RNA Secondary Structure Prediction using Ant Colony Optimisation

Neil McMillan

Master of Science

School of Informatics

University of Edinburgh

2006

Abstract

It is important to know the secondary structure of RNA for applications such as drug development and modelling single stranded viruses. Predictive methods have various degrees of accuracy but are significantly faster and cheaper than empirical methods such as X-ray crystallography. This project explores how Ant Colony Optimisation (ACO) performs on the task of RNA secondary structure prediction (RNASSP). An ant colony system is developed and experiments are conducted to examine its behaviour on this problem and to determine a good set of parameters. The performance and accuracy of this approach is then compared with alternative methods. The main findings are that whilst the accuracy of ACO is as good as dynamic programming for small sequences it is significantly slower to execute. For longer sequences both slower and less accurate than dynamic programming.

Acknowledgements

I would like to thank my supervisor, Dr. Gillian Hayes, for her help and advice throughout this project. I would also like to thank Tom McCallum for his suggestions on the areas to cover during experiments.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Neil McMillan*)

Table of Contents

Chapter 1

Introduction

RNA plays many important roles in nature such as protein synthesis and it is believed to have a role in viral infections such as HIV, the common cold and polio. It is relatively straightforward to determine the sequence of nucleotides comprising a segment of RNA but it is much more difficult to determine the 3D structure of the molecule once it has folded. Empirical approaches include X-Ray Crystallography and Nuclear Magnetic Resonance Spectroscopy but they are expensive and very time consuming compared to determining the sequence alone. Predictive methods include Genetic Algorithms, Dynamic Programming Algorithms and Kinetic Folding, each achieving results with varying degrees of accuracy and all significantly faster and cheaper than empirical methods.

This project explores how Ant Colony Optimisation (ACO) performs on the task of RNA secondary structure prediction (RNASSP). An ant colony system is developed and experiments are conducted to examine its behaviour on this problem and to determine a good set of parameters. The performance and accuracy of this approach is then compared with alternative methods. The main findings are that whilst the accuracy of ACO is as good as dynamic programming for small sequences it is significantly slower to execute and does not appear to offer as good a range of suboptimal solutions.

In chapter 2, the problem of RNA secondary structure prediction is defined and in chapter 3 ACO is introduced. Chapter 4 describes the proposed method of applying

ACO to the RNASSP problem. Chapter 5 presents and discusses experimental results, and finally chapter 6 is the conclusion.

Chapter 2

RNA Secondary Structure Prediction

This chapter describes the problem area, starting with an introduction to RNA (ribonucleic acid) and secondary structure. Empirical and predictive methods of determining RNA secondary structure are also described.

2.1 RNA

RNA is a biological molecule made from a sequence of nucleotides linked together by covalent bonds. There are four types of nucleotide in RNA: adenine (A), cytosine (C), guanine (G) and uracil (U). RNA plays several important roles in nature.

- During protein synthesis messenger RNA (mRNA) serves as a carrier of information from DNA. During transcription the genetic information in the DNA is copied into mRNA which is then translated into protein by a ribosome.
- The 3D structure of transfer RNA (tRNA) is crucial for transferring a specific amino acid to a growing protein. Each tRNA molecule binds to a particular type of amino acid, and a triplet of nucleotides (the anticodon) at the base of the molecule define when the amino acid is appended to the protein.
- RNA also has catalytic and structural roles in the cell and it is believed to have a role in viral infections such as HIV, polio and the common cold.

2.2 RNA Secondary Structure

Similarly to DNA, RNA can form stable double helix structures of complementary strands, however since RNA is usually single stranded it bends and folds back on itself to form hydrogen bonds between complementary nucleotides in close proximity (base pairs). The canonical base pairs in RNA are the Watson-Crick base pairs A–U and C–G (named after their discoverers), and the less stable 'wobble' pair G-U (so called because the bases bond in a skewed fashion). These base pairs play a significant role in determining the spatial structure and energy state of an RNA molecule. There are other base pairs that can form in practice but they are much less stable and are normally ignored in algorithms for predicting secondary structure. There are also other factors contributing to the structure of an RNA molecule that are ignored in the definition of secondary structure. These include base stacking [Petersheim et al., 1983] in which nitrogen bonds contribute stability to the structure, and the properties of aqueous solution affecting the dynamics of folding.

The number of degrees of freedom in folding RNA molecules is very high so the structural prediction problem is very hard to solve. Instead, it is possible to focus on an intermediate level representation of the folding. This secondary structure representation only indicates which base pairs are formed and ignores the final 3D shape of the molecule. A secondary structure can be thought of as a set of stems of various lengths (see Figure 2.1). In a stem there are several neighbouring base pairs, which contribute stacking energy to the structure.

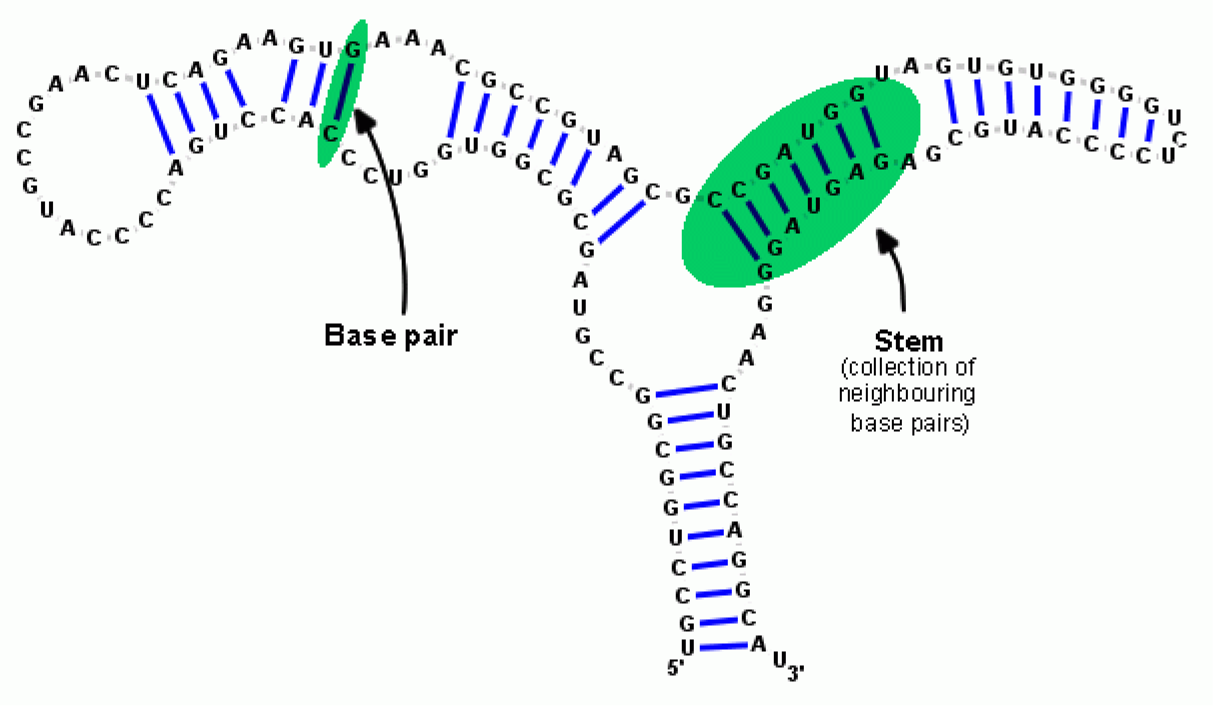

Figure 2.1: RNA secondary structure. This example is a ribosomal RNA fragment from Escherichia coli. Structure diagram generated by jViz.

Since secondary structure does not address define the three dimensional shape of the RNA molecule it is an incomplete description of the structure. Nevertheless it is useful to know secondary structure for several reasons:

1.Determining secondary structure is useful as a first step in predicting the full three-dimensional structures and the interpretation of the biological function of RNA molecules. This is because base pairing and stacking, which are accounted for in the definition of secondary structures, cover the major part of the free energy of the spatial structure.

2.Secondary structure alone can provide a lot of information about the molecule's structure and facilitates the identification of important sites. [Higgs, 2000]

3.RNA secondary structures are also useful when searching a genome for non-coding but functional forms of RNA. For example, microRNAs often have long stem-loop structures interrupted by small internal loops.

4.Secondary structure is sometimes more conserved than the sequence itself when comparing RNA molecules from different organisms with common ancestors. For example, some viruses belonging to the same family have little sequence similarity but have highly conserved secondary structure motifs. Also, many of the tRNA sequences with known structures have very similar clover-leaf patterns (see the image of the secondary structure in Figure 2.2).

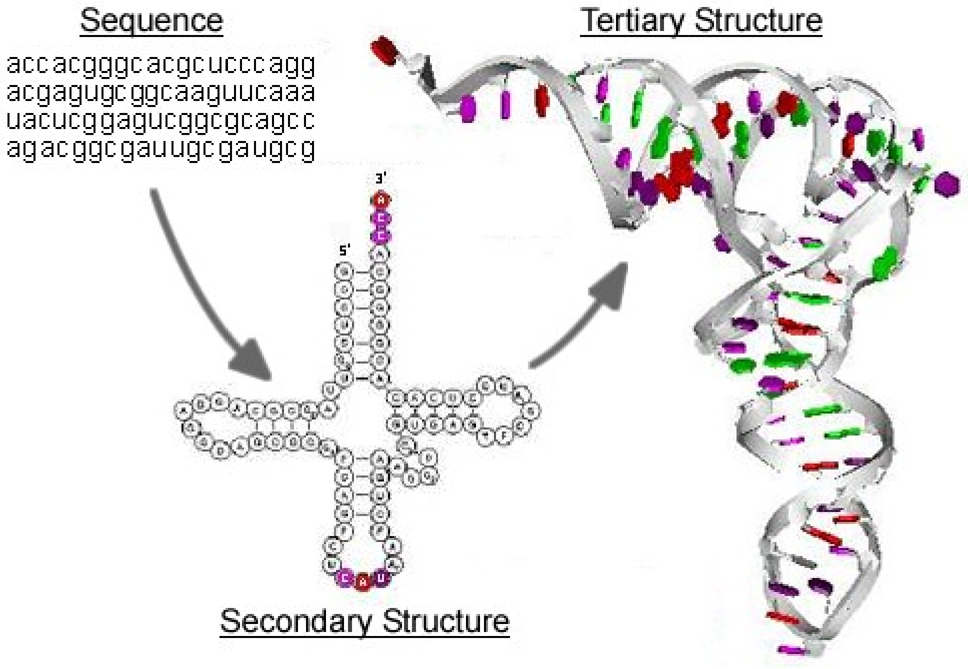

Figure 2.2: A tRNA sequence, shown with its secondary structure and spatial structure determined from X-ray crystallography. Image adapted from [Chemis, 2000]

Although secondary structure is defined as a set of base pairs present in the structure, the set must satisfy several constraints in order to form a valid secondary structure. If the nucleotides at positions *i* and *j* are complementary then a base pair may form as long as at least 3 nucleotides remain unpaired within a hairpin loop formed by these base pairs ($|j - i| \geq 4$). Considering another base pair *k-l*, it is compatible with the pair *i-j* if they are non-overlapping (e.g. $i < j < k < l$ or $k < l < i < j$) or if they are nested (e.g. $i < k < l < j$ or $k < i < j < l$). If these conditions are not met then the base pairs are overlapping (e.g. $i < k < j < l$) and a pseudoknot is formed. Pseudoknots are considered to be tertiary interactions. It is much more complicated to calculate the free energy of pseudoknotted structures so excluding them from the definition of secondary structures is a helpful simplification. Figure 2.3 shows a range of secondary structures that can be described purely by listing base pairs or stems.

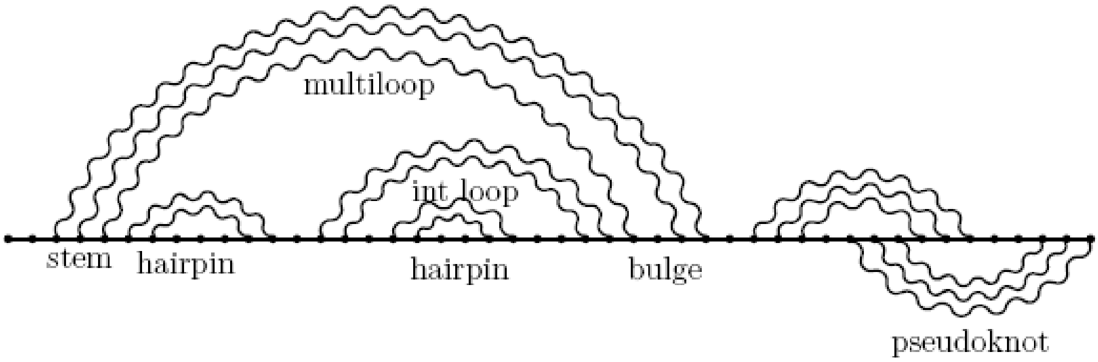

Figure 2.3: Diagrammatic representation of most secondary structures [Rivas and Eddy, 1999]

2.3 Determining RNA Secondary Structure

2.3.1 Empirical methods

It is relatively straightforward to determine the sequence of nucleotides comprising a segment of RNA but it is much more difficult to determine the 3D structure of the molecule once it has folded. Empirical approaches include X-Ray Crystallography [Holbrook, 1997] and Nuclear Magnetic Resonance Spectroscopy [Varani et al., 1996] but they are expensive and very time consuming compared to determining the sequence alone. Since the rate at which RNA is being sequenced is growing faster than the rate at which the structure can be determined predictive methods will continue to become increasingly important, especially as the reliability of such methods improves.

2.3.2 Predictive methods

Predicting the secondary structure of an RNA molecule from its sequence is a combinatorial optimisation problem. The number of possible structures increases exponentially with the length of the sequence so it is not feasible to search the entire space exhaustively. As with most combinatorial optimisation problems many different methods have been applied to this problem. I shall give a brief overview of these.

2.3.2.1 Dynamic Programming

The most thermodynamically stable structure of a molecule is the one with the minimum free energy (MFE). Since there are a finite number of possible secondary structures for a given sequence according to the restrictions described previously, it would be possible to enumerate them all and find the structure with the minimum free energy. Dynamic Programming Algorithms (DPAs) offer a shortcut to an exhaustive search by using a recurrence relation that breaks the problem into smaller parts. Nussinov & Jacobson [1980] proposed a 'maximum matching model' where the fitness function was based simply on the number of base pairs in the structure. The algorithm could accurately find the structure with the largest number of base pairs but this is usually quite different from the actual RNA structure.

To find the MFE structure the fitness function must take into account more chemically accurate energy parameters [Freier et al., 1986]. Some freely available software has been developed using these energy rules, such as mFold [Zuker, 1989] and the Vienna RNA secondary structure server [Hofacker, 2003].

The main weakness of many recursive-folding algorithms such as DPAs is that they produce a single optimal solution, but the natural RNA fold often has a sub-optimal energy state [Van Batenburg et al., 1995]. This could be for several reasons: the energy function failing to account for all factors in the calculation of the energy; the limitation of secondary structure as a representation of a real molecule; or the nature of the folding process directing the molecule's structure towards a locally, but not globally, optimal energy state. However it is possible to compensate for this limitation by producing a range of sub-optimal solutions, such as in Zuker's DPA [Zuker, 1989]. Some DPAs have also been able to deal with pseudo-knots [Akutsu, 2000].

2.3.2.2 Genetic Algorithms

Genetic algorithms (GA) are non-deterministic optimisation algorithms using concepts from biological evolution and they originated from the studies of cellular automata by John Holland [Holland, 1975]. The problem instances are represented as a chromosome (a binary sequence of values) and a fitness function and selection procedure ensure that good chromosomes are chosen and bad solutions rejected. Operations such as crossover and mutation generate new variants of the better chromosomes to form the next generation. GAs are good at combinatorial optimisation because a suitable fitness function will guide the population towards an optimal solution whilst avoiding a time-consuming exhaustive search.

Many types of GA have been tried, and [Shapiro et al., 2001] demonstrated that their GA could predict more correct base pairs and stems than would be possible with a DPA. In a similar vein [Benedetti and Morosetti, 1995] presented a GA that would produce optimal and sub-optimal RNA secondary structures. Even though the free energy values of these solutions are very similar the structures are quite varied, meaning that the range of solutions is more likely to match the actual biological folded structure than the optimal free energy solution alone.

GA approaches have many variations, some of which are conducive to optimising performance, for example the massively parallel GA by [Shapiro and Navetta, 1994]. Another variation is the permutation-based algorithm by [Wiese et al., 2005] which opened the way for cross fertilisation of ideas from the field of combinatorial problems.

2.3.2.3 Kinetic Folding

Dynamic programming and genetic algorithms using free energy as a fitness function both assume that the most likely structure is the one with the minimum free energy. However reality is unlikely to be so simple. The complexity of kinetics of folding could mean that real RNA molecules could fold up to the structure that forms most easily rather than the one with the minimum free energy.

One simple kinetic algorithm [Abrahams et al., 1990] repeatedly adds compatible stems to an existing structure. The stem that is added lowers the free energy of the structure by the largest amount. The algorithm uses published, experimentally determined free energy values and is able to produce structures containing pseudo-knots. Other kinetic algorithms [Mironov et al., 1985] used Monte-Carlo simulations to model the folding process of RNA molecules.

2.3.2.4 Comparative Methods

If sequences are available for a particular molecule across a range of related organisms then it is possible to predict the secondary structure using comparative methods. If the molecule has the same function in the various organisms then the structure must be the same even if the sequences vary slightly. This approach begins with a multiple alignment of all related sequences then searches for sites where there is a positive correlation between changes in nucleotide at two positions in the sequence.

Mutations can occur anywhere in the sequence with uniform probability. If an unpaired nucleotide mutates then it is not likely to affect the function of the molecule. However if a paired nucleotide mutates so that a base pair is no longer possible the structure could be disrupted. This would introduce an evolutionary drive towards a reverse mutation or a compensatory mutation. Reverse mutations could not be detected but with a sufficient number of related sequences for the same molecule compensatory mutations can reveal where base pairs are formed.

2.3.2.5 Stochastic Context Free Grammars

Stochastic context free grammars (SCFGs) are another probabilistic method of secondary structure prediction that can make use of the information gained from comparative sequence analysis. An advantage of this approach is that they can integrate information from other sources such as a biophysical model of structure plausibility based on energy minimisation and information extracted from databases. However it is not clear how best to combine the probabilistic evolutionary information with the energy minimisation model to produce a meaningful mathematical function [Dowell and Eddy, 2004]. Investigation into a range of simple SCFGs showed that they performed almost as well as energy minimisation methods such as Zuker's dynamic programming.

Chapter 3

Ant Colony Optimisation

Ant colony optimisation (ACO) is a population-based meta-heuristic for combinatorial optimisation problems. It is inspired by the ability of ants to find the shortest path between their nest and a source of food.

Marco Dorigo first introduced ACO in his PhD thesis [Dorigo, 1992] and applied it to the travelling salesman problem. It has since been applied to the quadratic assignment problem [Maniezzo et al., 1994], the vehicle routing problem [Bullnheimer et al., 1999], bin packing, stock cutting [Ducatelle and Levine, 2001] and 2D Hydrophobic-Polar protein folding [Shmygelska and Hoos, 2003]. Its application to protein folding demonstrates its potential in solving bioinformatics problems, but it is not known to have been applied to RNASSP.

This chapter gives an introduction to the inspiration for ACO, describes two varieties of the algorithm and discusses the role of local search and heuristics.

3.1 Biological Inspiration

ACO algorithms are based on the foraging behaviour of real ants. Many species of ant leave a pheromone trail in a quantity relative to the quality of the food source discovered. This influences the decision processes of other ants such that ants are more likely to choose a path with a stronger pheromone trail. Over time the behaviour of the ants converges to the benefit of the colony as a whole. This is known as *stigmergy*: a method of communication in emergent systems through modification of the environment.

An example of this behaviour was investigated by Deneubourg and his colleagues [Deneubourg et al., 1990]. They used a double bridge to connect a nest of Argentine ants with a food source and studied the pheromone trail-laying and trail-following behaviour of the ants. When the lengths of the bridges were adjusted such that one was longer than the other they found that in the early stages the distribution of the ants was mostly random, but in most experiments the ants all ended up using the shortest path.

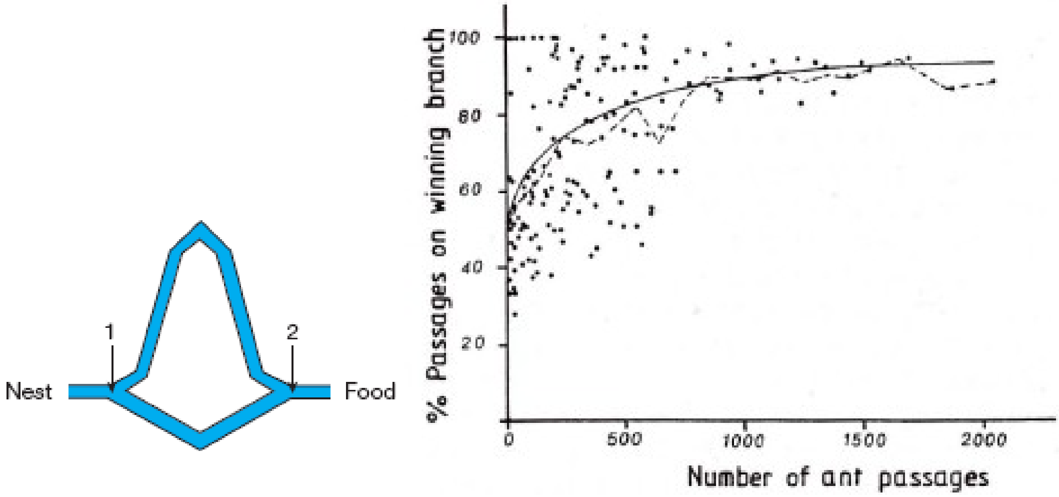

Figure 3.1: The double bridge experiment. Diagrams from [Bonabeau et al., 2000; Goss et al., 1990]

At the start of a trial there is no pheromone on the two bridges so the ants have no preference for which path to take. Discounting any other factors, 50% of the ants travel on each bridge. However the ants on the shorter path arrive first at the food and are ready to go back to the nest. This time there is higher pheromone on the shorter path which influences their choice of path. Over time the pheromone will build up faster on the shorter path and a positive feedback effect is produced such that in most cases all the ants travel on the shortest path.

The effect of this emergent behaviour in insects is not limited to finding the shortest path. For example, in termite nest building [Grassé, 1959] the termite workers collect balls of mud and use them to build pillars. They infuse each mud pellet with pheromone and initially deposit them randomly. When a pile of pellets reaches a certain size the level of pheromone stimulates the termites to deposit more pellets in the same area. This positive feedback effect leads to pillars being formed.

3.2 Algorithm

3.2.1 ACO for the Travelling Salesman Problem

The first ACO algorithm was ‘Ant System’ (AS), introduced by Dorigo in his PhD thesis [Dorigo, 1992]. It was applied to the travelling salesman problem (TSP), which is the problem of finding the shortest tour visiting all nodes in a fully-connected graph.

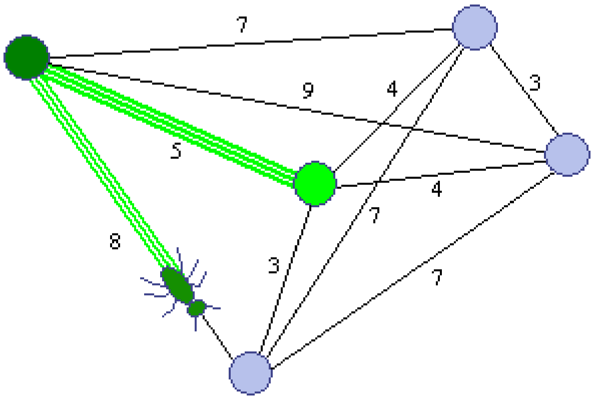

Figure 3.2: An example of an ant building a tour in the travelling salesman problem. The nodes represent cities and the numbers are the distances between cities. The central city is the randomly chosen starting point for this ant.

In the algorithm there are a number of ants (n) and a number of cities (m). Each ant starts in a random city and builds a complete tour by choosing each successive city probabilistically based on equation 3.1. At city *i*, ant *k* will choose city *j* with a probability given by:

$$p_k(i,j) = \frac{[\tau(i,j)]^{\alpha}.[\eta(i,j)]^{\beta}}{\sum\limits_{g \in N_i^k} [\tau(i,g)]^{\alpha}.[\eta(i,g)]^{\beta}} \tag{3.1}$$

In equation 3.1 $\tau(i,j)$ is the value of the pheromone between cities *i* and *j*, $\eta(i,j)$ is a heuristic indicating how good a choice city *j* would be from city *i*. Usually $\eta(i,j) = \frac{1}{d(i,j)}$, where *d(i, j)* is the distance between *i* and *j*. N_i^k is the allowed neighbourhood of ant *k* from city *i,* i.e. the cities that have not yet been visited. α and

β are parameters determining the relative weighting of the heuristic and pheromone contributions.

After all ants have built a complete tour, the lengths of the tours are calculated and the pheromone trail is updated based on the pheromone update rule (equation 3.2). In this equation $0 < \rho < 1$ is an evaporation constant. This ensures that the pheromone trails do not grow infinitely large and effectively allows solutions to be 'forgotten' if they are not reinforced often.

$$\tau(i,j) = \rho.\tau(i,j) + \Delta\tau_{ij} \tag{3.2}$$

The density of pheromone that is laid on edge (i, j) after each iteration is given by equations 3.3 and 3.4 where *Q* is a constant representing how much pheromone is deposited and L_k is the length of *k*'s tour. This is known as a local updating rule. However there are other variations of ACO such as the global updating rule in which only the iteration best ant lays any pheromone on the trail. To control more exactly the balance of exploration and exploitation of knowledge throughout the algorithm it is also possible to use a ε-greedy scheme in which ε of the time (e.g. 5%) a random city is chosen and the rest of the time the 'best' city is chosen according to $\max_j p(i,j)$.

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_k(i,j) \tag{3.3}$$

$$\Delta\tau_k(i,j) = \begin{cases} Q / L_k & \text{if ant k uses edge (i, j)} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

3.2.2 ACO for Bin Packing

Another NP-hard combinatorial optimisation problem to which ACO has been applied is the bin packing problem (BPP). The one-dimensional version of the task is to fit a set of items *S* each with weight w_i into a number of bins of fixed capacity *C*, whilst using the minimum number of bins (see Figure 3.3). There are many methods of solving this problem including heuristics such as fit first decreasing [Coffman et al., 1996], a reduction algorithm [Martello and Toth, 1990], genetic algorithms [Reeves, 1996] and ant colony optimisation [Ducatelle and Levine, 2001].

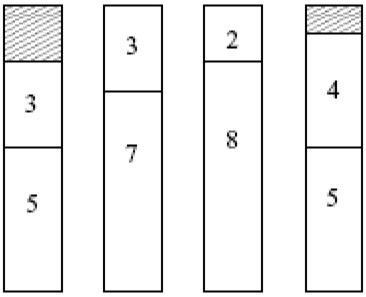

Figure 3.3: Example of a solution to a bin packing problem with *S* = {3, 5, 3, 7, 2, 8, 4, 5} and *C* = 10. Diagram from [Ducatelle and Levine, 2001].

The ACO solution to BPP uses a very similar equation to 3.1, but the important difference is in how the problem is represented, i.e. how the solutions are built up. Although it's entirely possible to represent solutions as permutations of items (making the problem equivalent to the TSP), in fact each ant *k* starts with an empty bin *b* and adds items *j* probabilistically to its solution *s* based on equation 3.4. It starts a new bin when no further items can be added to the current bin.

$$p_k(s,b,j) = \frac{[\tau_b(j)]\,.[\eta(j)]^\beta}{\sum\limits_{g\in J_k(s,b)}[\tau_b(g)]\,.[\eta(g)]^\beta} \quad \text{(if , 0 otherwise)} \tag{3.4}$$

The heuristic $\eta(j)$ is simply the size of item *j*, which means that larger items are 'preferred'. The pheromone value $\tau_b(j)$ for item *j* in bin *b* is the average of the pheromone values between *j* and all items already in *b*. The pheromone matrix is for item *sizes* rather than actual items so the pheromone values indicate how good two items of the given sizes are together. $J_k(s,b)$ is the set of items that are candidates for inclusion in bin *b,* i.e. they are small enough to fit into the remaining space. Again, β is a parameter controlling relative contributions from the heuristic and pheromone.

Another notable difference between the BPP and TSP algorithms is the pheromone update rule (equation 3.5). Only the best solution s_{best} for that iteration lays down any pheromone, and for that solution there is a pheromone update for every combination of item sizes *i* and *j* in the same bin.

$$\tau(i,j) = \rho.\tau(i,j) + m.f(s_{best}) \tag{3.5}$$

3.3 ACO with Local Search

Combining local search with ACO has been shown to product better results than ACO alone [Dorigo and Stützle, 2002; McCallum, 2005]. Local search algorithms start with a complete or partial initial solution and iteratively improve upon it. The neighbourhood of the current solution is searched and when a better solution is found it replaces the current solution. Local search continues until there are no more improvements to be found in the neighbourhood. This is known as a local optimum. The choice of neighbourhood function is of course crucial for producing a range of solutions that can be reached from the current solution in a single step, and therefore determines the performance of the algorithm.

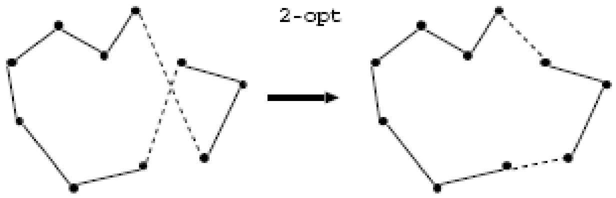

Figure 3.4: Schematic illustration of the 2-opt algorithm. The proposed move reduces the total tour length if we consider the Euclidean distance between the points. From [Dorigo and Stützle, 2001].

An example of local search for the TSP is *k-opt* in which *k* edges in a tour of all cities are cut and replaced with alternative edges which maintain a tour structure and lead to a shorter tour overall. Figure 3.4 shows an example of *2-opt*. For a given solution, *k-opt* is applied systematically to different combinations of *k* edges in the tour. Local search terminates when the solution can no longer be improved.

The largest problem with local search algorithms is that they get trapped in local optima. This means that there can be, and often is, a better solution somewhere else in the search space but it cannot be reached from the current solution because of the neighbourhood function and because the algorithm will not allow solutions to become temporarily worse in the hope that they will get better in the long term.

From the point of view of ACO, local search is good because it improves the solutions it comes up with as much as possible within the limitations of the neighbourhood function. And from the point of view of local search, ACO is good because it produces reasonable solutions to use as the starting point for local search and it makes use of the solution found at the end of one search (by updating the pheromone trail) instead of simply starting with a new random initial solution.

3.4 Heuristics

Providing information to the ants while they are building their solutions can help to direct the ants towards parts of the search space that are more likely to contain good solutions. For some problems, such as TSP, this information can be computed once during the initialisation of the algorithm, e.g. $\eta(i,j) = 1/d(i,j)$ (see section 3.2.1). But for other problems the heuristic depends on the partial solution generated by the ant so far, so it must be computed at run-time. This incurs higher computational cost and it is possible that performance could actually decrease with the use of dynamic heuristics. McCallum [2005] showed that heuristics do not make a significant difference to the quality of the solution if local search is applied.

Chapter 4

RNA Secondary Structure Prediction Using Ant Colony Optimisation

As with most combinatorial optimisation problems there are many approaches one can take to find a solution. Chapter 2 described several existing methods for RNA secondary structure prediction (RNASSP). Since ant colony optimisation (ACO) has been successfully applied to many combinatorial problems (chapter 3) and has not previously been applied to RNASSP there is sufficient reason to investigate its performance on this particular problem.

This chapter describes the methods used during this project in applying ACO to RNASSP.

4.1 Problem Representation

The majority of free energy minimisation approaches to RNASSP use stems as the components to form structures. The first step is therefore to determine the set of all possible stems from the sequence. A stem consists of at least three consecutive canonical base pairs and at least three nucleotides in the loop (see Figure 4.1a).

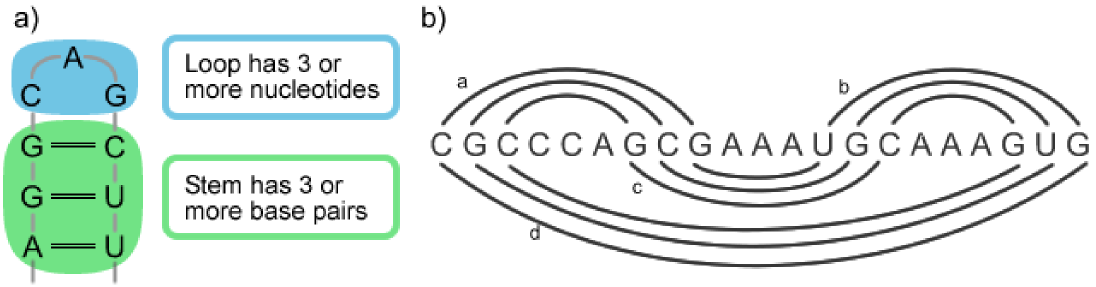
a)
A
C
G
G
C
G
U
A
U
Loop has 3 or more nucleotides
Stem has 3 or more base pairs
b)
a
b
C G C C C A G C G A A A U G C A A A G U G
c
d

Figure 4.1: a) The restrictions on possible stems for ACO b) A sample sequence with all possible stems shown.

Even a brute force algorithm can identify all possible stems quite quickly. Pseudo-code for such an algorithm is shown in Figure 4.2. When run on the sequence in Figure 4.1b, the algorithm would identify the set {*a*, *b*, *c*, *d*} as all the possible stems in the structure. For this sequence there are no other valid stems of length 3 or more according to the 3 canonical base pairings (see section 2.2). As can be seen from the diagram many of these stems are mutually exclusive e.g. having stem *a* in the structure precludes stems *c* and *d* because each nucleotide can only take part in at most one base pair. The set of all possible stems is referred to as the stem pool.

Figure 4.2: Pseudo-code for determining the set of all possible stems from a sequence of nucleotides. *S* is the minimum stem length (often 3), and *L* is the minimum length of a loop formed by a stem (also often 3). Note that the while loop is also doing bounds checking on k and exiting the loop if the value of k would causing indexing off either end of the sequence.

```
Stems ← Initialise to empty set
for $i$ = 0 …length of sequence do
 for $j = i + 2S + L – 1$ … length of sequence do
   Initialise $k$ to 0 (this is the counter for the stem size)
   while valid base pair between positions ($i$+$k$) and ($j$-$k$) do
    Increment $k$
    if $k >= S$ then
      Stems ← Insert stem ($i, j, k$)
    endif
   endw
  endfor
endfor
```

As described in chapter 2, a collection of stems can form structures such as hairpin loops, internal loops, multi-branch loops and bulges (see Figure 2.3). As will be shown in the next section it is also possible to represent pseudoknots with a collection of stems but because of the limitations of the free energy calculation function these structures are disallowed.

Having identified the stem pool it is necessary to define a suitable representation of the problem to describe how the ants can produce good combinations of stems that are not mutually exclusive. Two options for how the ants build candidate secondary structures are investigated in this project.

4.1.1 Direct structure construction (DSC)

One option for the ants to define the secondary structure is to directly construct a valid set of stems e.g. {c, d} or {a, b} according to Figure 4.1b. The ant would repeatedly add compatible stems probabilistically to the structure until no further additions are possible. The final solution contains no pseudo-knots and no mutually exclusive stems as defined in section 2.2.

Building a set of stems, unlike building a tour in TSP, does not depend on the last stem added to the structure, rather it depends on the set of stems formed so far. The probability of an ant *k* choosing stem *i* given that the partial solution generated so far is given by equation 4.1.

$$p_k(i) = \frac{[\tau(i)]^{\alpha}.[\eta(i)]^{\beta}}{\sum_{g \in N_k}[\tau(g)]^{\alpha}.[\eta(g)]^{\beta}} \quad \text{(if } i \in N_k\text{, 0 otherwise)} \qquad (4.1)$$

N_k (the allowed neighbourhood of stems given ant *k*’s partial solution) is defined to be the set of stems that meet all these conditions:

1.the stem is not already in *k*’s partial solution

2.the stem does not conflict with any stem already in *k*’s partial solution

3.the stem does not form a pseudo-knot when added to the structure

Figure 4.4 shows pseudo-code for checking all of these conditions. The rules for conflicts and pseudo-knots are explained in section 2.2 and Figure 4.3 shows examples of compatible and incompatible stems.

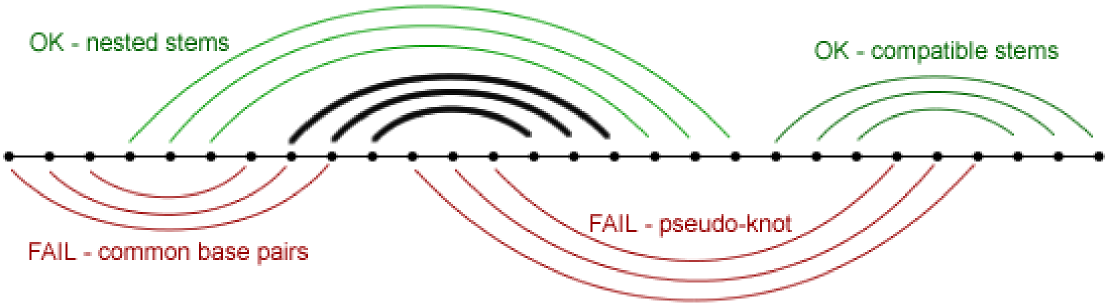

Figure 4.3: Diagram showing compatible and incompatible stems. The heavy black stem is the only stem in the test structure, the green stems are compatible with that structure, and the red stems are incompatible, either due to conflicting base pairs or the formation of pseudo-knots.

Figure 4.4: Pseudo-code for checking the compatibility of target stem *t* with the stems in the structure *k*. The extents of a stem means the lowest and highest indices of nucleotides forming base pairs in the stem.

```
for each stem $s$ in partial solution $k$ do
  if $s = t$ then
    return fail-duplicate
  else if any base pairs in common between $s$ and $t$ then
    return fail-conflict
  else if overlap between extents of stems $s$ and $t$ and
        $s$ not nested within $t$ and $t$ not nested within $s$ then
    return fail-pseudoknot
  endif
endfor
return success
```

4.1.2 Permutation-based construction (PBC)

The permutation-based genetic algorithm presented by Wiese et al. [2005] proposed an alternative to the existing binary representation of the secondary structure of RNA. Instead of a simple string of zeros or ones representing whether each corresponding stem is present in the structure, they used a permutation to encode this information. As well as conferring performance benefits, this approach showed that it was possible to represent RNASSP as a permutation problem.

Since it is not possible for all stems to be present in a secondary structure at one time, the order in which the stems appear in the permutation will define their priority when there are conflicts. Taking an example permutation {*c*, *d*, *a*, *b*} (refer to Figure 4.1b) this would result in stems *c* and *d* being included in the final structure because they are compatible stems, but stems *a* and *b* would not be included because they conflict with *c* and *d* which appear earlier in the permutation.

Building a permutation of stems is equivalent to creating a tour of cities so the ACO algorithm is very similar to the TSP algorithm described in section 3.2.1. The probability of an ant *k* choosing stem *j* when the last stem it added to the permutation was *i* is given by equation 4.1. This is the same as the TSP equation 3.1 but is reproduced here for convenience.

$$p_k(i,j) = \frac{[\tau(i,j)]^\alpha . [\eta(i,j)]^\beta}{\sum\limits_{g \in N_i^k} [\tau(i,g)]^\alpha . [\eta(i,g)]^\beta} \qquad (4.2)$$

Essentially this approach is similar to the DSC method above except that is divided into two stages. In the first stage ACO generates a permutation of all stems in the stem pool. The compatibility of stems is completely ignored and the only objective is to define an ordering of all stems. The second stage takes the permutation ordering and attempts to add the stems to the structure one at a time. If there is a conflict according to the same rules described in section 4.1.1 then the candidate stem is simply not added to the structure. In this way the permutation is reduced into a set of mutually compatible stems.

There are various options available for heuristic and pheromone components of this equation, and these are discussed in later sections of this chapter.

4.2 Pheromone trail definition

There were two main approaches to defining the pheromone trail for this problem. The simplest approach is to have a one-dimensional pheromone matrix with a value for every possible stem in the structure. Using this approach updating the pheromone trail would entail laying pheromone for every stem that appears in the structure generated by the ant. In PBC it would be pointless to lay pheromone for every stem in the permutation because that would include every stem in the stem pool.

Another approach is to have a two-dimensional matrix that takes into account combinations of stems in structures. In PBC every consecutive pair of stems would have pheromone applied. In DSC every pair-wise combination of stems in the structure would have pheromone applied.

Preliminary investigation indicated that the two-dimensional pheromone matrix was both costly in memory allocation and no better in terms of results. There was insufficient time for a thorough comparison of these approaches. All experiments in this dissertation used the one-dimensional pheromone trail.

4.3 Pheromone update rule

In addition to defining the dimensionality of the pheromone matrix, the method of updating the matrix is also important. Chapter 3 covers two types of update rule: the local updating rule used in the TSP example and the iteration-best update rule used in the BPP example. There is also a global update rule in which only the ant with the globally best solution gets to deposit pheromone at the end of each iteration. These three pheromone update rules are explored in section 5.1.3.

4.4 Free Energy Fitness Function

Like Dynamic Programming and Genetic Algorithms the premise behind this method of secondary structure prediction is that the most likely biological structures have the lowest free energy. Therefore a method of calculating the free energy of the secondary structure is required. The implementation of this function is outside the scope of this project because it is a rather complex calculation involving thermodynamic parameters from empirical measurements. It has been implemented several times already in existing open-source secondary structure prediction software packages. Free energy calculation functions were extracted from two such packages: the Vienna RNA Package [Hofacker, 2006] and the MultiRNAFold package [Andronescu et al., 2005]. These were made into static libraries for the ACO program. The majority of the experiments used the Vienna free energy function as it was more stable and could produce energy values for a larger range of structures.

Free energy of molecular structures is measured in kilocalories per mole (kcal/mol) and is usually negative because energy would be required (e.g. by heating) in order to break the chemical bonds. The fitness function used in ACO experiments is simply the negative of the free energy in the secondary structure according to the thermodynamic properties of interactions between nucleotides [Freier et al., 1986]. Fitness evaluation is usually a bottleneck in evolutionary computation, however making use of the Vienna free energy calculation the fitness function takes only 0.6ms to execute on a sequence of over 3000 nucleotides running on the platform described in section 5.1.1.

4.5 Empirically Determined RNA Secondary Structures

In order to establish the accuracy of the predictions produced by ACO it is necessary to use a variety of empirically determined RNA secondary structures. The Gutell Lab Comparative RNA Website [Cannone et al., 2002] has a large database of RNA sequences with known structures, mostly determined by X-ray crystallography. A selection of structures from this database is used for evaluation of the algorithms.

4.6 Heuristics

As described in section 3.4 heuristics are an important part of ACO. The role of heuristics on this problem is explored by using a variety of different heuristics and values for β, the parameter controlling the contribution from the heuristics (see equation 4.1). Some simple heuristics that are tested are based on the free energy of each stem in the stem pool. In the one-dimensional case $\eta(i) = (-energy_s(i))^\beta$, where $energy_s(i)$ is the free energy of a molecule with sequence *s* containing only the stem *i*. In the two-dimensional case, pair-wise combinations of stems are considered such that $\eta(i, j) = (-energy_s(i, j))^\beta$. Using the energy of the stems in isolation or combination for the heuristic would appear to be a sensible approach because for all valid structures in which the stem(s) could find themselves, the heuristic will be making a good approximation of the contribution of the stem(s) to the overall energy of the structure. Thermodynamic parameters particular to individual structures may be ignored by the heuristic but at least the lowest common denominator of energy contributions is covered.

One finding in Tom McCallum's PhD thesis was that ACO algorithms do not handle misleading heuristics well [p. 269, McCallum, 2005]. The accuracy of the heuristic limits the ability of the algorithm to find good solutions. To determine how important heuristics are for the RNASSP problem, and to help evaluate the quality of the one- and two-dimensional heuristics, this dissertation investigates misleading and random heuristics (section 5.1.4).

4.7 Local search

Since it has been shown that adding local search to ACO can improve upon the results [Dorigo and Stützle, 2001] a variety of local search approaches are investigated, compared and contrasted. The local search method will be determined to some extent by the problem representation. If using the permutation based representation it would be possible to make use of a variety of TSP heuristics such as k-opt and Lin-Kernighan [Lin and Kernighan, 1973] however to make the local search methods usable across different problem representations, only the final set of stems in the structure will be considered as the starting point for local search.

Considering a set of stems in a structure, the easiest way to reduce its free energy is to add more compatible stems. Any additional base pairs that can be added to the structure will help to make the free energy lower. However, depending on the initial structure, it may not be possible to add any further stems. In this case it may be worth removing one or more stems and replacing it with an alternative from the stem pool. If the replacement reduces the free energy then it can be kept. There are many different policies for this strategy of local search covering factors such as: how many stems to replace, how many repetitions of the local search procedure to do, whether to stop after the first improvement is found, wait until N improvements have been found or go through all possibilities and choose the best.

A selection of local strategies are explored:

- Add – simply try to add any compatible stem from the stem pool if it lowers the free energy
- Replace – remove each stem from the structure in turn. Try to replace it with an alternative from the stem pool. Stop if a replacement is found that lowers the free energy.
- Best of 3 Improvements – as Replace but pick the best of 3 improvements found.
- Best Replace – do the complete search over all possible replacements of all stems in the structure. Choose the best replacement.

4.8 Summary

This chapter has described the overall approach of using ant colony optimisation on the RNA secondary structure prediction problem. Each of the algorithms parameters described here will be experimentally evaluated in the next chapter (section 5.1).

Chapter 5

Experimental Results

This chapter summarises the results obtained from experiments with ACO for RNASSP. Firstly, various parameters and options for ACO are examined. The aim is to understand the behaviour of the algorithm and to determine the most suitable set of parameters for this problem. Secondly the most successful ACO algorithm is compared with some alternative state-of-the-art approaches.

5.1 Investigation into ACO parameters

5.1.1 Experimental set-up

The platform on which all experiments were run was Intel Pentium 4 with hyper-threading, 3.0 Ghz, running Windows 2000 SP 4. All algorithms and programs were written in C++ and compiled with Microsoft Visual Studio 6.0. Additionally, several Python scripts were written to perform tasks such as running batches of experiments, organising experimental data, converting data files and analysing results. All code was written by the author of this dissertation except the free energy calculation source code that was taken from the Vienna RNA package [Hofacker, 2003] and the

MultiRNAFold package [Andronescu et al., 2005]. All free energy values that are listed in this report are calculated using the Vienna RNA package.

The ACO algorithm was implemented according to the description in chapter 4. Unless otherwise stated all experiments are run using the following parameters:

- The evaporation constant ρ is 0.999. *
- The pheromone matrix is initialised with random values between 1 and 5. *
- The pheromone deposit is 10. *
- The default pheromone update rule is 'iteration-best': in each iteration only the ant that generates the best solution deposits any pheromone.
- α, controlling the contribution of the pheromone, is set to 1. *
- β, controlling the contribution of the heuristic, is set to 1.
- The minimum loop size is 3. *
- The normal definition of a stem requires 3 or more paired nucleotides but many of the experiments in this report use a minimum stem size (M) of 4. This is to reduce the size of the stem pool, making experiments much faster to run.
- Although equations 4.1 and 4.2 in section 4.1 define the probability distribution for adding stems to a partial solution they do not describe initialisation of the solution. When building a candidate solution the first stem is chosen using a uniform-random distribution.
- The default heuristic is one-dimensional and based on the free energy of each stem in isolation (see section 4.6).
- No local search is applied by default.
- The termination condition for the ACO algorithm is usually if there is no improvement in the global best solution after 50 iterations. However some experiments use a fixed number of iterations.

Parameters marked with * are constant in all experiments, other parameters are varied in some experiments.

Sequences used are all ribosomal RNA from different organisms. Further information about the sequences can be found by looking up the accession number in the Gutell Lab Comparative RNA Website [Cannone et al., 2002]. Sequences used are: Escherichia Coli (length 120, accession number V00336); Rhodobacter Capsulatus (length 123, accession number: X04585); Saccharomyces Cerevisiae (length 115, accession number: X54252); Haliotis Rubra (length 543, accession number: b.I1.e.H.rubra.1.C1.SSU.1506); Tetrahymena Thermophila (length 506, accession number: V01416); Caenorhabditis Elegans (length 697, accession number: X54252).

5.1.2 Number of ants

With the ant colony optimisation algorithm a natural place to begin investigation is to vary the number of ants to determine if the population-based nature of the algorithm is significant. Figure 5.1 shows how changing the number of ants whilst keeping all other variables the same affects the quality of the solution. Figure 5.2 shows similar results for an alternative sequence. The number of ants was varied from 1 up to 216 while the number of iterations varied from 1080 down to 5 to compensate for the number of ants. Therefore the product of the number of ants and number of iterations remained constant throughout all trials. The number 1080 was chosen because it is divisible by a good range of whole numbers. Each point on the graphs is the mean of 50 repetitions of ACO with the same parameters and the vertical error bars represent the standard error of the mean.

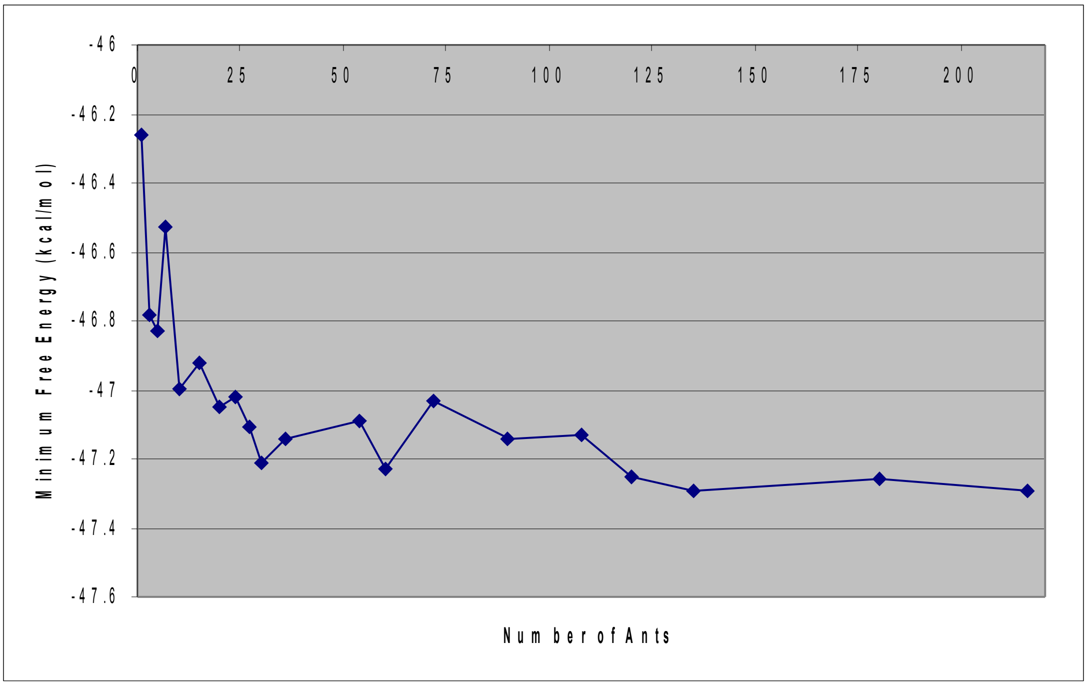

Figure 5.1: Varying the number of ants from 1 up to 216. Sequence: E. Coli. Problem representation: PBC. Pheromone update rule: iteration-best.

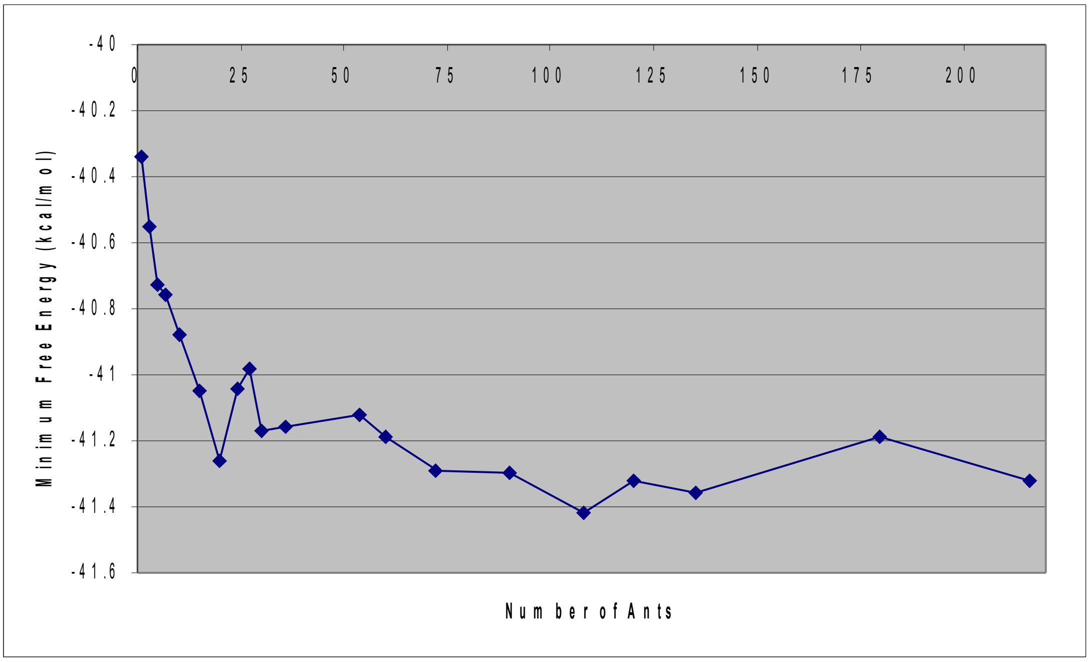

Figure 5.2: As Figure 5.1 except the sequence tested was R. capsulatus.

From Figure 5.1 and Figure 5.2 it can be seen that initially, as the number of ants increases from 1 to around 25 or 30, the average quality of the solutions improves. However, beyond this point the quality seems to level off and there is no longer any significant difference in the solutions. It seems clear that the number of ants is related to the quality of the solutions produced by ACO. Since these results are quite typical for four different sequences tested it would be reasonable to set the number of ants to around 30 for future experiments to increase the chance of getting good solutions.

When running the ACO algorithm with a low number of ants there would not be many candidate solutions at the end of the iteration. This means that with an iteration-best pheromone update policy the solution that has caused pheromone to be laid may not be particularly good. As a result the newly added pheromone could negatively contribute towards solutions produced in subsequent iterations. This could explain why more ants are, up to a point, better than few ants: essentially it increases the likelihood that the ant laying the pheromone trail actually produced a good solution.

If it were indeed the case that the iteration-best pheromone update is causing the observed difference in performance when varying the number of ants, then we would expect that using an update policy that is neutral to the number of ants would confirm this hypothesis. Such a policy is the local pheromone update rule. In this case there is no elitism and every ant, no matter how many run in parallel per iteration, will contribute to the pheromone trail.

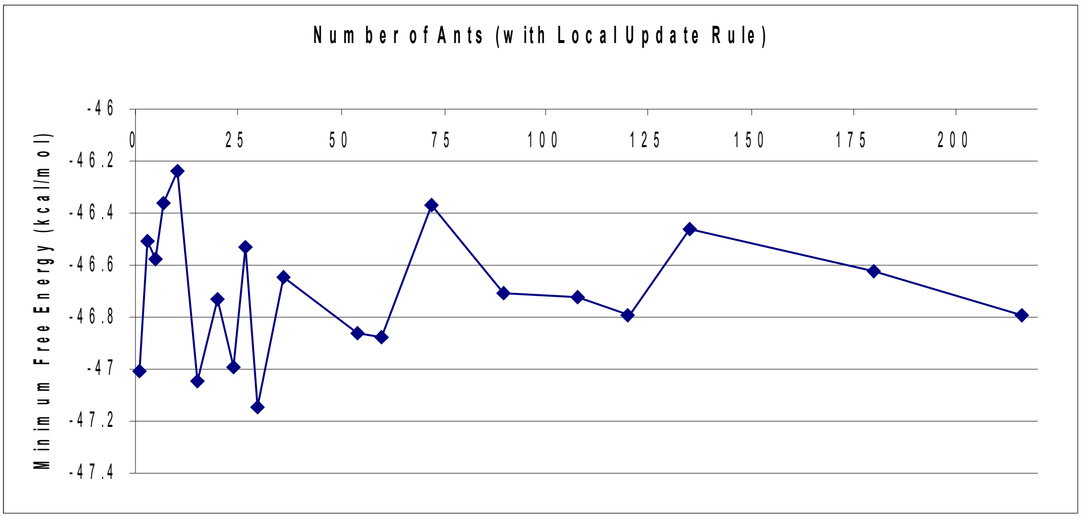

Figure 5.3: As 5.1 except that the local pheromone update rule was used.

Figure 5.3 shows the results of an experiment using local pheromone update rule. This means that every ant in every iteration contributes to the pheromone trail. 20 repetitions were performed for each point on the graph and again, the vertical error bars correspond to the standard. It can be seen from the graph that the relationship between the quality of the solutions and the number of ants no longer holds.

To summarise the results of this section, the number of ants has a definite effect on the quality of the solutions but this effect is also dependent on the choice of pheromone update rule. Furthermore, when using the iteration-best update rule a good choice for the number of ants is 30.

5.1.3 Pheromone update rule

The result of the experiments varying the number of ants brings up the question of how the pheromone update rule interacts with the number of ants, and what effect the pheromone matrix has on the solutions in general.

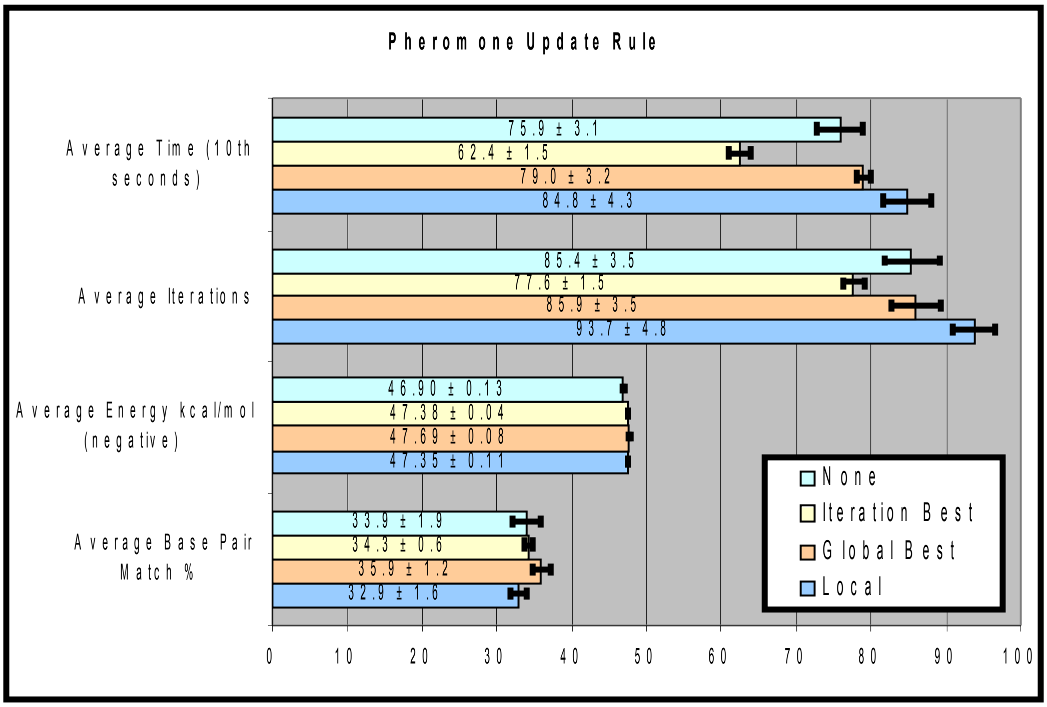

Figure 5.4: Comparison of 4 different pheromone update rules. Sequence: E. Coli. Problem representation: PBC.

Figure 5.4 shows a comparison of 4 different pheromone update rules. The values shown are averaged over 50 executions of each problem representation and the horizontal error bars represent the standard error of the mean. 'Base Pair Match %' means the percentage of base pairs in the best ACO structure that are in common with the stems in the actual RNA structure for the given sequence. The problem representation is permutation-based construction and based on the results of section 5.1.2 the number of ants was set to 30. Note that free energy values are negative (e.g. –46.9 kcal/mol) but for convenience they are shown as positive on the chart. The scale for the x-axis is different for each of the 4 measurements.

When pheromone update is disabled the algorithm takes longer to terminate, requires more iterations and produces lower quality solutions (in terms of minimum free energy) than when some sort of pheromone update occurs. This is to be expected because the pheromone matrix will contain useful information about which stems (or combination of stems) have led to good solutions in the past. However the margin of difference in performance when pheromone update is on and off is not very large – around 0.3 kcal/mol. This would suggest that the pheromone trail is not making a significant contribution to the progress of the ACO algorithm.

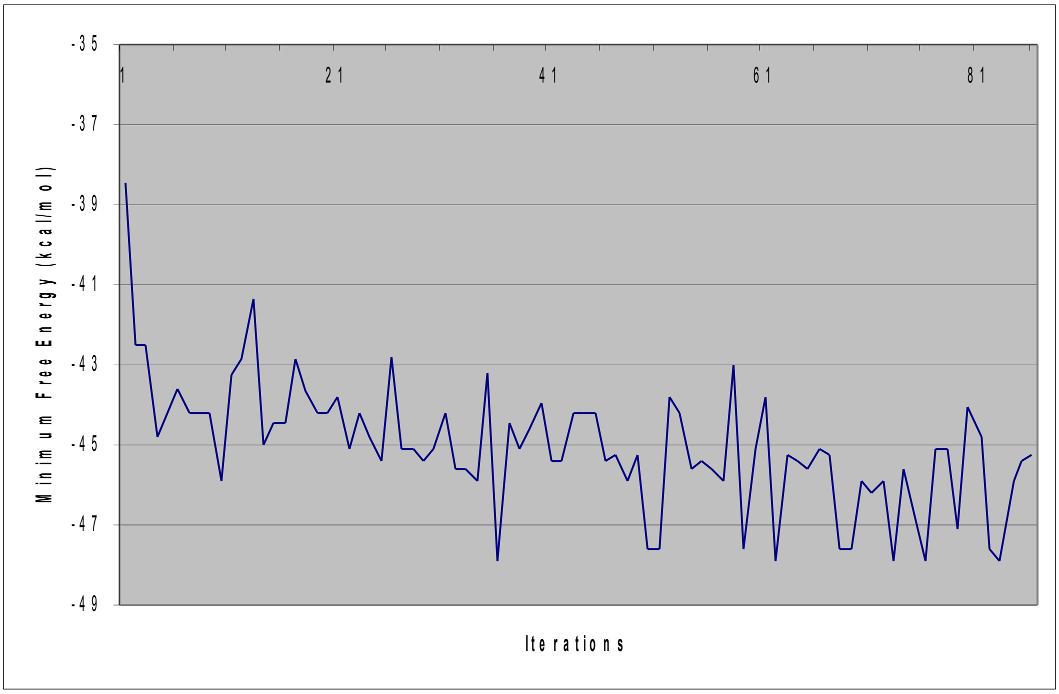

Figure 5.5: This graph shows the free energy of the best solution in each iteration in a typical run of the ACO algorithm using 30 ants and the iteration-best pheromone update rule. The experiment parameters are the same as in figure 5.4.

In ACO the pheromone trail is a method of storing knowledge and it should allow for ‘learning’ i.e. higher quality solutions at the end compared to the beginning. Figure 5.5 shows the progress of the ACO algorithm on a typical run and reveals how the quality of the iteration-best varies from beginning to end. There is some noticeable learning at the start of the algorithm during the first 5 iterations. Then for the next 30 iterations there is a lot of variation in quality of solution from around –41 to –46 kcal/mol. At around 35 iterations the algorithm happens upon a significantly better solution at –47.90 kcal/mol. During the last 50 iterations of the experiment several other equally good solutions are found. There is no improvement on the earlier result but the frequency of these good solutions is higher than before, presumably because it has again ‘learned’ from the discovery of the first good solution. After 50 iterations of no improvement in terms of minimum free energy the algorithm terminates and returns a global best solution.

Although it is clear that the pheromone trail is enabling learning, experiments show that with pheromone update disabled the quality of solutions is only slightly lower. To explore this further it is necessary to look for other factors contributing to producing good solutions even when pheromone update is disabled. The obvious candidates are the heuristic and problem representation.

5.1.4 Heuristic Types

To determine why good solutions are still produced even when pheromone update is disabled a variety of heuristics were investigated. Figure 5.6 and Figure 5.7 show the results of experiments using four different types of heuristic plus an experiment without any contribution from heuristics at all. Section 4.6 describes the one- and two-dimensional heuristics. ‘Random’ is a static one-dimensional heuristic with random values for each stem. ‘Deceptive’ is based on the one-dimensional heuristic but is inverted so that the stems with the lowest energy values have the lowest heuristic values (recall that the objective is to *minimise* free energy). To get a representative range of results, the sequence and problem representation is different in the two graphs.

Figure 5.6 and Figure 5.7 shows that heuristics do not make a significant difference to the quality of the solutions produced by ACO. In terms of minimum free energy there was little difference between a one-dimensional heuristic, a two-dimensional heuristic and no heuristic at all. By contrast the random and deceptive heuristics on average produced solutions with higher (less negative) free energy. Overall, the quality of solutions was affected by the choice of heuristic but not by a large amount, considering that the difference between the best and worst performances was at most 1 kcal/mol.

Varying the heuristics has a more significant effect on the number of iterations and time taken to execute the algorithm. The simple one-dimensional heuristic produced the best performance in terms of number of iterations and run-time.

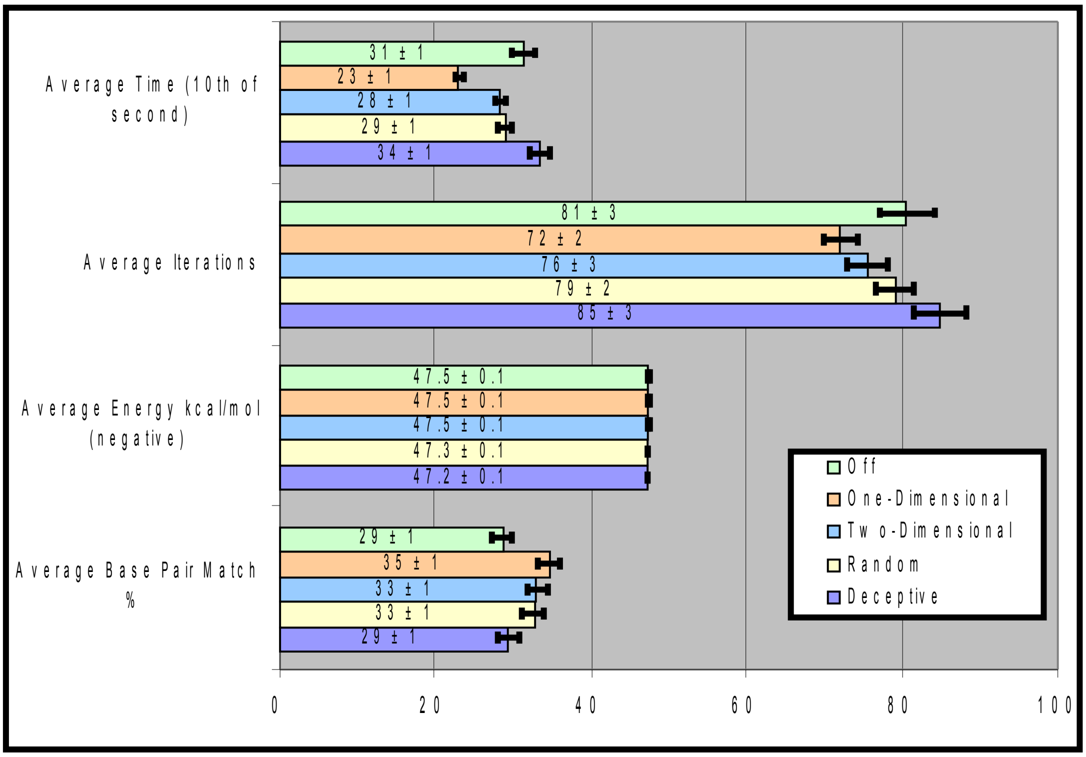

Figure 5.6: Comparison of various heuristics. Sequence: E. Coli. Problem representation: DSC.

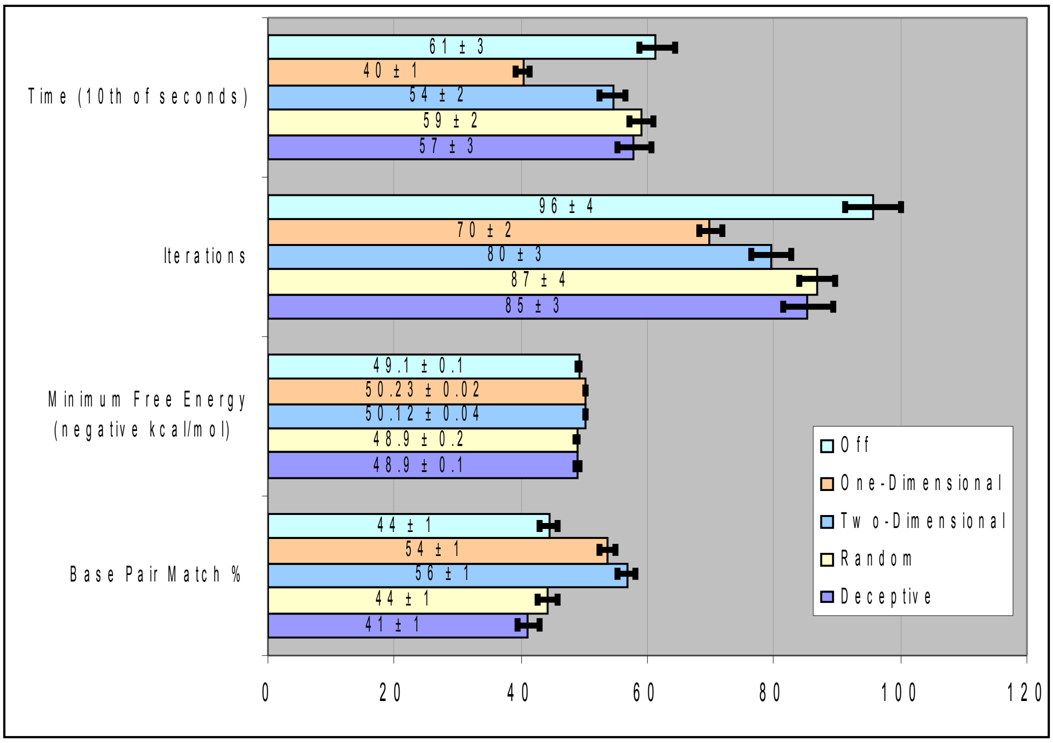

Figure 5.7: Comparison of various heuristics. Sequence: S. Cerevisiae. Problem representation: PBC.

So far experiments have shown that both the pheromone update rule and choice of heuristic make only slight difference to the quality of solution produced by ACO (in terms of minimum free energy), although they both have a noticeable effect on the run-time of the algorithm.

5.1.5 Problem representation

The problem representation is another possible reason why the pheromone trail does not appear to affect the quality of the solutions much. Two alternative methods of constructing solutions are explored: permutation-based construction (PBC) and direct structure construction (DSC) (explained in section 4.1). Figure 5.8 shows results averaged over 50 repetitions of each problem representation. From the chart it can be seen that DSC superior to PBC in terms of execution time and possibly on percentage base pair match, but in other respects the two approaches are very similar. This experiment was repeated for two other sequences (S. Cerevisiae and R. Capsulatus) with very similar results.

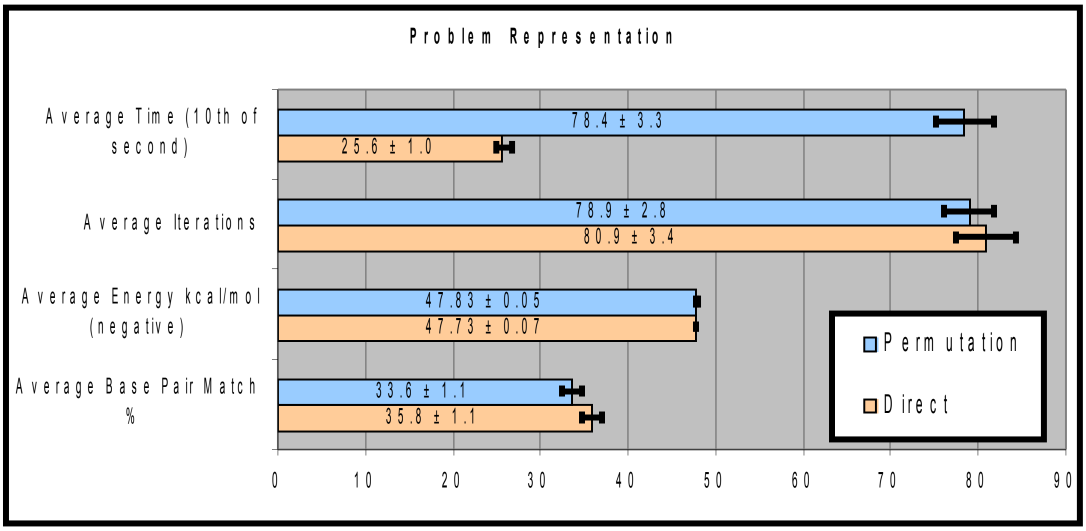

Figure 5.8: Comparison of two different problem representations. Sequence: E. Coli.

The permutation-based approach is equivalent to the direct approach once all of the conflicting stems have been removed from the permutation and both approaches can produce the same range of solutions. However there is unnecessary redundancy in the range of solutions that can represented by the permutation-level description of structures i.e. many different permutations of stems result in the same set of stems once it has been converted into a structure. This has the following effects:

- In cases where a two-dimensional pheromone matrix is used, the pheromone associated with neighbouring stems in a permutation does not contribute much relevant information to the building of the solution. This is because it is unlikely (except at the beginning of building permutation when the likelihood of stem conflicts is lower) that two neighbouring stems will make it into the final structure.
- As can be seen from the difference in time taken to complete an ACO run, there is significantly more computation required to build permutations than to simply build the structures directly. This is because every stem in the stem pool needs to be added probabilistically to the permutation, whereas in the direct method stems are only added to the solution until there are no further compatible stems.

Although both approaches produce similar quality solutions, direct structure construction has clear performance benefits so it is used for all subsequent experiments.

5.1.6 Summary

Based on the investigation so far into the problem representation, heuristic type and pheromone update method, an informal explanation for why reasonable solutions are being produced even when pheromone update is disabled could be as follows. Both the problem representations are designed to produce valid structures with as many stems as possible. Already, this rules out a whole range of invalid and unlikely structures, reducing the size of the search space. Taking into account ACO generating a large number of candidate solutions and keeping track of the best so far, this makes the algorithm quite likely to happen upon a reasonable solution. Furthermore, the energy-based heuristic makes it more likely that good stems are chosen by emphasising those stems contributing most to the free energy. Also, the pheromone trail can contribute useful information for directing the construction of solution in later iterations. Altogether, these various aspects complement each other in forming reasonable solutions.

The rest of this section continues to explore ACO parameters in order to understand how ACO performs on the RNASSP problem and to determine a reasonable set of algorithm parameters.

5.1.7 Heuristic Beta

Equation 4.1 in section 4.1.1 shows the probability function for DSC. Two parameters, α and β, control the contributions of the pheromone trail and heuristic respectively. So far, both parameters have been fixed at 1.0 but varying β will reveal more information about the effects of the heuristic.

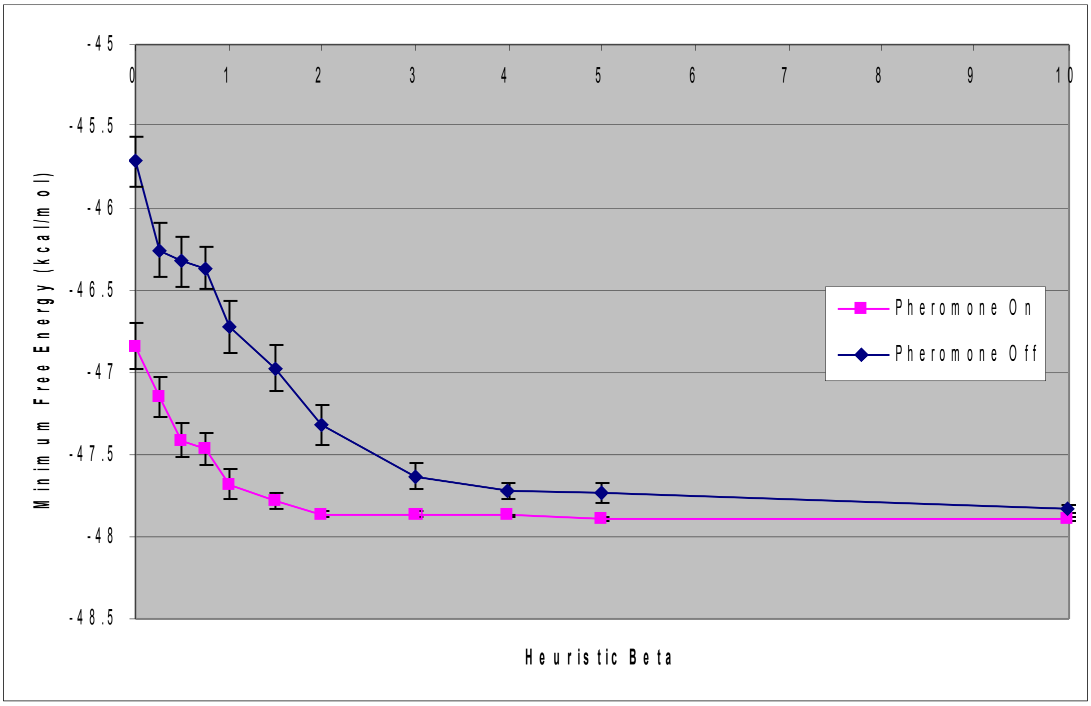

Figure 5.9: Varying β with pheromone update on/off. Sequence: E. Coli.

Figure 5.9 shows the minimum free energy solution from ACO at various values of β. Each point on the graph is the average of 50 experimental runs and the errors bars show the standard error. As can be seen from the graph, varying β shows definite results in terms of the performance of ACO with respect to minimum free energy. One possible explanation is that increasing the contribution of the heuristic with respect to the pheromone produces better results i.e. the heuristic is more important than the pheromone for producing good solutions. However, the second plot on the graph shows that even when pheromone update is disabled the effect of varying β is the same: monotonic improvement in terms of minimum free energy as β increases.

Another explanation for this effect is that the β parameter is emphasising the stems that already have large heuristic values. For example consider two stems *a* and *b,* where *b* has a heuristic value twice as large as *a*. With β set to 10, *b* would become 1024 times as large as *a*. The effect is to strongly promote the stems with the most negative free energy when measured in isolation for the heuristic. This indicates that the heuristic based on the free energy of stems in isolation provides significant a priori knowledge about which stems to put into the structure to achieve minimum free energy overall.

The heuristic is clearly beneficial to the performance of ACO, but it's possible that it is making up for a shortcoming in the DSC problem representation. As described in section 5.1.1 the first stem is selected from the stem pool according to a uniform-random probability. However the first stem is by far the most significant in terms of restricting the range of structures that can be formed[1]. It would be interesting to determine how the performance of ACO is affected when the heuristic is taken into account during the selection of the first stem.

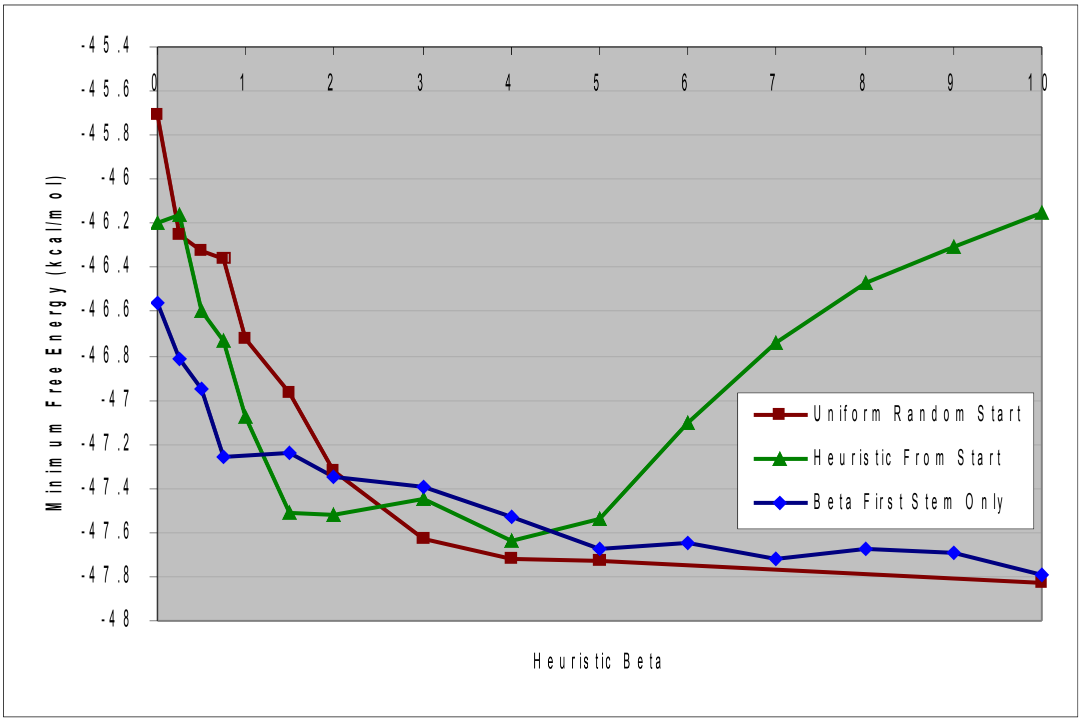

Figure 5.10: Various usage of heuristic when selecting the first stem in the structure. Sequence: E. Coli.

Figure 5.10 shows three alternative ways in which the one-dimensional heuristic is used when selecting the first stem in the structure. Uniform Random Start is the default method which is used in all other experiments in this dissertation. Heuristic

[1] The first stem is always compatible with the existing (empty) structure so is guaranteed to be added, and it precludes a significant number of other stems being added due to base-pair conflicts and pseudoknots. Subsequent addition of stems will of course also reduce range of possible structures but not as much (in absolute terms).

From Start means that the heuristic is always used even when choosing the first stem (this is only possible with a one-dimensional heuristic). Beta First Stem Only means that the first stem uses the β parameter, the rest use $\beta = 1$. Each point on the graph is an average of 50 experimental repetitions. Pheromone update is disabled for all experiments. Other experiment parameters are the same as before.

The graph (Figure 5.10) shows that when the heuristic is used when choosing the first stem, increasing the β parameter initially yields on average better solutions. This is probably due to the increased likelihood of choosing a good first stem. But as β goes above approximately 4 the quality of solutions starts to decrease. This could be because the excessive exploitation of high-heuristic stems hinders exploration of other possibilities and makes it less likely that one of ants will find a better solution by chance. Uniform Random Start wouldn't suffer from this problem because each ant's solution is guaranteed to have some exploration in the selection of the first stem.

Applying β only to the first stem only allows the algorithm to perform better than the other two methods for small values of β. But more interestingly it shows that the heuristic can be applied to the choice of the first stem without sacrificing exploration of the search space. Further work could determine a strategy for varying β during the construction of solutions to further fine-tune the exploration-exploitation balance.

Summarising the results of this section, using a value for β around 4 emphasises stems with greater contribution to the free energy of the structure without compromising exploration of the search space. Also, the use of the heuristic in selecting the first stem in the structure

5.1.8 Minimum Stem Size

As described in section 5.1.1 all experiments so far have used a minimum stem size (*M*) of 4 even though a common definition of a stem has a minimum stem size of 3 (see Figure 4.1a). Real biological structures have stems of size 1 up to any number. For combinatorial optimisation methods the minimum stem size affects the size of the stem pool and hence the range of components that can be used to form structures. It has been assumed so far that this simplification of the problem does not interfere with the other variables being examined and that it is simply the potential quality of the solution and the computation time that would be affected by the choice of minimum stem size.

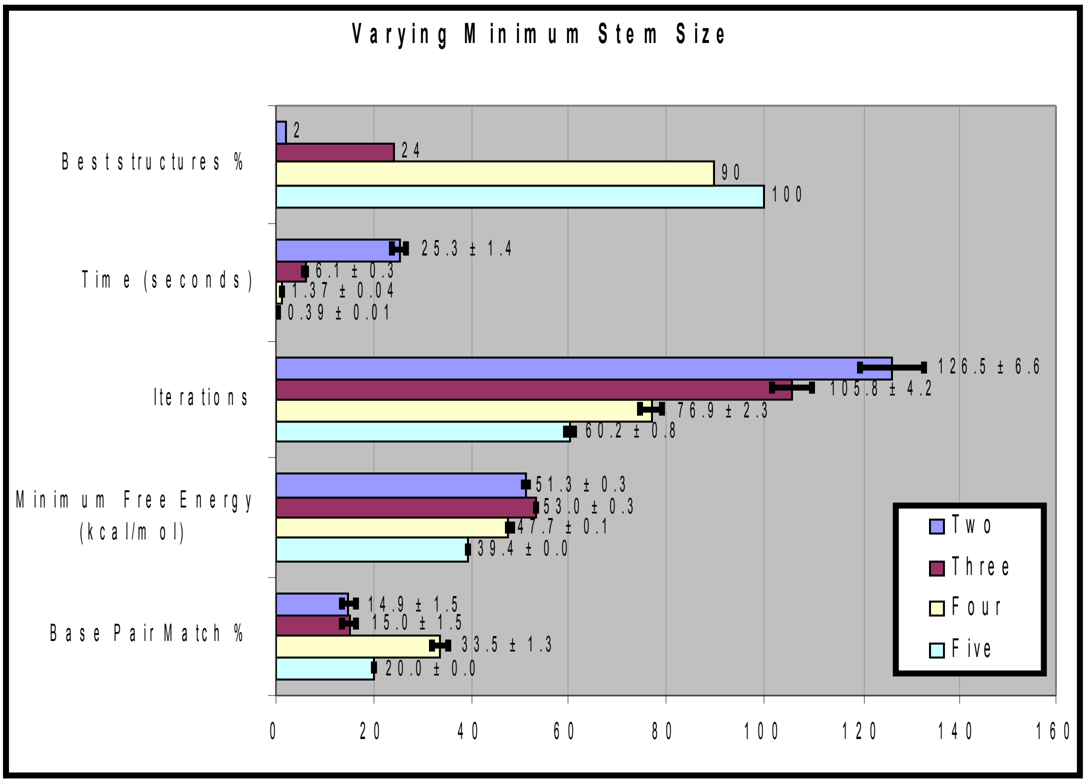

Figure 5.11: This chart shows the difference in algorithm performance when the minimum stem size is varied from two to five base pairs. The top chart 'best structures %' shows what percentage of the 50 experiments yielded the 'best observed free energy' (see Table 5.1). The measurements shown on the other charts are the average of 50 repetitions.

Table 5.1: This table shows further details for the experiment described in Figure 5.13. 'Best observed free energy' is the best out of all 50 repetitions of the experiment.

Minimum stem size *(M)*	**Size of stem pool**	**Best observed free energy (kcal/mol)**
2	1518	-54.8
3	550	-54.9
4	208	-47.9
5	92	-39.41

In general using a lower minimum stem size has the following effects, as can be seen from Figure 5.11 and Table 5.1:

1. Increases the number of stems in the stem pool. This increases the size of the search space but allows for the possibility of finding structures with lower free energy.
2. Increases the number of iterations before termination of the algorithm.

3. Significantly increases the time to complete the algorithm.

Although using $M = 2$ theoretically allows the algorithm to find structures with lower free energy than when it is set to 3, in practice the average minimum energy is slightly lower (see Figure 5.11). This is most likely because the range of possibilities allowed by the stem pool includes better solutions, but it also includes an even larger proportion of sub-optimal solutions. The ‘best observed free energy’ in Table 5.1 confirms this because $M = 2$ results in higher free energy than when $M = 3$. This can only be because the increase in the size of the stem pool makes it harder to find good solutions, even those that were possible with $M = 3$. This is confirmed by the measurement of the percentage of structures that have the ‘best’ free energy. When $M = 3$, 24% (or 12 out of 50 ACO runs) produced a structure with the best observed free energy of –54.9 kcal/mol. This suggests that it is quite likely that this solution (or set of solutions) is optimal for the given stem pool. However when $M = 2$ (or 1 out of 50 ACO runs) produced the best observed free energy of –54.8 kcal/mol. This suggests that the algorithm is not able to reliably find the optimal solution for the given stem pool. It is uncertain whether these findings reflect on the choice of minimum stem size (M) or on the algorithms ability to work with larger stem pools. If the latter then the algorithm may perform poorly on other problems where the stem pool is larger due to the size of the sequence rather than the choice of minimum stem size. Further investigation is required.

5.1.9 Maximal Stems

The number of stems in the stem pool determines the complexity of the problem. If there were a method to reduce the size of the stem pool this would reduce the search space and the length of time required for the algorithm. Considering only maximal stems would considerably reduce the size of the search space and maintain a representative of each stem group.

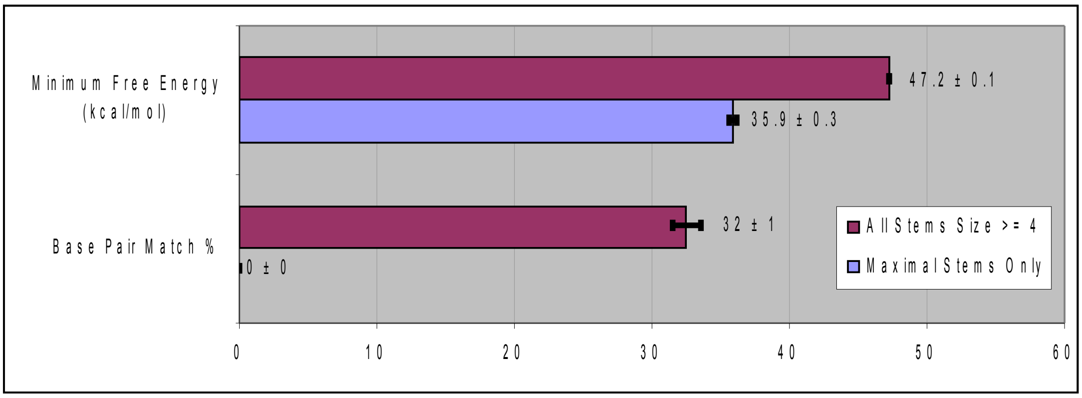

Figure 5.12 shows the results of the experiments into the effects of using only maximal stems. Each measurement is the average of 50 repetitions. When only maximal stems are used the minimum free energy of the ACO algorithm is significantly worse than usual. It is clear that a greater variety of stems is required to make it possible to get optimal structures. Furthermore, the fraction of base pairs in common with the actual biological sequence falls from about a third to zero when the stem pool is restricted to maximal stems. It would seem that the stems featuring in real biological structures are not necessarily locally maximal but happen to combine well to maximise the total number of base pairs in the structure.

As the results from each case are so different, the experiment was only run on one sequence. It is clear that maximal stems are not a useful simplification of the problem so no other experiments in this dissertation make use of this approach.

5.1.10 Local Search

As described in section 4.7 adding local search to ACO can often yield improved results. Figure 5.13 compares a variety local search mechanisms used in conjunction with ACO. For this problem local search adds approximately 0.5 kcal/mol to the average quality of the solutions it generates, but this is at the expense of computing time. Although the number of iterations decreases with the use of local search, the actual time to execute the program is roughly tripled. From the minimum free energy results it is clear that local search is not adding much quality to the solutions. However this could be because using a minimum stem size of 4 makes the problem too easy and even the non-local search algorithm can do well. There is not much scope for improvement so a more challenging test is required.

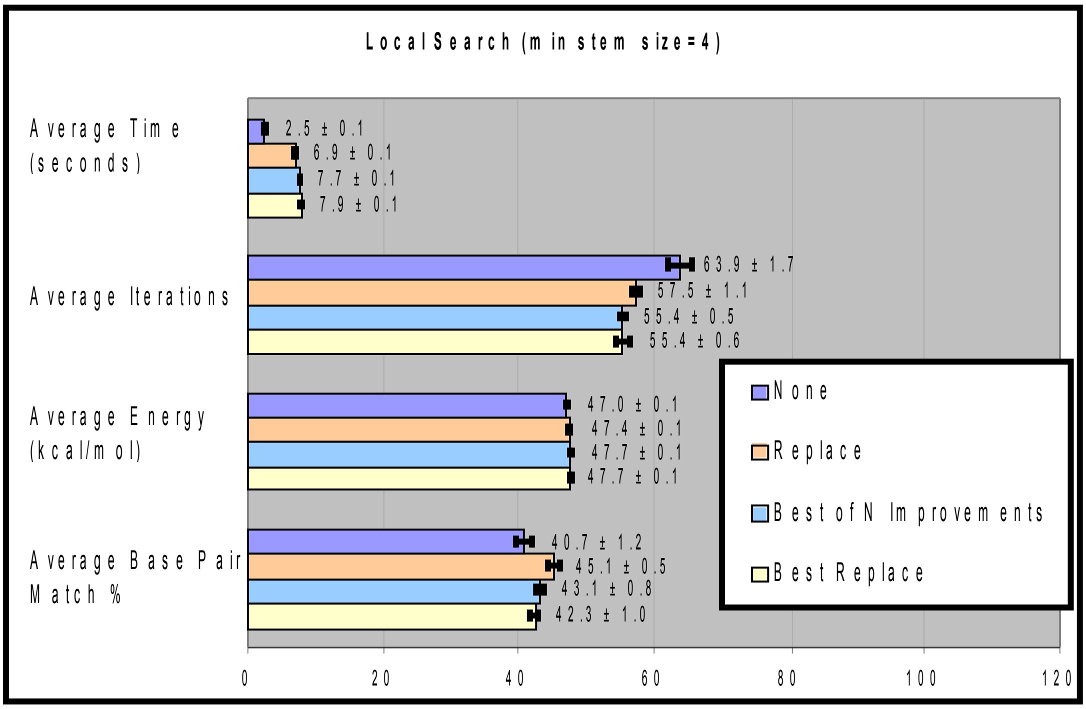

Figure 5.13: This chart shows three local search mechanisms, alongside no local search for comparison.

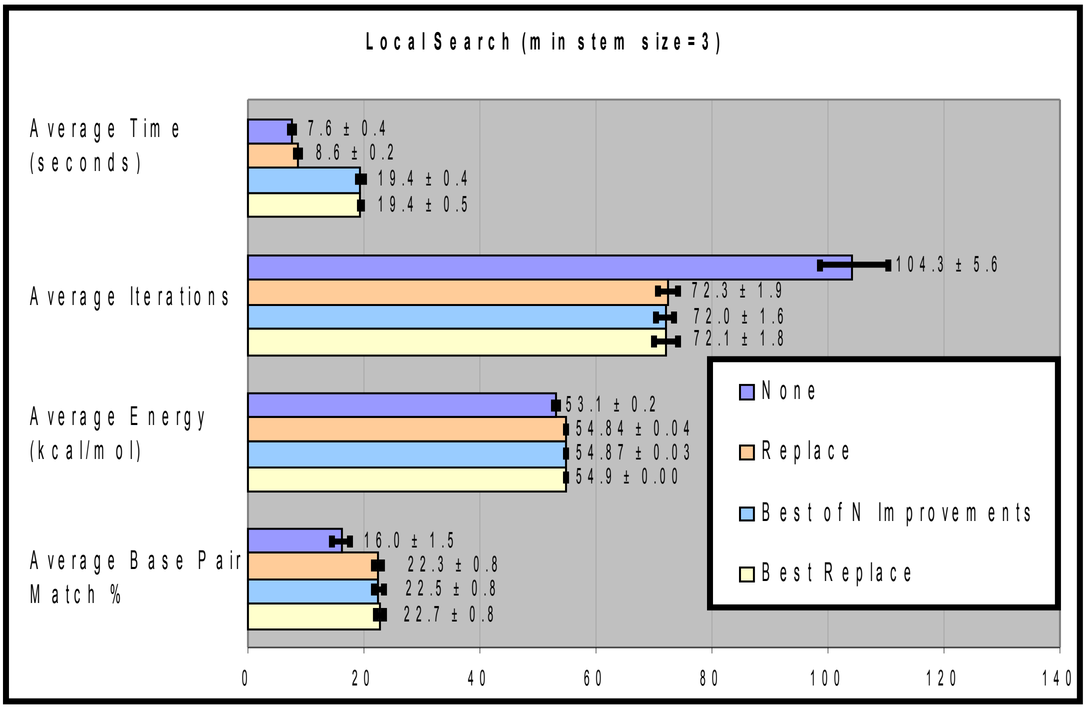

Figure 5.14: This chart is the same as Figure 5.13 except that the minimum stem size is 3.

Figure 5.14 shows the same experiment using a minimum stem of 3. This time local search shows a greater improvement over the basic ACO. Furthermore the relative cost in run-time of the 'Replace' local search over the basic version is not as high as when the minimum set size was 4. A possible reason for this could be as follows. The 'Replace' local search stops as soon as any improvement is found. With a minimum stem size of 3 the problem is harder so on average the solution produced by ACO will have more scope for improvement. Therefore it is more likely that some improvement will be found quickly. In general, the more score there is for improvement the cheaper it is for local search to find it. However this doesn't apply so much to local search policies such as 'Best Replace' and 'Best of 3 Improvements' because they intentionally continue the search for better solutions beyond the 'easy to find' initial improvement.

5.2 Comparison with alternative methods

In this section ACO is compared with some basic combinatorial optimisation approaches and some alternative state-of-the-art approaches to RNA secondary structure prediction. Firstly all approaches are described, including the selected parameters for ACO. Secondly the results of the comparison are presented and discussed.

5.2.1 Choosing the best ACO parameters

Experiments so far into the performance and accuracy of ACO have revealed parameters for ACO that work well. For comparison with alternative approaches it would be useful to select the best parameters in order to represent the full capability of the ACO approach. These parameters are as follows: number of ants = 30, problem representation = direct structure construction, minimum stem size = 3, pheromone update rule = iteration-best, heuristic = one-dimensional free energy based, heuristic β = 4.0, local search = replace.

5.2.2 Random search

If ACO were achieving no ‘learning’ during the algorithm then it would effectively be doing an iterative random search based on a heuristic. It has already been established that learning is taking place for the problem parameters that were tested but it is still useful to do a direct comparison with a random search method. The random search method presented here uses the same direct structure construction approach as ACO, except that the pheromone and heuristic are not used. The addition of each stem to the existing structure is based on a uniform-random distribution across compatible stems.

To make a fair comparison with ACO, both algorithms are restricted to 2500 iterations. However since ACO uses multiple ants it is restricted to commensurately fewer iterations.

5.2.3 Hill Climbing

Hill-climbing (HC) is a simple approach to solving combinatorial optimisation problems. The basic idea is start with a random state and repeatedly move to a state that is at least as good as the current state. There are many varieties of hill-climber varying on the policy for state transitions e.g. a simple HC chooses the first state encountered that is better than the current, and steepest ascent HC always chooses the best next state. The main problem with HC is that the algorithm terminates when local optima are reached i.e. better states may exist, but there is no better state adjacent to the current state.

The implementation of HC that is presented here starts with a structure produced by uniform-random direct structure construction. This structure is then incrementally improved upon using the 'replace' local search (see section 5.1.10). It terminates when there no local improvement is possible.

5.2.4 Dynamic Programming

As described in section 2.3.2.1 there are several dynamic programming approaches to RNA secondary structure prediction. The following dynamic programming algorithms are compared with ACO:

- Vienna RNA package [Hofacker, 2003]
- MultiRNAFold package [Andronescu et al., 2005]
- RNA Structure 4.2 [Mathews et al., 2004], which is an implementation of mFold [Zuker, 2003] with a Windows GUI.

5.2.5 Comparison

A series of experiments were conducted using the same RNA sequence and a variety of secondary structure prediction methods. Figure 5.15 shows the results of these experiments. The results from hill climbing, random search and ACO are all shown as averages of 50 repetitions. The results from the three dynamic programming (DP) methods (MultiRNAFold, RNA Structure and Vienna) are from a single run because the same input always yields the same output. Two out of the three dynamic programming methods agree on the optimal structure, and the third is very similar. It is likely that this variation is due to slight differences in the energy calculation function rather than limitations of the DP method. All ACO runs equalled the performance of DP in terms of minimum free energy although there were slight variations in the quality of the stem match and base-pair match. Overall, ACO is on a par with dynamic programming for determining the optimal structure of the E. Coli sequence, 120 nucleotides long. By comparison the poorer performance of the simpler Hill Climbing and Random Search algorithms demonstrates that the problem is not trivial and that ACO is performing a useful task.

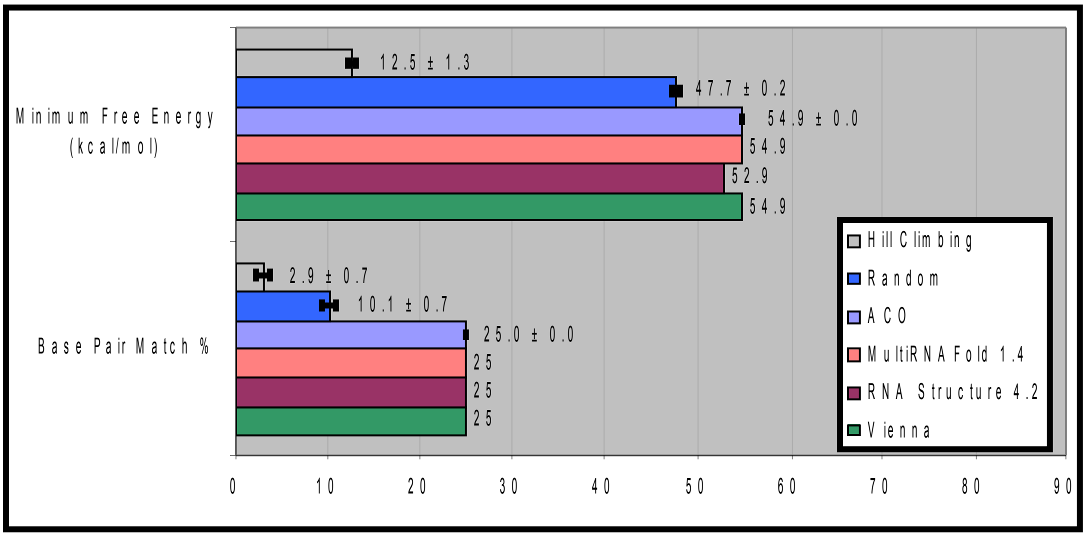

Figure 5.15: Comparison of 6 methods of RNA secondary structure prediction. Sequence: E. Coli.

In order to draw meaningful conclusions from these comparisons it is necessary to consider other, perhaps longer, sequences. Table 5.2 shows data from comparisons of ACO with the Vienna RNA package across 5 different sequences. All ACO data is the average of at least 10 repetitions, and DP data is from a single run. As the length of the sequences increases so does the number of stems in the ACO stem pool, and as a result the algorithm takes considerably longer to terminate. However the quality of the solutions with respect to the minimum free energy and similarity to the real biological structures also decreases as the length of the sequence increases.

Table 5.2: Summary of results for five sequences. Compares ACO with Vienna DP method.

		Sequence Length	**Size of Stem Pool**	**Natural Free Energy (kcal/mol)**	**Predicted Free Energy (kcal/mol)**	**Base Pair Match %**	**Time (seconds)**
S. Cerevisiae	ACO	118	582	-44.1	-52.8	67.0	12
	DP				-54.1	66.7	0.1
E. Coli	ACO	120	550	-47.0	-54.9	25.0	12
	DP				-54.9	25.0	0.1
H. Rubra	ACO	543	16289	-114.7	-190.7	27.3	598
	DP				-204.5	41.3	0.4
T. Thermophila	ACO	506	10296	-97.2	-159.3	39.6	371
	DP				-177.4	67.8	0.4
C. Elegans	ACO	697	34617	unknown	-107.5	15.8	1640
	DP				-142.5	18.6	2

As described in section 2.3.2.1, one of the drawbacks of simple DP is that it is sometimes restricted to providing a single, optimal solution. ACO can easily present a range of candidate solutions at once. A more valid comparison of the strengths of these alternative approaches may be to consider a set of results (e.g. the top ten) from each run, rather than just one solution with the lowest free energy. The Vienna RNA package is one of the few DP approaches that can produce multiple suboptimal structures. On a head-to-head comparison using S. Cerevisiae the best out of the top-ten structures produced by ACO had 69% of base pairs in common with the natural structure, but the best produced by Vienna had 84% in common. Using the E. Coli sequence, Vienna also shows better results than ACO. Although these are good results for both approaches it would appear that DP is superior to ACO both in terms of speed and quality of solutions.

Chapter 6

Conclusion

In this work the combinatorial optimisation technique 'Ant Colony Optimisation' has been applied to the problem of RNA secondary structure prediction for the first time. Two problem representations have been proposed, allowing the application of ACO to this particular problem. This work also proposes a one-dimensional heuristic based on the free energy of stems in isolation, and experiments show that it improves the quality of solutions whilst reducing the number of iterations required. The 'Replace' local search method is also proposed, yielding further improvements in quality of solution at the expense of processing time.

A wide range of parameters were varied and tested to determine a good set to be used as a representative sample of the capability of the ACO method. This was then compared with several dynamic programming (DP) methods. For short sequences around 120 nucleotides ACO yields similar results to DP but takes longer to execute. For longer sequences DP is both faster and more accurate than ACO. Overall, this work has resulted in a reasonably good method of RNA secondary structure prediction but it has been shown to be inferior to existing state-of-the-art methods.

Possible extensions to this work for the pursuit of knowledge would be to consider further parameters to the ACO algorithm such as α, controlling the contribution of the pheromone trail and ρ, the pheromone evaporation rate. In addition, alternatives for the heuristic, problem representation and local search could yield better results, and of course the algorithm would benefit from further optimisation. However if the objective is to find a faster or more accurate methods of RNA secondary structure prediction then ACO is most likely not the answer. If many related sequences are available then comparative methods should be considered. Alternatively for predicting the structure of individual sequences, dynamic programming works well. And of course the search continues to find yet better predictive methods.

Bibliography

[Abrahams et al., 1990] Abrahams, J. P., van den Berg, M., van Batenburg, E. & Pleij, C.: "Prediction of RNA secondary structure, including pseudoknotting, by computer simulation", Nucl. Acids Res. 18, 3035-3044, 1990.

[Akutsu, 2000] Akutsu T.: "Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots". *Discrete Appl. Math.*, 104, pp. 45–62, 2000.

[Andronescu et al., 2005] Andronescu, M., Zhang, Z.C. and Condon, A.: *"Secondary structure prediction of interacting RNA molecules"*, J. Mol. Biol., 345, 987–1001, 2005. URL: http://www.rnasoft.ca/download/README.html

[Benedetti and Morosetti, 1995] Benedetti G., Morosetti S.: "A genetic algorithm to search for optimal and suboptimal RNA secondary structures". *Biophys Chem*, 55, pp. 253-259. 1995

[Bonabeau et al., 2000] E. Bonabeau, M. Dorigo, and G. Theraulaz. "Inspiration for optimization from social insect behavior". *Nature*, 406:39–42, 2000.

[Bullnheimer et al., 1999] B. Bullnheimer, R.F. Hartl, C. Strauss, "An improved Ant System algorithm for the Vehicle Routing Problem", *Annals of Operations Research*, Volume 89, Issue 0, Jan 1999, Pages 319 – 328

[Cannone et al., 2002] Cannone J.J., Subramanian S., Schnare M.N., Collett J.R., D'Souza L.M., Du Y., Feng B., Lin N., Madabusi L.V., Muller K.M., Pande N., Shang Z., Yu N., and Gutell R.R.: "*The Comparative RNA Web (CRW) Site: An Online Database of Comparative Sequence and Structure Information for Ribosomal, Intron, and other RNAs."* Central Bioinformatics. **3**:15, 2002. URL: http://www.rna.icmb.utexas.edu/

[Chemis, 2000] Chemis interactive Molecular Library, 1999-2000. URL: http://www.geneticengineering.org/chemis/Chemis-NucleicAcid/RNA.htm

[Coffman et al., 1996] Coffman, E. G., Garey, M. R., and Johnson, D. S.: "*Approximation Algorithms for Bin Packing: A Survey"*, pages 46–93. PWS Publishing, Boston, MA, USA, 1996.

[Deneubourg et al., 1990] Deneubourg J.-L., Aron S., Goss S., and Pasteels J.-M.. "The self-organizing exploratory pattern of the Argentine ant", *Journal of Insect Behavior*, 3:159–168, 1990.

[Deogun et al., 2004] Deogun J., Donis E., Komina O., Ma F.: "RNA secondary structure prediction with simple pseudoknots", *In Proc Second Asia-Pacific Bioinformatics Conference 2004*, pp. 239-246. 2004.

[Dorigo, 1992] M. Dorigo, "Optimization, learning and natural algorithms, PhD Thesis," Dipartimento di Elettronica, Politechico di Milano, Italy, 1992.

[Dorigo and Gambardella, 1997] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[Dorigo and Stützle, 2001] Dorigo, M. and Stützle, T. (2001). The ant colony optimization metaheuristic: Algorithms, applications, and advances. to appear in Handbook of Metaheuristics, F. Glover and G. Kochenberger.

[Dowell and Eddy, 2004] Dowell R, Eddy S: “Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction.” *BMC Bioinformatics*, **5:**71-71, 2004.

[Ducatelle and Levine, 2001] F. Ducatelle and J. Levine, “Ant Colony Optimisation for Bin Packing and Cutting Stock Problems,” presented at UK Workshop on Computational Intelligence (UKCI-01), Edinburgh, 2001.

[Freier et al., 1986] Freier S. M., Kierzek R., Jaeger J. A., Sugimoto N., Caruthers M. H., Neilson T. & Turner D. H.: “Improved free-energy parameters for predictions of RNA duplex stability”, *Proc. Natl Acad. Sci. USA*, 83, pp. 9373–9377, 1986.

[Goss et al., 1990] Goss, S., Beckers, R., Deneubourg, J. L., Aron, S., and Pasteels, J. M.: “How trail laying and trail following can solve foraging problems for ant colonies”. In Huges, R. N., editor, *Behavioural Mechanisms of Food Selection*, volume 20G, Berlin. Springer-Verlag, 1990.

[Grassé, 1959] P.P. Grassé, “La reconstruction du nid et les coordinations interindividuelles chez *bellicositermes natalensis et cubitermes* sp. La théorie de la stigmergie: essai d’interprétation du comportement des termites constructeurs”, Insectes Sociaux 6, 41–81, 1959.

[Knudsen and Hein, 1999] Knudsen B. and Hein J.: “RNA secondary structure prediction using stochastic context-free grammars and evolutionary history”, *Bioinformatics*, 15, pp. 446–454. 1999.

[Higgs, 2000] Paul G. Higgs: “RNA secondary structure: physical and computational aspects”, Quarterly Reviews of Biophysics 33, 3, pp. 199–253, 2000.

[Hofacker, 2003] Ivo L. Hofacker: “Vienna RNA secondary structure server”, Nucleic Acids Research, Vol. 31, No. 13 3429–3431, 2003.

[Hofacker, 2006] Ivo Hofacker, Vienna RNA Package: RNA Secondary Structure Prediction and Comparison. http://www.tbi.univie.ac.at/~ivo/RNA/

[Holland, 1975] Holland, J.H.: "Adaptation in Natural and Artificial Systems", *University of Michigan Press*, Ann Arbor, Michigan, 1975.

[Holbrook, 1997] Holbrook S. R. and Kim S.-H.: "RNA Crystallography," in *Biopolymers.* vol. 44,. pp. 3;21, 1997.

[Lin and Kernighan, 1973] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Oper. Res.*, vol. 21, pp. 498–516, 1973.

[Maniezzo et al., 1994] V. Maniezzo, A. Colorni, and M. Dorigo, "The ant system applied to the quadratic assignment problem," Universite Libre de Bruxelles, Belgium, Tech. Rep. IRIDIA/94-28, 1994.

[Martello and Toth, 1990] Martello, S. and Toth, P.: "*Knapsack Problems, Algorithms and Computer Implementations"*. John Wiley and Sons Ltd., England, 1990.

[Mathews et al., 2004] Mathews, D.H.; Disney, M.D.; Childs, J.L.; Schroeder, S.J.; Zuker, M.; and Turner, D.H., "Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure," 2004. *Proceedings of the National Academy of Sciences, USA. 101.* 7287-7292. http://rna.urmc.rochester.edu/rnastructure.html

[McCallum, 2005] McCallum T., *"Understanding how Knowledge is exploited in Ant Algorithms"*, PhD Thesis, University of Edinburgh, 2005.

[Mironov et al., 1985] Mironov AA, Dyakonova LP, Kister AE., "A kinetic approach to the prediction of RNA secondary structures.", *J Biomol Struct Dyn.* 1985 Feb;2(5):953-62.

[Nussinov and Jacobson, 1980] Nussinov, R. & Jacobson, A. B.: “Fast algorithm for predicting the secondary structure of single-stranded RNA.”, *Proc. natn Acad. Sci. USA* 77, 6309-6313, 1980.

[NDB] The Nucleic Acid Database Project. Rutgers, The State University of New Jersey, 1995-2006, http://ndbserver.rutgers.edu/NDB/

[Pedersen et al., 2004] Pedersen J., Skou M., Irmtraud M., Forsberg R., Simmonds P., Hein J.: “A comparative method for finding and folding RNA secondary structures within protein-coding regions” in *Nucl. Acids Res*. 32: 4925-4936, 2004.

[Petersheim et al., 1983] Petersheim, M., & Turner, D. H.: “Base-Stacking and Base-Pairing Contributions to Helix Stability: Thermodynamics of Double-Helix Formation with CCGG, CCGGp, CCGGAp, ACCGGp, CCGGUp, and ACCGGUp” in *Biochemistry 22*, 256-263, 1983.

[Reeves, 1996] Reeves, C.: “Hybrid genetic algorithms for bin-packing and related problems”. *Annals of Operations Research*, 63:371–396, 1996.

[Rivas and Eddy, 1999] Rivas E, Eddy SR: “A dynamic programming algorithm for RNA structure prediction including pseudoknots.” *J. Mol. Biol* 1999, 285**:**2053-2068.

[Shapiro and Navetta, 1994] Shapiro B., Navetta J.: “A massively parallel genetic algorithm for RNA secondary structure prediction”, *The Journal of Supercomputing*, vol. 8, pp. 195–207, 1994.

[Shapiro et al., 2001] Shapiro B.A., Wu J.C., Bengali D. and Potts M.J. “The massively parallel genetic algorithm for RNA folding: MIMD implementation and population variation”. *Bioinformatics*, 17, pp. 137-148, 2001.

[Shmygelska and Hoos, 2003] Alena Shmygelska, Holger H. Hoos, An Improved Ant Colony Optimisation Algorithm for the 2D HP Protein Folding Problem, Lecture Notes in Computer Science, Volume 2671, Jan 2003, Pages 400 – 417

[Steeg, 1993] Steeg E.: “Neural networks, adaptive optimization, and RNA secondary structure prediction.” In: Hunter L (ed) Artificial intelligence and molecular biology. AAAI Press MIT Press, Menlo Park Cambridge, Mass. pp 121–160, 1993.

[Van Batenburg et al., 1995] F. H. D. van Batenburg. **A.** P. Gultyaev, and C. W. **A.** Pleij, 'An APL-programmed genetic algorithm for the prediction of RNA secondary structure" *Journal of Theoretical Biology*, vol. 174, pp. 2~9-280. 1995.

[Varani et al., 1996] Varani G., Aboul-ela F. and Allain, F.H.-T. NMR investigations of RNA structure. *Prog. NMR Spectrosc.*, 29, pp. 51-127, 1996.

[Wiese et al, 2005] Wiese K. C., Hendriks A., and Poonian J.: "Algorithms for RNA Folding: a Comparison of Dynamic Programming and Parallel Evolutionary Algorithms", Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC 2005), Edinburgh, Scotland, 2005.

[Zuker, 1989] Zuker, M.: “Computer prediction of RNA structure.” Meth. Enzymol. 180, 262-88, 1989.

[Zuker, 1989b] Zuker, M.: “On finding all suboptimal foldings of an RNA molecule.” Science, 244, 48–52, 1989.

[Zuker, 2003] M. Zuker, “Mfold web server for nucleic acid folding and hybridization prediction”. *Nucleic Acids Res.* 31 (13), 3406-15, (2003). http://www.bioinfo.rpi.edu/applications/mfold/