Inverse folding of RNA

Rune B. Lyngsø

1 Background

The advent of nano-sciences, where we want to design small entities with specific structure or behaviour, has already spurred an interest in using nucleic acid sequence molecules as the fundamental building block, with some rather curious examples (see e.g. [23, 2]). It is only natural that DNA and in particular RNA molecules have been considered for nano-scale design as

- numerous motifs with specific functionalities are known from biology
- large scale production can be carried out by normal replication of simple organisms
- we already possess an extensive understanding of structure formation for nucleic acid sequences, allowing relatively good predictions of both secondary and tertiary structure purely by computational means

The aim of the inverse folding problem for RNA is, given a target structure like e.g. the one depicted in Fig. 1, find a sequence that folds into this structure.

In this project we will exclusively focus on the secondary structure. The main driving force behind RNA structure formation is the creation of base pairs similar to the ones observed in the DNA double helical structure. In RNA thymine is replaced with uracil, so the Watson-Crick base pairs of RNA are between C and G, and between A and U. Additionally the so-called wobble base pair between G and U is commonly observed, and these six (when accounting for ordering) types of base pairs are known as the *canonical* base pairs of RNA. The main difference between DNA and RNA is that where the DNA helix is formed between complementary strands, an RNA molecule consists of just a single strand, or sequence, and the helices are formed locally between different parts of the sequence. The secondary structure of an RNA molecule is the set of base pair interactions observed in the full three dimensional structure.

In lieu of costly experiments to determine the true structure, the secondary structure predicted by standard methods is usually used to determine the fold of proposed sequences. RNA secondary structure prediction is one of the classical problems in computational biology [12]. For decades it has been applied to make inferences about the structure of sequenced RNA, and more recently as an integral part of non-coding RNA gene finding [27, 21]. One standard method is based on a model assigning free energies to secondary structures [26, 19].

Thermodynamics then states that the most stable structure is the one with lowest free energy, and this is usually taken to be the true structure. This model has the further advantage of defining a (Boltzmann) distribution over structures from which e.g. individual base pair probabilities can be estimated [20]. A closely related approach uses stochastic context-free grammars [11], allowing similar inferences to be made.

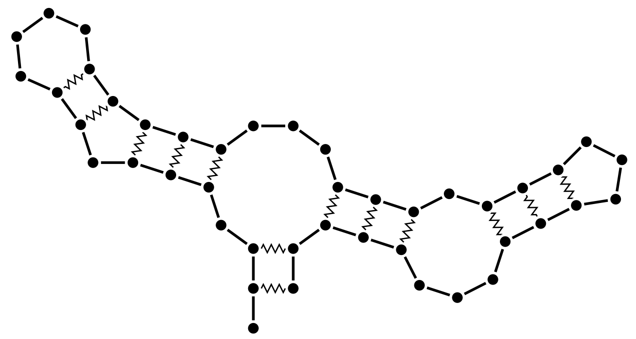

Figure 1: An example target RNA structure. The sequence backbone follows the straight lines with circles representing the nucleotides. Nucleotides connected by a zigzag line should form base pair interactions in the stable structure for the designed sequence.

The first method, RNAinverse for computational design of RNA molecules folding into a particular structure was published as little more than a footnote in a larger context of RNA secondary structure prediction and comparison [16]. It essentially starts from a random sequence and changes the base at positions that have an incorrect structure in the current prediction, until a sequence with the target structure is obtained. In the last decade, several new players have entered the field. The RNA-SSD [4, 1] and INFO-RNA [7] methods are very similar in fundamental structure to RNAinverse, with NUPACK:DESIGN [29], Inv [14], and MODENA [25] differing more in approach and problem addressed.

INFO-RNA mainly differs in how the initial sequence is generated by finding a sequence with minimum folding energy given the target structure. It further uses energy changes to guide the local search process.

RNA-SSD also tries to be more clever with the initial guess, but in a more heuristic way just trying to avoid repetitions of long sequences and their complement. The search is made more efficient by a hierarchical decomposition of the target structure and use of small *caps* in place of missing substructures. The most recent version also allows constraints on possible bases at specific positions.

NUPACK:DESIGN uses an optimisation criteria based on minimising the expected number of incorrectly paired positions for the designed sequence.

This allows design of sequences where the structure is stable not only compared to the completely unpaired structure but also compared to other structures. Moreover, the method also allows design of heterodimers and heteropolymers, i.e. set of sequences that assemble into a target structured complex.

Inv mainly differs in being able to design sequences for structures containing crossing base pairs, also known as pseudoknots (and in the paper containing several pages of pseudocode).

MODENA uses a more complex multi-objective (closeness to target structure and energy of predicted structure) scoring scheme. It furthermore introduces application of genetic algorithms to the problem, using a target structure constrained crossover operation to recombine candidate sequences. A more modular view on prediction methods also allows different prediction methods to be used, but this also results in a loss in efficiency as the bottom-up approach of RNAinverse or the decomposition method of RNA-SSD to mostly limit predictions to shorter sequences are not employed.

2 Project

The aim of this project is first to develop an improved method for the original inverse RNA folding problem, and then to extend this method to handle new variants of the inverse folding problem. There are a number of other issues that could also be addressed, and a list of some extensions is included in the last section for inspiration. The project described in this section aims to address the most relevant and realistic topics, but it is not set in stone should you want to explore other aspects of the inverse folding problem. Also, this section does give a rather detailed description. This is not intended to be a strict work plan that has to be followed in minute detail, and should be modified according to your interests and ideas and experiences gained as the project progresses.

Week 1–3: Improved Genetic Algorithm

One of the simplest optimisation techniques is a local guided search, e.g. hill climbing. Some (or all) configurations that are neighbours to the currrent configuration according to one or more simple types of updates – e.g. sequences that can be reached by changing a single nucleotide in the current sequence – are considered, and based on their scores one of these is chosen as the new configuration. This is exactly the approach of RNAinverse [16]. It works well when there is a single optimum, but with many local optima there is a high risk of getting trapped in a mediocre local optimum rather than finding the global optimum.

This problem can to an extent be alleviated by using multiple starting points, but only as long as the number of local optima does not vastly outnumber the number of starting points it is feasible to use. What is needed to better explore the search space is to be able to make large jumps. If these jumps are completely random, this amounts to no more than multiple starting points. The idea of genetic algorithms (GAs) is to attempt to guide the jumps to mainly generate novel, but still good, configurations.

The technique is inspired by natural evolution: a population is maintained and allowed to mutate and interbreed, and the configurations making it to the next generation are selected based on their fitness. In addition to the simple updates of a standard local search heuristic – usually referred to as mutations in GA terminology – GAs also defines crossover events that combines two configurations into one. For example, if configurations are sequences of length n, a crossover between sequences s and t could create a new sequence as $s[1..i] \cdot t[i+1..n]$, i.e. by taking the first i characters from s and the remaining $n-i$ characters from t. From a set of current configurations, new configurations are generated by mutations and crossovers. Finally the next generation of configurations are selected either deterministically or at random from the combined set of current and new configurations, based on a fitness computed for each configuration. The general structure of a GA is illustrated in Alg. 1.

Algorithm 1 Abstract structure of a genetic algorithm

```
Create initial population $P$ of putative solutions
while we still think we can do better and have more time do
    $M = \emptyset$
    for number of mutations we want to try do
        Create new configuration $y$ by applying mutation to an $x \in P$ and add
        it to $M$
    $C = \emptyset$
    for number of crossovers we want to try do
        Create new configuration $z$ by applying crossover between two configu-
        rations $x, y \in P$ and add it to $C$
    Update population $P$ for next generation by selecting a subset of desired
    size from $P \cup M \cup C$ based on the fitness of each configuration
```

The crossover operation only improves on making random jumps when good total solutions consist of good partial solutions. If there is no correlation between the fitness of the two configurations going into a crossover and the configuration resulting from it, we might as well introduce completely random configurations. The inverse RNA folding problem does seem ideally suited for a genetic algorithm (GA) approach: it appears to be hard to find an optimal solution [24], however we would expect a good global solution consists of good local solutions – the latter indeed being the rationale for the decomposition approach of RNA-SSD and bottom-up approaches of other methods. It is thus surprising that MODENA seems to be the only GA suggested so far. The MODENA GA does do some clever things, but still leaves room for improvement.

The first task in this project consists of developing your own, hopefully better, GA for the inverse RNA folding problem. A rough outline of this could be as follows.

- Read the key background litterature for understanding RNA secondary structure and its prediction [12, 17, 20], one or more of the inverse folding papers [16, 7, 4], and the MODENA paper [25], as well as the Wikipedia entries on nucleic acid secondary structure and genetic algorithm.
- Download the Vienna package from `www.tbi.univie.ac.at/~ivo/RNA/` and get it to work. The main programs you will need are RNAfold, RNAdistance, and possibly RNAinverse to compare against.
- Implement a basic GA for inverse RNA folding. You can either
  - reimplement the MODENA approach, possibly even using the NSGA-II [9] bundle available from `www.iitk.ac.in/kangal/codes.shtml`. It seems to be fairly straight forward to define objective functions in this framework. However, mutation and crossover seems to be more difficult to control. To gain control over this part would probably require rewriting the relevant parts of the software. Also, there doesn't seem to be any user manual available.
  - implement your own GA. Though initially you could just consider a single objective, it may be a good idea to prepare for multi-objective optimisation. It can be argued that the stablity objective used in MODENA is not that relevant, but better objectives, e.g. capturing probabilities of the target structure, could be included to make a good combination of objectives. Moreover, the next part of the project will focus on multiple target structures, where it would be natural to have one or more objectives for each target. The good news is that the non-dominated sorting algorithm at the core of NSGA-II [9] is fairly simple, so it shouldn't be difficult to implement.
- Improve the basic GA by making more informative choices in the mutation, crossover, and selection steps:

  Mutation The natural mutation for this problem consists of changing a single nucleotide in the current sequence, or possibly two nucleotides required to form a base pair in the target structure. In MODENA all nucleotides are changed at random with a uniform probability, while the RNAinverse method chooses a random position where the predicted structure does not match the target structure. The Boltzmann distribution [20] computations allow us to assess the probability of each base pair and unpaired nucleotide in the target structure for a given sequence. I.e. we get an estimate of how close we are to getting it right, or, if a target base pair or unpaired base is in the predicted structure, how close we are to getting it wrong. It would make sense to use this information to choose nucleotides to mutate in a non-uniform way. However, some experimentation will be needed to test whether it is better to bias the choice towards first improving parts that are close to being right or parts that are far from being right.

There are a few more complex extensions that shouldn't be among the first things you do, but that could be considered at a later stage. One is to also use Boltzmann probabilities to bias the choice of the replacement nucleotide, with a preference for replacements that brings us closer to getting the target structure right. To do this efficiently will require more involved use of the Vienna package, possibly even modifications to the source code, though. Another extension would be to replace short substrings instead of single nucleotides, as this would make it easier to prevent a particular stretch forming unwanted base pairs. However, to be of any value this will require developing methods for choosing good replacement strings, i.e. strings whose reverse complement does not occur anywhere in the current sequence.

Crossover As already mentioned, GAs only work when good total solutions consist of good local solutions. What constitutes a local solution depends on how the crossover events are defined, in this case how we choose the crossover points. The example given above, taking a prefix from one sequence and the corresponding suffix from the other, would not work well for the inverse RNA folding problem. If two positions required to form a base pair in the target structure are not taken from the same sequence, there is a good chance the nucleotides will not form a canonical base pair. This is observed in MODENA where base pairs define the possible crossover points – if base pair $i \cdot j$ is used for crossover when recombining s and t, $s[1..i-1]$ and $s[j+1..n]$ will be used from s with $t[i..j]$ from t inserted in the middle[1].

The MODENA approach limits crossovers to occur next to a nucleotide forming a base pair in the target structure. There can be situations where the best crossover position is in a stretch of unpaired positions. One can observe that as long as the crossing over and crossing back occurs in the same loop (see [17, Fig. 6]), it will be ensured that positions forming a base pair in the target structure are taken from the same sequence. For example, for the structure in Fig. 1 and assuming that the start of the sequence is the unpaired base at the bottom, we would have to choose a pair of positions from one of the sets $\{2, 42\}$, $\{3, 4, 19, 20, 21, 22, 41\}$, $\{5, 18\}$, $\{6, 17\}$, $\{7, 8, 16\}$, $\{9, 15\}$, $\{10, 11, 12, 13, 14\}$, $\{23, 40\}$, $\{24, 39\}$, $\{25, 26, 35, 36, 37, 38\}$, $\{27, 34\}$, $\{28, 33\}$, and $\{29, 30, 31, 32\}$, cross over after the first position, and cross back after the second position. Moreover, we can also make a single crossover, without crossing back, between two consecutive positions that are external in the target structure; in Fig. 1

[1]The situation is slightly more complicated as MODENA can utilise multiple crossover base pairs in a single crossover event.

the only such case is after position 1. By first choosing a set with probability proportional to its size, and then choose a pair from the set uniformly at random ensures that crossover positions are chosen uniformly at random.

Rather than drawing from a uniform distribution, as suggested for mutations we can also use Boltzmann probabilities in crossover events, biasing it so we are more likely to copy from the sequence that locally is closer to matching the target structure. In the extreme case one could deterministically for each unpaired position and pair of positions forming a base pair in the target structure choose to copy the nucleotide(s) from the sequence having the larger probability for being correct. However, one should be careful not to make GAs too deterministic as this can prevent escaping from a region of the search space that locally appears good but globally is rather poor. It can be necessary to allow some seemingly bad moves at times to make the GA capable of escaping a local optimum.

A further bias could be obtained by implementing adaptive *recombination hotspots*. Keeping track of the location of crossover points and the fitness of the resulting sequence relative to the parents, it may be that certain crossover regions emerge as generally resulting in larger improvements. A similar thing can of course be implemented for mutations, but with the problem at hand there is more reason to believe that it can work for crossovers.

Selection The ultimate test of the fitness of a sequence designed to fold to a target structure is of course to use some kind of distance measure between target and predicted structure. MODENA further adds a stability criteria to arrive at a pair of objectives. However, the stability used in MODENA is the free energy of the predicted structure, which only measures stability relative to the completely unfolded structure, and not relative to all the other competing structures.

A better measure of stability is the Boltzmann probability of the structure, i.e. finding a sequence where the Boltzmann probability of the target structure is maximised. This can be refined even further, using the Boltzmann probabilities of the base pairs and unpaired positions of the target structure – if two sequences are equally good at matching the target structure, we would tend to prefer the one that even when wrong has most of the structure match the target. However, the latter gives an objective for each base pair and each unpaired position, which may be a bit unwieldy. Experimentation with combinations of summary statistics of base pair and unpaired position Boltzmann probabilities like average (used in NUPACK:DESING [29]), minimum, or product of values should be explored to determine which provides the best performance.

Rather than just using a fixed fitness function, it may also be sensible to let the fitness function adapt to where the target structure appears to be most difficult to design a sequence for. A simple scheme would be to weigh fitness contributions from positions inversely proportional to the fraction of sequences in the current population that have predicted structure match the target structure at that position.

The good and bad thing about GAs is that there are always more things you can try. The suggestions above provides a solid solution based on key features of how RNA secondary structure stability is modelled, and known methods for analysis within this model. If you come across something else you think is sensible to explore, go for it, but be careful not to get bogged down tinkering.

In the above, only the part of the GA relating to evolving a population of configurations has been considered. If this works well, it shouldn't matter too much where we start from. However, an initial population still needs to be generated, and if this is done in an intelligent manner it may shorten the search. Probably the simplest way is to choose a random base for each unpaired position in the target structure and a random canonical base pair for each pair of positions forming a base pair in the target structure. This can either be done uniformly, or e.g. based on the distributions available at `www.daimi.au.dk/~compbio/pfold/`. More complicated approaches to this are briefly discussed in the next section

Week 4–5: Multiple Targets

Though MODENA claims to be multi-objective, mostly it just find sequences perfectly matching the target structure relatively early in the search. From this point onward the only remaining objective is to minimise the energy of the predicted structure. A more proper multi-objective inverse RNA folding problem would be in situations where we seek affinity to multiple target structures. A classic example of RNAs with multiple structures are riboswitches [18], that in the presence of a small target molecule, changes in temperature, or for other reasons undergo structural changes. Quite often these changes are relatively small on the secondary structure level, but the active and inactive structures of the SV11 RNA sequence [6] provides an example with complete reorganisation of secondary structure.

Looking beyond known biological riboswitches, in nano technology it would be interesting to design molecules that can adapt or react to the environment, rather than just form one fixed structure. Indeed, one of the few examples of inverse RNA folding methods that allows multiple targets was developed in the context of information processing, or computation [22]. Another alternative use would be to design RNA sequences that can be embedded in genomic sequence and use an organisms own mechanisms, e.g. for maturation of μRNAs, to excise the designed RNA sequence. Once excised, the 'mature μRNA' could then fold into the desired structure.

Experimental techniques are already being applied to design riboswitches [5], but computational methods mostly aim for just a single target structure. Apart from the method already mentioned [22], the group behind the original RNAinverse method has also published a method for designing for multiple targets [13] essentially using the same adaptive random walk approach as in RNAinverse. The method of [22] also applies an adaptive walk, but in a deterministic, locally exhaustive manner. Adapting the GA already developed to handle multiple targets is very likely to provide a straight forward improvement on the current state-of-the-art for a relevant problem. However it will require addressing some new issues:

Litterature The basics don't change much, we are still dealing with predictive methods for RNA secondary structure and genetic algorithms. However, it will be a good idea to read [13] as it provides a concise and structured presentation of design for multiple targets covering motivation and the issues of objective and dependencies discussed below, as well as transition barrier.

Fitness function/Objective Usually the fitness function used in a GA is closely linked with the objective that we ultimately want to optimise, and more often than not they are identical. In particular when designing sequences for multiple targets, even if we have a single objective aim, it may be preferential to use a multi-objective fitness function. This would allow maintaining configurations in the GA population that show good affinity to one of the structures, even if it hasn't yet evolved an affinity for the other structure. This should be kept in mind when reading the following, even if the discussion will focus on the ultimate objective you should think about how this can be separated into multiple sub-objectives.

When designing for a single target, the ultimate objective is quite simple, namely to design a sequence that has the target structure as its predicted most stable structure. To make the design more robust to noise in the thermodynamic model and to have 'most stable' actually mean 'stable', we can add further requirements of high Boltzmann probabilities for all parts of the target structure. With two targets, a predicted structure can match at most one of them.

In [13] several examples of objectives are proposed, aimed at slightly different scenarios. Example 1 addresses the generic case, were we the aim is to design a sequence that has target structures Ω_1 and Ω_2 as stable structures with roughly similar stability. The objective function proposed for a sequence x is

$$\Xi(x) = E(x,\Omega_1) + E(x,\Omega_2) - 2G(x) + \xi\left(E(x,\Omega_1) - E(x,\Omega_2)\right)^2 \qquad (1)$$

with $E(x,\Omega_i)$ being the energy of folding x into target structure i and ξ a constant weighting the relative importance of overall stability vs. similarity in stability. $G(x)$ is the *ensemble free energy* of x, or the energy a structure on x should have to make the Boltzmann probability of the structure equal the probability under the uniform distribution; the entity $E(x,\Omega) - G(x)$

essentially measures how much more likely a structure Ω is for x than the average structure on a logarithmic scale. As discussed in [13, Example 2], computing the energies and $G(x)$ at different temperatures allows design for temperature dependent switch between the two target structures.

Maximising (1) with $\xi = 0$ will return a sequence with product of probabilities for structures Ω_1 and Ω_2 being maximum. However, it is not guaranteed that either will be the most stable. Including a term of $\max(E(x, \Omega_1), E(x, \Omega_2)) - \min_\Omega\{E(x, \Omega)\}$ would further bias the search towards sequences for which the target structures are the most stable structures.

As for the single target case, we can use the detailed Boltzmann probabilities of base pairs and unpaired positions in the target structures to further refine the objective. One thing to be aware of, though, is behaviour when the target structures have 'interchangeable parts'. Assume Ω_i is the set of base pairs in structure i and these can be partitioned into $\Omega_i = A_i \cup B_i$ such that $\Omega_{1,2} = A_1 \cup B_2$ and $\Omega_{2,1} = A_2 \cup B_1$ are both legal structures. Then large probabilities for the base pairs and unpaired positions in the target structures may be the result of high probabilities for structures $\Omega_{1,2}$ and $\Omega_{2,1}$ rather than for Ω_1 and Ω_2. Indeed, because of the additivity of the thermodynamic model, for every base pair $i \cdot j \in \Omega_1 \cap \Omega_2$ a sequence designed for both targets will also have the structure composed of everything outside $i \cdot j$ from one target structure and everything inside $i \cdot j$ from the other target structure as stable structure. If the insides and outsides are different between the target structures, this means there is no way to design sequences that only have the target structures as stable structures. This is not to say that it won't still be of interest to design sequences for which the target structures are among the stable structures, just don't despair if you cannot find a sequence where they are the only stable structures.

Dependencies The only dependencies we have in the single target case are between two positions forming a base pair in the target structure. Nucleotides for such positions should be chosen such that they form a canonical base pair. This affected design of initial population and mutations, where bases are not chosen independently but drawn from distributions over canonical base pairs, and even more so the crossover events, where we had to make sure that the two positions where taken from the same sequence.

With two target structures, the dependencies can include much larger sets of positions, potentially all positions could be dependent on each other. If $i \cdot j \in \Omega_1$ and $j \cdot k \in \Omega_2$, then the nucleotide in position j has to form a canonical base pair with both the nucleotide in position i and the nucleotide in position k. So the nucleotides in positions i and k cannot be chosen arbitrarily, but depend on each other. This dependency can of course extend even further, with more base pairs sharing a single position.

> If we define the dependency graph of structures Ω_1 and Ω_2 to have a node for every position, and an edge between two nodes if the corresponding positions form a base pair in either Ω_1 or Ω_2, then positions corresponding to nodes in the same connected component will be dependent. It is proved in [13] that there are always valid sequences, i.e sequences with canonical base pairs for all positions forming a base pair, if and only if the dependency graph is bipartite, and that with two target structures the dependency graph will always be bipartite. With three or more target structures the dependency graph may not be bipartite, e.g. we could have base pairs $i \cdot j$, $j \cdot k$, and $i \cdot k$ in three target structure resulting in a triangle in the dependency graph.
>
> However, with two target structures we always have a non-trivial set of valid sequences to search for a solution. To stay within the set of valid sequences, when mutating a position we may have to update all positions in its connected component (this is discussed in detail in [13]). For crossover events we need to make sure that all positions in the same connected component are taken from the same sequence. This is likely to make it too complicated and limiting to define crossovers by a position where we cross over and a position where we cross back. Bipartitioning of the set of connected components, possibly with a preference for keeping consecutive positions in the same partition, will be easier to implement and probably work better. The bipartitioning can, as for the single target case, be based on Boltzmann probabilities or similar measures of the local fitness of the two sequences used in the crossover.

The discussion above should cover the essentials going into updating the GA for multiple targets. There are a few other aspects that could be addressed, but which may be too expensive or too difficult to implement to be applied to each new sequence generated by the GA. In [13, Example 3] an objective including the height of the barrier separating the two target structures is proposed. This barrier is measured by the maximum energy of any structure on a path connecting the two targets by formation or breaking of a single base pair at a time, minimised over all paths. If we want sequences that allow a rapid switch between the two targets, such an objective makes a lot of sense. However, even with the heuristic computation suggested in [13] it may be too costly to compute for all sequences encountered. However, it could be used to rank a smaller set of the best sequences encountered when using the simpler objectives discussed above.

As already discussed, when a sequence doesn't fold to the target structure, it is usually better that it almost folds to the target structure than if it folds to something completely different. So we would want to design a sequence that apart from having the target structures as stable structures also has most of its other stable structures being similar to the target structures. This is somewhat captured by the marginal Boltzmann probabilities of base pairs and unpaired positions, but as previously stated these can be misleading when we have multiple targets. An alternative is to try to capture the full distribution over structures. A method based on clustering structures sampled from the Boltzmann distribution [10] is available from the Sfold server at `sfold.wadsworth.org/cgi-bin/srna.pl`. This determines centroid structures of the structures sampled, and we would prefer sequences where the centroids of the largest clusters are similar to the target structures. However, it is only available through this server, and not as a downloadable software, so again it may only be feasible to use it for final ranking of sequences identified using a simpler objective.

Week 6: Flexible Targets

With three or more targets, it may be impossible to find a valid sequence that could fold into all targets with all base paired positions forming canonical base pairs. Even with just a single target structure, the experience of previous attempts at inverse RNA folding has shown that some structures can be difficult to design sequences for. Currently inverse RNA folding methods require the design of a sequence with a specific length and with all structural elements fixed. In many cases the target we are designing for could be much more flexible, e.g. by allowing variation in loop sizes, variable helix lengths etc., as long as a given core substructure or motif is present. The aim of relaxing the design criteria would be to allow a higher fraction of successful designs, as well as possibly lower the time required to find a suitable sequence.

Though this model appears less constrained, this may actually make it harder to work with. With fully fixed targets, mutations could only involve changes of nucleotides, and crossovers where easily defined as there was a one-to-one correspondence between nucleotides in the two sequences recombining – choosing a subset of positions in one sequence would have to be accompanied by choosing the complement in the other sequence. With the more flexible formulation, mutations can involve insertions and deletions of nucleotides, and for crossovers we cannot just choose crossover points in the target structure and then project them onto the sequences.

Given the complexity of this task, it may not turn out to be feasible to complete within the time frame of the project. Moreover, it may be easier to deal with in an ad hoc manner, adding modifications to your existing GA and see what works, rather than starting with an abstract formulation of a fully developed approach. It is thus strongly advised that you only initiate this task once the first two tasks are more or less completed.

3 Data

The standard textual representation of RNA secondary structures is the dot-bracket notation, where a string is represented by a sequence of dots and matching brackets. If we use '.' to denote dots and '(' and ')' to denote brackets, a '.' denotes an unpaired position while a matching '(' and ')' denotes that the corresponding positions are forming a base pair. For example, the structure in

Fig. 1 is represented by the sequence

.((.(((.((....)))))...(((.(((...)))...)))))

Designing your software to read structure representations in this format, but allowing a flexible choice of what represents dots and brackets (for example, Rfam uses '$<$' and '$>$' for brackets), should make it easy to harvest targets from databases of known structures or to feed in randomly generated targets.

The two main repositories for RNA secondary structures are Rfam [15] at `rfam.sanger.ac.uk` and RNA STRAND [3] at `www.rnasoft.ca/strand`. In RNA STRAND, once you have found the structure of interest, you can simply choose to format it as dot-parentheses, and the structure is ready to cut and paste. In Rfam it is a bit more tricky obtaining the secondary structure in dot-bracket format. If you do not care for typing it in from a pictorial rendition you will have to download the `Rfam.seed.gz` file from the ftp section and search through the file. Additionally we also have a set of curated structures (downloaded from RNA STRAND) from a previous projects that can be used. You should aim to find the structures that have been used as tests for the existing methods. Beyond that it is more a matter of setting up test sets that explores variations in size and complexity. To generate random structures, it may be a good idea to use a model attempting to capture natural structures, e.g. the stochastic context-free grammar at `www.daimi.au.dk/~compbio/pfold/grammar.txt`.

4 Optional Extensions

This section consists of a few suggestions of other problems and issues that could be incorporated into the project work, but should not be considered part of the core project. Pursue at your own leisure.

4.1 Distribution Constraints

The INFO-RNA method tends to generate sequences with a high GC content, mainly due to the initial stage where a most stable sequence is generated. GC base pairs are in general more stable than other types of base pairs, so for all base paired positions INFO-RNA is highly likely to use only these two nucleotides. In [4] some analysis of GC content in the designed sequences is performed, but GC content is not actively considered in the design process.

Most recent methods do allow for position specific constraints, where in addition to folding into the target structure the designed sequence is also required in certain positions to have a particular nucleotide (or more generally restricted to a subset of the four nucleotides). However, it would be interesting to introduce more global constraints, only limiting the overall nucleotide or dinucleotide distribution (the frequency with which each nucleotide, or pair of consecutive nucleotides, are observed in the sequence) to design more 'natural' RNA sequences. For example, [28] showed that naturally occuring sequences could not be distinguished from random sequences with the same dinucleotide distribution simply by looking at minimum folding energy. So if we want to hide our designed RNA in genomic sequence it would be important to include dinucleotide distribution as additional constraint.

In its most simple form, this can be approached by biasing the local search towards making changes bringing the sequence closer to the desired distribution. More complicated methods where the updates keeps the right distribution by identifying sets of updates likely to improve the design but with effects on (di)nucleotide distribution cancelling out can also be pursued.

4.2 Efficient Updates

So far we have only been discussing the thermodynamic based computations of the Vienna package as black box methods, where we for each new sequence compute energies, Boltzmann probabilities, and predicted structures from scratch. The methods are based on recursions, where very similar sequences have identical values for large parts of these recursions. Rather than recomputing everything from scratch whenever we create a new sequence from one or more existing ones, it would be more efficient to just update the parts of the recursions that may have changed. In particular for sequences created by mutations, this could significantly reduce the time needed to assess the fitness of a new candidate. However, it will require a deeper understanding of the Vienna package, possibly even to the point of making some modifications to the source code.

Along a similar vein, rather than working with complete sequences in the GA, one could take inspiration from the decomposition approach of [4]. Each configuration in the population could be sequence fragments only covering part of the target structure, with the rest of the sequence assumed to fold to the target. At certain time points, the current fragments could be combined to full sequences. This would allow for faster fitness computations, but again would require a more complete understanding of the Vienna package. Also, it would additionally complicate the steps of the GA, requiring development of methods for choosing how to fragment the full structure.

4.3 de Bruijn Sequences for Initial Design

Though we will later see this may not be the case, one could imagine that the easiest way to design a sequence for a target structure is simply to generate it, rather than the current consensus of generating an initial sequence and then improve on this. A step towards this goal would be to develop better methods for generating the initial sequence. RNAinverse simply starts with a random sequence, while later methods attempt more intelligent initial designs. INFO-RNA finds the sequence resulting in the target structure having the lowest free energy, but this may result in other structures also having extremely low free energy. Hence, the target structure may only be stable compared to the unpaired structure but not compared to other structures. RNA-SSD does attempt to build an initial sequence avoiding repetitions of patterns and their reverse complements, but this appears to be done in an ad hoc manner.

A similar problem of avoiding strongly structured elements in mRNA has previously been considered in [8]. Here the aim is to make sure that no structure has low energy. The paper mentions de Bruijn sequences, but proceeds to develop heuristics based on alternative methods. Order k de Bruijn sequences are sequences that contain each length k string exactly once, e.g. an order 2 binary de Bruijn sequence could be 00110 or 01100. Order k de Bruijn sequences can easily be constructed from Hamiltonian paths in finite automata, see e.g. `en.wikipedia.org/wiki/De_Bruijn_sequence`. Here we are in a somewhat more complicated situation, as we also want to have base pairing segments be reverse complements of each other and otherwise avoid reverse complements. Still, de Bruijn sequences and other similar concepts could be explored for more rational designs of the initial sequence.

4.4 Robust Hardness Results

As alluded to earlier there is some evidence that inverse RNA folding is not an easy problem. In [24] it is proved **NP**-complete to determine if for a given hidden Markov model (HMM) and target path there exists a sequence with the target path as Viterbi path, i.e. most probable path. Stochastic context-free grammars (SCFGs) are generalisations of HMMs. As already mentioned, SCFGs can be used for RNA secondary structure prediction. So it would appear that determining whether there is a sequence that has a given derivation (i.e. secondary structure) as its most probable is hard.

However, the proof in [24] assumes that the HMM is part of the input. When predicting RNA secondary structures we usually use the same SCFG for all sequences. The question is whether it is possible to strengthen the proof to use a fixed HMM, or even better, a standard SCFG for RNA secondary structure prediction. Alternatively, one could explore the possibility of exploiting the fixed nature of the HMM/SCFG to design sequences with given target paths as viterbi paths. This would prove that the result in [24] does not apply to a fixed design problem, like inverse RNA folding, but only when the model we are designing for is part of the input.

References

[1] •R. Aguirre-Hernández, H. H. Hoos, and A. Condon. Computational RNA secondary structure design: empirical complexity and improved methods. *BMC Bioinformatics*, 8:34, 2007.

[2] E. S. Andersen. Prediction and design of DNA and RNA structures. *New Biotechnology*, 27(3):184–193, 2010.

[3] M. Andronescu, V. Bereg, H. H. Hoos, and A. Condon. RNA STRAND: The RNA secondary structure and statistical analysis database. *BMC Bioinformatics*, 9:340, 2008.

[4] •M. Andronescu, A. P. Fejes, F. Hutter, H. H. Hoos, and A. Condon. A new algorithm for RNA secondary structure design. *Journal of Molecular Biology*, 336:607–624, 2004.

[5] G. Bauer and B. Suess. Engineered riboswitches as novel tools in molecular biology. *Journal of Biotechnology*, 124(1):4–11, 2006.

[6] C. K. Biebricher and R. Luce. In vitro recombination and terminal elongation of RNA by Qβ replicase. *European Molecular Biology Organisation Journal*, 11(13):5129–5135, 1992.

[7] •A. Busch and R. Backofen. INFO-RNA – a fast approach to inverse RNA folding. *Bioinformatics*, 22(15):1823–1831, 2006.

[8] B. Cohen and S. Skiena. Natural selection and algorithmic design of mRNA. *Journal of Computational Biology*, 10(3–4):419–432, 2003.

[9] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multi-objective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[10] Y. Ding, C. Y. Chan, and C. E. Lawrence. RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA*, 11(8):1157–1166, 2005.

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

[12] ••S. R. Eddy. How do RNA folding algorithms work? *Nature Biotechnology*, 22:1457–1458, 2004.

[13] ••C. Flamm, I. L. Hofacker, S. Maurer-Stroh, P. F. Stadler, and M. Zehl. Design of multistable RNA molecules. *RNA*, 7(2):254–265, 2001.

[14] J. Z. M. Gao, L. Y. M. Li, and C. M. Reidys. Inverse folding of RNA pseudoknot structures. *Algorithms for Molecular Biology*, 5:27, 2010.

[15] P. P. Gardner, J. Daub, J. Tate, B. L. Moore, I. H. Osuch, S. Griffiths-Jones, R. D. Finn, E. P. Nawrocki, D. L. Kolbe, S. R. Eddy, and A. Bateman. Rfam: Wikipedia, clans and the “decimal” release. *Nucleic Acids Research*, 39:D141–D145, 2011.

[16] •I. L. Hofacker, W. Fontana, P. F. Stadler, L. S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie*, 125:167–188, 1994.

[17] ••R. Lyngsø. Lecture notes. RNA secondary structures, 2010.

[18] M. Mandal and R. R. Breaker. Gene regulation by riboswitches. *Nature Reviews Molecular Cell Biology*, 5:451–463, 2004.

[19] D. H. Mathews, J. Sabina, M. Zuker, and D. H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288:911–940, 1999.

[20] $^{\bullet\bullet}$J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–1119, 1990.

[21] J. S. Pedersen, G. Bejerano, A. Siepel, K. Rosenbloom, K. Lindblad-Toh, E. S. Lander, J. Kent, W. Miller, and D. Haussler. Identification and classification of conserved RNA secondary structures in the human genome. *PLoS Computational Biology*, 2(4):e33, 2006.

[22] E. I. Ramlan and K.-P. Zauner. Design of interacting multi-stable nucleic acids for molecular information processing. *Biosystems*, 105(1):14–24, 2011.

[23] P. W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.

[24] M. Schnall-Levin, L. Chindelevitch, and B. Berger. Inverting the Viterbi algorithm: an abstract framework for structure design. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 904–911, 2008.

[25] $^{\bullet\bullet}$A. Taneda. MODENA: a multi-objective RNA inverse folding. *Advances and Applications in Bioinformatics and Chemistry*, 4:1–12, 2011.

[26] I. Tinoco, O. C. Uhlenbeck, and M. D. Levine. Estimation of secondary structure in ribonucleic acids. *Nature*, 230:362–367, 1971.

[27] S. Washietl, I. L. Hofacker, and P. F. Stadler. Fast and reliable prediction of noncoding RNA. *Proceedings of the National Academy of Sciences of the United States of America*, 102(7):2454–59, 2005.

[28] C. Workman and A. Krogh. No evidence that mRNAs have lower folding free energies than random sequences with the same dinucleotide distribution. *Nucleic Acids Research*, 27(24):4816–4822, 1999.

[29] $^{\bullet}$J. N. Zadeh, B. R. Wolfe, and N. A. Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of Computational Chemistry*, 32(3):439–452, 2011.

References marked with $^{\bullet\bullet}$ are considered key references, while those marked with $^{\bullet}$ are considered important references.