co., Inc., San Franciso, California, fifth edition, 2004.

[94] Darrell Whitley, Timothy Starkweather, and Daniel Shaner. The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, pages 350–372. Van Nostrand Reinhold, New York, 1991.

[95] Kay C. Wiese, Alain Deschênes, and Edward Glen. Permutation based RNA secondary structure prediction via a genetic algorithm. In Ruhul Sarker, Robert Reynolds, Hussein Abbass, Kay Chen Tan, Bob McKay, Daryl Essam, and Tom Gedeon, editors, *Proceedings of the 2003 Congress on Evolutionary Computation (CEC2003)*, pages 335–342, Canberra, 8-12 December 2003. IEEE Press.

[96] Kay C. Wiese and Edward Glen. A permutation based genetic algorithm for RNA secondary structure prediction. In Ajith Abraham, Javier Ruiz del Solar, and Mario Koppen, editors, *Soft Computing Systems*, volume 87 of *Frontiers in Artificial Intelligence and Applications*, chapter 4, pages 173–182. IOS Press, Amsterdam, 2002.

[97] Kay C. Wiese and Edward Glen. A permutation-based genetic algorithm for the RNA folding problem: a critical look at selection strategies, crossover operators, and representation issues. *BioSystems - Special Issue on Computational Intelligence in Bioinformatics*, 72:29–41, 2003.

[98] Kay C. Wiese, Scott D. Goodwin, and Sivakumar Nagarajan. ASERC - a genetic sequencing operator for asymmetric permutation problems. In H. Hamilton and Q. Yang, editors, *Canadian AI 2000, LNAI 1822*, pages 201–213. Springer-Verlag Berlin Heidelberg, 2000.

[99] Barry Wilkinson and Michael Allen. *Parallel programming: Techniques and Applications using Networked Workstations and Parallel Computers.* Prentice Hall, 1999.

[100] Gregory V. Wilson. *Practical Parallel Programming.* MIT Press, Cambridge, Massachusetts, 1995.

[101] Carl R. Woese and Norman R. Pace. Probing RNA structure, function and history by comparative analysis. In R. F. Gesteland and J. F. Atkins, editors, *The RNA World.* Cold Spring Harbor, NY, 1993.

[102] Ming Wu, Jeffrey A. McDowell, and Douglas H. Turner. A periodic table of symmetric tandem mismatches in RNA. *Biochemistry*, 34:3204–3211, 1995.

[103] Stefan Wuchty. Suboptimal secondary structures of RNA. Master's thesis, University of Vienna, 1998.

[104] Stefan Wuchty, Walter Fontana, Ivo L. Hofacker, and Peter Schuster. Complete suboptimal folding of RNA and the stability of secondary structures. *Biopolymers*, 49:145–165, 1999.

[105] Tianbing Xia, Jr. John SantaLucia, Mark E. Burkard, Ryszard Kierzek, Susan J. Schroeder, Xiaoqi Jiao, Christopher Cox, and Douglas H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry*, 37:14719–14735, 1998.

[106] Tianbing Xia, Jeffrey A. McDowell, and Douglas H. Turner. Thermodynamics of nonsymmetric tandem mismatches adjacent to GC base pairs in RNA. *Biochemistry*, 36:12486–12497, 1997.

[107] Michael Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244:48–52, 1989.

[108] Michael Zuker. Prediction of RNA secondary structure by energy minimization. In Annette M. Griffin and Hugh G. Griffin, editors, *Computer Analysis of Sequence Data*, pages 267–294. Humana Press Inc., July 1994.

[109] Michael Zuker. Calculating nucleic acid secondary structure. *Current Opinion in Structural Biology*, 10:303–310, 2000.

[110] Michael Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Research*, 31(13):3406 – 3415, 2003.

[111] Michael Zuker, John A. Jaeger, and Douglas H. Turner. A comparison of optimal and suboptimal RNA secondary structures predicted by free energy minimization with structures determined by phylogenetic comparison. *Nucleic Acids Research*, 19(10):2707–2714, 1991.

[112] Michael Zuker, David H. Mathews, and Douglas H. Turner. Algorithms and thermodynamics for RNA secondary structure prediction: a practical guide. In J. Barciszewski and B.F.C. Clark, editors, *RNA Biochemistry and Biotechnology*, NATO ASI Series. Kluwer Academic Publishers, 1999.

[113] Michael Zuker and Patrick Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9:133–148, 1981.