Journal of Bioinformatics and Computational Biology

RNAiFold: A constraint programming algorithm for RNA inverse folding and molecular design

JUAN ANTONIO GARCIA-MARTIN
CNB-CSIC, Darwin 3, Campus Cantoblanco, 28049, Madrid, Spain.
ja.garcia@cnb.csic.es

PETER CLOTE
Biology Department, Boston College, 140 Commonwealth Avenue Chestnut Hill, Massachusetts, 02467, USA.
clote@bc.edu

IVAN DOTU
Biology Department, Boston College, 140 Commonwealth Avenue Chestnut Hill, Massachusetts, 02467, USA.
dotu@bc.edu

Synthetic biology is a rapidly emerging discipline, with long-term ramifications that range from single-molecule detection within cells to the creation of synthetic genomes and novel life forms. Truly phenomenal results have been obtained by pioneering groups – for instance, the combinatorial synthesis of genetic networks, genome synthesis using *BioBricks*, and hybridization chain reaction (HCR), in which stable DNA monomers assemble only upon exposure to a target DNA fragment, biomolecular self-assembly pathways, etc. Such work strongly suggests that nanotechnology and synthetic biology together seem poised to constitute the most transformative development of the 21st century.

In this paper we present a Constraint Programming (CP) approach to solve the RNA inverse folding problem. Given a target RNA secondary structure, we determine an RNA sequence which folds into the target structure; i.e. whose minimum free energy structure is the target structure. Our approach represents a step forward in RNA design – we produce the first complete RNA inverse folding approach which allows for the specification of a wide range of design constraints. We also introduce a Large Neighborhood Search approach which allows us to tackle larger instances at the cost of losing completeness, yet while retaining the advantages of meeting design constraints (motif, GC-content, etc.). Results demonstrate that our software, *RNAiFold*, performs as well or better than all state-of-the-art approaches; nevertheless, our approach is unique in terms of completeness, flexibility and the support of various design constraints. The algorithms presented in this paper are publicly available via the interactive webserver HTTP://BIOINFORMATICS.BC.EDU/CLOTELAB/RNAIFOLD; additionally, the source code can be downloaded from that site.

Keywords: Computational Biology; RNA inverse folding; Synthetic Biology; RNA molecular design.

1. Introduction

Much of the work in synthetic biology concerns what might be called "synthetic genomics", pertaining to synthetic regulation of genes [12] and the development of genomic building blocks, from which "parts" of a novel genome can be constructed [44]. In contrast to such work, in this paper, we instead consider RNA molecular design using computational methods from dynamic programming [7] and constraint programming [27], with subsequent experimental validation using in-line probing [46]. Ribonucleic acid molecules are currently of great interest to the biological community, due to their primordial role in the presumed RNA world [42], anterior to DNA and proteins, and especially due to the many surprising, recently discovered regulatory roles played by RNA [8,11,32,35]. As in the case of proteins, the function of RNA is often determined by its structure; consider, for instance, the regulation of genes and alternative splicing by allostery (riboswitches) [11,26] and the catalysis of enzymatic reactions (ribozymes) [18]. Due to the extensive study of RNA (secondary) structure, there is now software available for secondary structure prediction [28,36,37], motif discovery [23,58], structure alignment [24,39], riboswitch detection [10], precursor microRNA gene finders [53], non-coding RNA gene finders [49], etc. Due to the regulatory importance of RNA and the availability of such software, it is clear that some of the next important steps in sythetic biology will concern the computational design and experimental validation of RNA structures [1], as in the pioneering work of the lab of Niles Pierce [54].

1.1. *RNA inverse folding*

Given an RNA sequence, the *structure prediction* problem is to determine the *native structure* into which the sequence folds. Since the pioneering work of Anfinsen [3], it is widely accepted that the native structure of a given macromolecule can be identified with its minimum free energy (MFE) structure. The 'RNA inverse folding' problem is the inverse; i.e. given a target structure, determine an RNA sequence whose MFE structure is the target structure. There are several widely-used thermodynamics-based software suites, which compute the MFE structure of pseudoknot-free sequences in time that is cubic in the RNA sequence length – for instance, Vienna RNA Package `RNAfold` [25, 28], `mfold` [56], `UNAFOLD` [36], and `RNAstructure` [37], all of which implement the Zuker algorithm [57], though with slightly different energy parameters [38, 52]. Since RNA MFE secondary structure can be efficiently computed, while determination of the MFE pseudoknotted (hence, *a fortiori*, tertiary) structure is an NP-complete problem [34], in this paper we focus exclusively on the inverse folding problem for RNA secondary structures.

There is experimental evidence that RNA secondary structure forms independently of the tertiary structure [6]. From this data and newer NMR data [5], it is broadly believed that RNA folds in a hierarchical fashion [13], although there are exceptions [50, 51]. Since it appears that RNA secondary structure largely forms a scaffold for tertiary structure formation, any solution of the RNA secondary struc-

ture inverse folding problem is a major step towards functional RNA molecular design.

Several algorithms exist for the RNA inverse folding problem: `RNAinverse` [29], `RNA-SSD` [2], `INFO-RNA` [9], `MODENA` [47], `NUPACK-DESIGN` [54], `Inv` [21]. All of these algorithms can be classified as heuristic methods, which start with an initial sequence that is iteratively modified until it either folds into the target structure or some stopping criterion is reached.

The first approach found in the literature is `RNAinverse`, which forms part of the Vienna RNA Package [25, 29]. `RNAinverse` divides the given target structure S_0 into smaller subunits and attempts to find an RNA sequence by an *adaptive walk*, or greedy algorithm. Sequence positions are randomly mutated; mutations are accepted if the objective function improves. In this case, the objective function is the Hamming distance between the MFE secondary structure of the current sequence and the target structure S_0. `RNAinverse` can return the correct solution, an approximate solution, or no solution at all.

`RNA-SSD` [2] is a different and very efficient algorithm, which nevertheless, shares the same overall approach of applying a divide-and-conquer strategy by hierarchically decomposing the target structure. In comparison with `RNAinverse`, `RNA-SSD` uses a more sophisticated initialization procedure to choose an initial RNA sequence, and applies *stochastic local search* in place of of an adaptive walk. `RNA-SSD` is capable of finding the correct sequence for structures over one thousand nucleotides long.

The third approach is `INFO-RNA` [9]. Its main difference from previous approaches lies in the initialization step, which uses a dynamic programming algorithm to choose the sequence $s_1, \ldots, s_n$ that is compatible with the target structure S_0, having the lowest free energy. Although the free energy $E(s_1, \ldots, s_n; S_0)$ of target secondary structure S_0 on $s_1, \ldots, s_n$ is less than or equal to the free energy $E(s'_1, \ldots, s'_n; S_0)$ for all distinct sequences $s'_1, \ldots, s'_n$ that are compatible with S_0, this does *not* mean that the MFE structure of $s_1, \ldots, s_n$ is target structure S_0. `INFO-RNA` performs at least as well as `RNA-SSD`, and due to the initialization step, tends to yield RNA sequences, whose MFE structure has lower energy than sequences returned by other algorithms. Although this might seem to be a desirable feature, the solutions returned by `INFO-RNA` have high GC content and tend to have little resemblance with biologically active RNA, found in databases such as Rfam [22].

The fourth approach, `MODENA` [47], differs considerably from other inverse folding approaches, since it relies on a *multi-objective optimization* algorithm. `MODENA` uses the well-known NSGA2 [15] genetic algorithm to find solutions in the set of weak Pareto optimal solutions with respect to two optimization functions: structure stability (energy of the MFE structure of the proposed sequence) and structure similarity (distance between the MFE structure for the candidate sequence and the target structure). `MODENA` compares favorably to `INFO-RNA` and `RNAinverse` when benchmarked on a data set from Rfam [22].

NUPACK-DESIGN [54], is a remarkable, pioneering project of the Niles Pierce Lab, to design RNA molecules that have subsequently been synthesized and tested for folding properties, both *in vitro* and *in vivo*. NUPACK-DESIGN employs a similar approach to that of RNA-SSD, but, in this case, instead of finding sequences whose MFE structure is the given target structure, NUPACK-DESIGN attempts to find sequences having minimal *ensemble defect* [16] (See Appendix A and B).

Finally, the algorithm Inv [21] uses a stochastic local search routine to determine a sequence whose minimum free energy *pseudoknotted* structure is a given target *3-noncrossing* RNA structure. Here, a 3-noncrossing structure is a (possibly pseudoknotted) structure, in which no three base pairs mutually cross each other. Inv relies on the dynamic programming (exponential time) minimum free energy structure prediction algorithm for 3-noncrossing structures [30], and the fact that each 3-noncrossing RNA structure has a unique loop-decomposition.

In this paper we present two algorithms to solve the inverse folding problem for RNA secondary structures. The first is a Constraint Programming (CP) implementation which performs surprisingly well, compared to the previously mentioned approaches. However, CP performs an exhaustive exploration of the search space which can lead, in some cases especially when the structures are large and complex, to a prohibitive inverse folding time. For this reason, we have also developed a Large Neighborhood Search (LNS) method which builds on the underlying CP framework, which achieves better results for larger structures. LNS can also be used when completeness is not required; i.e. when it is not necessary to prove that no solution exists, in the case that none does exist.

2. Methods

As previously mentioned, our algorithm is based on a Constraint Programming formulation of the RNA inverse folding problem. Constraint programming (CP) has become one of the main methodologies for solving hard combinatorial optimization problems. Its salient features are its rich modeling language and its computational model based on *branch and prune*. At the modeling level, CP models a complex application in terms of decision variables, domains which specify the possible values for the variables, and constraints which capture its combinatorial substructures, giving the underlying solver significant information on the application structure. For instance, CP solvers feature global constraints such as *alldifferent*$(x_1,\ldots,x_n)$, which specifies that the variables $x_1,\ldots,x_n$ must be given different values. This contrasts with frameworks such as mixed-integer programs where all the constraints are linear.

Our algorithm is developed using the COMET framework [27] and RNAfold (from the Vienna RNA Package [25]) adapted as a plug-in with COMET. The programming language, COMET, features a very efficient CP engine along with several global constraints that are key for the efficiency of our approach.

When implementing a program using CP, we need to determine two different

aspects: modeling (variables, domains and constraints) and search (variable and value heuristics). As mentioned in the introduction, we have developed two different algorithms, using CP and LNS. The modeling part is common to both and only the search part differs. We describe each of these in the following subsections.

Modeling

The RNA inverse folding problem can be stated as follows: given a secondary structure S_0, presented as a dot-bracket expression of length n, find the RNA sequence (i.e. a word in the alphabet $\{A, G, C, U\}$) whose minimum free energy (MFE) structure is S_0. In our case, MFE structure is predicted by `RNAfold`, a tool from the Vienna RNA Package [25]. The secondary structure can be alternatively viewed as a set of canonical base pairs ($\{GC, CG, AU, UA, GU, UG\}$) and a set of unpaired positions.

Variables and Domains

The first modeling choice corresponds to the variables that define the problem and the values they can take (i.e. variable domains). In order to boost efficiency and create a framework that easily permits the addition of sequence constraints, we define several sets of variables.

- X: A set of variables corresponding to the nucleotides of the solution sequence $X = \{x_1, x_2, ..., x_n\}$ (corresponding to the 4 different nucleotides).
- UP: A set of variables, $UP = \{up_1, up_2, ..., up_k\}$, corresponding only to unpaired nucleotides in the target structure S_0, where k is the number of unpaired positions in S_0.
- BP: A set of variables, $BP = \{bp_1, bp_2, ..., bp_\ell\}$, corresponding to every base pair in S_0, where ℓ is the number of base pairs in S_0. Note that ℓ base pairs correspond to $2 \cdot \ell$ nucleotides in the sequence, the specific canonical base pairs found in RNA structures.
- BPT: A set of variables, $BPT = \{bpt_1, bpt_2, ..., bpt_\ell\}$, corresponding to every base pair in S_0 and indicating the type of the base pair ($\{GC, AU, GU\}$).
- GC: A set of boolean variables, $GC = \{gc_1, gc_2, ..., gc_n\}$, for each position in the sequence representing whether it is assigned to a G or a C or not.

It is important to distinguish between *search* variables and *auxiliary* variables. *Search* variables are the ones on which the search will focus, i.e., the ones that will be explicitly assigned a value. *Auxiliary* variables help simplify constraint declarations and/or heuristics, and they need to be unequivocally determined via *channeling* constraints [a]. In our approach, UP and BP are search variables, while X, BPT

[a]In Constraint Programming, *channeling constraint* refers to a type of constraint that links two different modelings of the same problem and ensures that the solutions for both modelings are consistent with one another.

and GC are auxiliary variables.

A straightforward approach would be to choose letters among $\{A, G, C, U\}$, and pairs of letters among $\{GC, CG, AU, UA, GU, UG\}$, as domains for X and BP, respectively. However, this is not only more computationally costly, but also the correspondence between sequence variables X and base pairs and unpaired variables becomes very complex. For this reason we choose to use an integer representation for all the domain values.

Going a step further, we choose integers corresponding to the marks in an optimal *Golomb ruler* [4, 45] of size 5, for the domain values of X ($\{A, G, C, U\}$). A Golomb ruler is a ruler with marks placed at certain integer positions such that all the pairwise differences between marks are different. An optimal Golomb ruler, given a certain number of marks, is a Golomb ruler of minimum length. For 5 marks, the optimal Golomb ruler has marks in positions $\{0, 1, 3, 7, 12\}$. Excluding 0 which is always the first mark by definition, the domains of all the variables are the following.

- $dom(X) = \{1, 3, 7, 12\}$ corresponding to $\{G, A, C, U\}$.
- $dom(UP) = \{1, 3, 7, 12\}$ corresponding to $\{G, A, C, U\}$.
- $dom(BP) = \{-11, -9, -6, 6, 9, 11\}$ corresponding to $\{GU, AU, GC, CG, UA, UG\}$.
- $dom(BPT) = \{36, 81, 121\}$ corresponding to $\{GC, AU, GU\}$.
- $dom(GC) = \{0, 1\}$.

Note that (as will be formally described below) each base pair value is the difference of its sequence values, and each base pair type is the squared difference of its sequence values. This allows for a direct implementation of certain constraints (see below) which, in turn, represents a great speed-up when checking their consistency and performing their propagation.

Additionally, we maintain the following dictionaries.

- $BPstart$. Given a base pair, the position of its first nucleotide in S_0.
- $BPend$. Given a base pair, the position of its last nucleotide in S_0.
- $UPdict$. Given an unpaired variable, its corresponding position in S_0.

Constraints

There are three types of constraints in our approach: *channeling* constraints, *structural* constraints and *sequence* constraints. The first two types of constraints are always used, while the last type of constraint is optional. Sequence constraints are used to specify biologically important motifs, GC-content, and other biologically relevant features desired for RNA molecular design. These are not to be confused with structural constraints, which enforce that the sequence folds into the target structure.

Channeling constraints allow us to unequivocally determine the value of all

auxiliary variables from the search variables. They are the following.

- For each base pair i, $BP_i := x_{BPstart(i)} - x_{BPend(i)}$.
- For each base pair i, $BPT_i := (x_{BPstart(i)} - x_{BPend(i)})^2$.
- For each unpaired position i, $UP_i := x_{UPdict(i)}$.
- For each position i, $GC_i := (x_i == 1 \wedge x_i == 7)$.

Structural constraints will ensure that the sequence folds into the target structure S_0. In order to minimize computational cost, we break down the structure hierarchically, as previously done in most prior methods, `RNAinverse` [29], `RNA-SSD` [2]. Each constraint will ensure that a certain substructure is the minimum free energy structure for the corresponding subsequence. First, we create a tree-like decomposition T_1, where nodes correspond to substructures; from this, we next create a *reduced* tree-like decomposition T_2, obtained by repeatedly merging adjacent nodes of T_1 together. As explained below, adjacent nodes u, v of T_1 are merged when it happens that the substructure S_u corresponding to node u is energetically unstable (free energy of S_u is positive), while the substructure S_{uv} is energetically stable (free energy of S_{uv} is negative). Here if S_u [resp. S_v] represent the substructures corresponding to adjacent nodes u, v of T_1, then uv is the substructure corresponding to the concatenation of S_u with S_v. This operation is iterated, thus yielding a *reduced* tree T_2, with the property that the substructure corresponding to each node of T_2 has negative free energy. Finally, constraints will be generated to correspond to the nodes of reduced tree T_2, as depicted in the right panel of Figure 2.

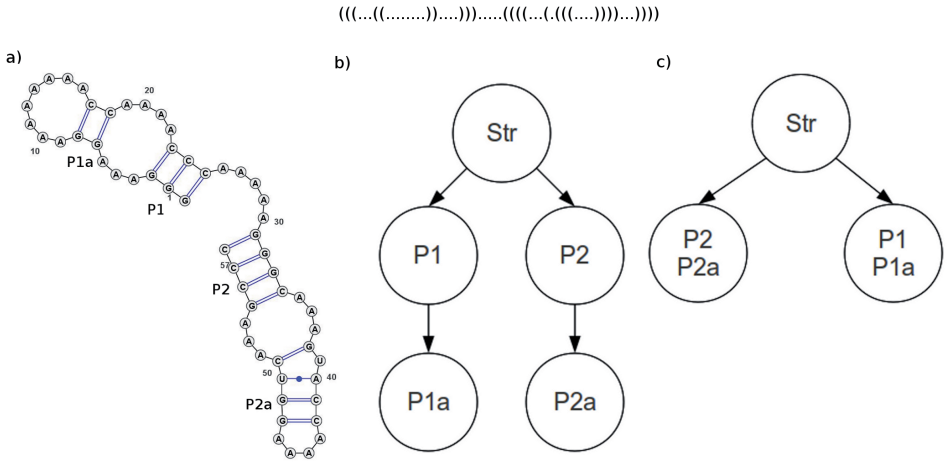
The structure decomposition tree T_1 is defined as follows:

- The root of the tree is a node, corresponding to the (entire) target structure S_0.
- Recursively, create a node for each helix in the target structure. As shown in Figure 2 for the example target secondary structure S_0 given by

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

  the root of T_2 (corresponding to S_0) has two children, corresponding to helices $P1, P2$. For each node/substructure, recursively perform the same decomposition where a node is considered a parent node for the helices into which it can be decomposed. In our illustrative example, $P1$ [resp. $P2$] has child $P1a$ [resp. $P2a$]. If the currently considered node/helix $u \in T_1$ leads to a multiloop (also called multi-way junction), then u has children $v_1, \ldots, v_{k-1}$, corresponding to the $k-1$ remaining helices that are incident to the multiloop. If the currently considered node/helix $u \in T_1$ leads to an internal loop or bulge of size greater than 2, then u has a single child v, corresponding to the remainder of the stem after the internal loop or bulge.
- Leaves of the tree correspond to terminal helices, i.e., stem loops, as depicted by $P1a$ and $P2a$ in Figure 2.

Fig. 1. RNA structure and its tree-like decomposition.



(Left) Target RNA secondary structure S_0. *(Middle)* Structure decomposition tree T_1. *(Right)* Reduced tructure decomposition tree T_2.

Formally, for the purpose of our decomposition, a *helix* is a set of *consecutive* base pairs, where consecutive is loosely defined as to allow, within a helix, bulges of size at most 2, and internal loops of sizes at most (1×1), (2×1), (1×2), (2×2).[b]

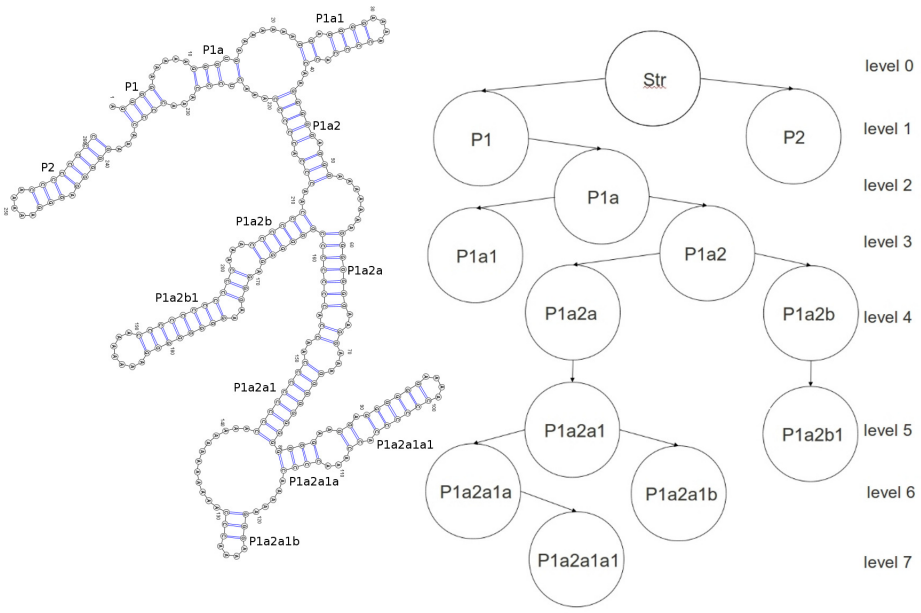
After computing the structure decomposition tree T_1, we subsequently perform a recursive merge operation, proceeding from leaves to the root. Initially T_2 is defined to be T_1. We recursively merge adjacent nodes of T_2 until no further merge operations are needed. This produces the final *reduced tree* T_2. Two adjacent nodes of T_2 are merged together, if either of the following holds.

- The stacking free energy of the stem (assuming that all base pairs in the stem are GC pairs) does not exceed, in absolute value, the free energy of the apical loop; i.e. the stem-loop structure is not energetically favorable, assuming base pairs are realized by GC pairs. This happens, for instance, in the stem-loop `((.......))`.
- The outermost or external base pair of the stem is separated from the rest of the stem by a bulge or internal loop of any size.

As mentioned, the merge operation is performed recursively from leaves to root. The reduction of tree T_1 to T_2 is very important, since certain nodes/substructures

[b]An internal loop of size $(n \times m)$ is enclosed by base pairs (i, j) and $(i+n+1, j-m-1)$, where positions $i+1, \ldots, i+n$ and $j-m, j-m+1, \ldots, j-1$ are unpaired.

Fig. 2. RNA structure and its tree-like decomposition.



(Left) RNA structure for Rhizobiaceae group bacterium NR64, with EMBL accession number Z83250. Image produced using `VARNA` [14]. *(Right)* Tree decomposition of helices for Z83250.

u of T_1 might be energetically unstable, meaning that no sequence would fold into the structure corresponding to u. Figure 2 depicts the reduction procedure, where, given the target structure S_0 (left panel)

```
(((...((........))....)))....((((....(.(((....)))))...))))
```

we obtain the structure decomposition tree T_1 (middle panel), and after the merge procedure, the reduced tree decomposition T_2 (right panel). Finally, each node in the reduced tree T_2 corresponds to a structural constraint that is considered by our algorithm `RNAiFold`. Figure 2 depicts the structure decomposition tree T_1 for the Rhizobiaceae group bacterium NR64 RNA, with EMBL accession number Z83250.

We also maintain a global structural constraint, which ensures that the whole sequence folds into the target structure S_0. However, note that this constraint will never be checked until all the other constraints are met, for a candidate sequence.

Finally, sequence constraints are optional constraints that allow us to further specify desired features of a solution sequence. They are the following:

- Lower and upper bound on the number of base pairs of each type. Given a list of lower bounds *lbs* and a list of upper bounds *ubs* for each type of base pair ($\{GC, AU, GU\}$), we can use a global constraint within COMET: $cardinality(lbs, BPT, ubs)$.
- Maximum number of consecutive nucleotides of each type. Ensuring that the number of nucleotides of a particular type is bound by a specified maximum *maxcs*, can be realized by the following global constraint: $stretch(0, X, maxcs)$.
- Lower and upper bound on GC content. This is handled in an analogous manner, as in the base pair types: $cardinality(lb, GC, ub)$.

CP Search

When implementing the search part of a Constraint Programming problem, we need to focus in the order in which variables will be assigned and on the order in which values will be assigned to the variables. Our CP algorithm is complete, meaning that it explores the search space exhaustively. This implies that, given sufficient time, our CP algorithm will either return a solution or prove that none exists. Moreover, we can as well return all the solutions, i. e., all the sequences that fold into the given target structure. Variable and value ordering heuristics give us the order in which we traverse the search space.

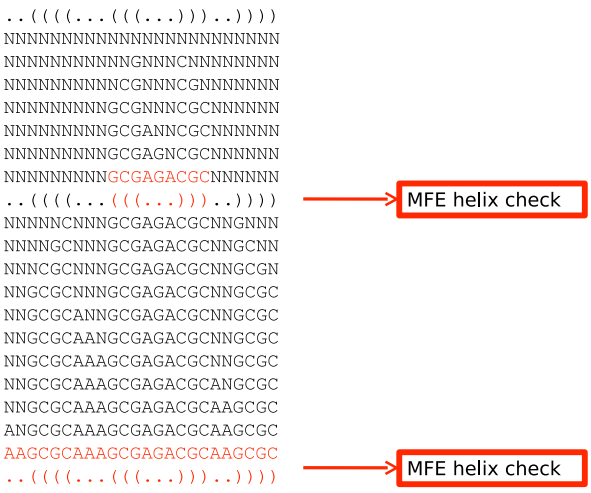
Variable ordering

Our ordering is specified in a stepwise manner:

(1) Variables are first grouped according to the structural constraint to which they belong. Structural constraints are ordered by levels, from top (parent) to bottom (child), as shown in the example tree in the right panel of Figure 2. Note that the rest of constraints are not involved in variable ordering, also, they are checked and propagated after any individual variable assignment.
(2) Within each constraint, BP variables are assigned first; subsequently, UP variables are assigned.
(3) Within BP, variables are assigned from inside to outside of the helix.
(4) Within UP, variables are grouped in consecutive runs; runs are ordered from large to small.
(5) Within a UP run, variables are assigned from left to right.

To illustrate this ordering we have extracted the intermediate variable assignments for a toy example, which is depicted in Figure 3. We name this heuristic *levels bottom-up*.

Fig. 3. Trace of a toy example to illustrate variable ordering.



In red full helix assignments corresponding to constraint check.

Value ordering

BP values are assigned the most stable value. If it is the start or the end of a helix, the order is the following: $\{GC, CG, AU, UA, GU, UG\}$. Otherwise, the order is determined by the stacking energy contribution given the previously assigned base pair. Additionally, we introduce a random component that is added to the energy contribution, thus ensuring different values depending on the random seed. This random component is a parameter of our algorithm, but all the results presented in the following sections use an additional random energy between 0 and $2Kcal/mol$.

UP values are assigned in the following order: $\{A, U, G, C\}$.

Note that randomizing the heuristic does not compromise completeness, it only entails that different runs of the algorithm will (potentially) yield different solutions, since the order in which the search space is visited would be different.

Parallelism

COMET allows for parallelization of solvers. A given number of solvers are run in parallel; if or when a solver finds a solution, all the other solvers halt. Given the fact that our variable ordering contains a random component, different parallel runs are bound to explore the search space in a different fashion, and, thus, they can find a solution within a different run time. We take advantage of this feature and run our algorithm with 4 parallel solvers. The parallel implementation of COMET en-

sures that completeness is maintained by sharing information among all the parallel solvers.

The type of parallelism we use constitutes no shared memory. It is basically the replacement of a *for* loop for a parallelized version named *parall* in COMET. Operationally, the *parall* creates a thread to execute the loop body for each iteration. These threads are joined after the loop, i.e., the instruction following the loop is only executed after all threads completed their execution. Each thread has its native runtime control block and stack, as well as equivalent data structures for the COMET runtime.

LNS

Large Neighborhood Search is a meta heuristic that attempts to find a high quality solution by iteratively changing a candidate (or tentative) solution. As opposed to other methods where differences between tentative solutions between two successive iterations is minimal, LNS fixes a small part of the tentative solution and explores (exhaustively if possible) the remaining, unfixed positions. This explains the origin of the name, 'Large Neighborhood Search'.

COMET supports a straightforward implementation of LNS, where we reuse the program design and constraints from the CP implementation, while we add a 'restart' component. This restart component will fix some of the variables to their current values and will unassign the remaining variables. Thus, we only need to specify when to restart and what to do when we restart.

First of all, we choose to restart after an amount of time, which is proportional to the length of the target structure. Second, we choose to fix only BP variables that are correct with respect to the target structure. However, this can be problematic. Indeed, imagine that we restart and we fix a single helix in the tree which was not solved during the search and thus, the LNS algorithm never attempted to solve substructures of the parent or ancestor in the decomposition tree. If we fix the base pairs, given that we have a fixed value ordering for UP, the search will explore exactly the same space, and it will restart at the same point, with no improvement. For this reason, we introduce two additional features:

- a random component for the value ordering of UP variables;
- a *hard* restart – if, after a certain number of restarts (which we fix at 5), we have always fixed the exact same set of variables, we start from scratch; i.e., we do not fix any variables.

Additionally, we have added a slightly modified variable ordering heuristic, which we call *leaves to root*, in which leaf nodes in the decomposition tree are always visited before any interior nodes, regardless of the level. For instance (and opposed to *levels bottom-up* heuristic), in Figure 2, node *P1a2b1* in level 5 will be assigned prior to node *P1a2a1a* in level 6.

3. Results

In this section we present a comparison of our approach against the approaches mentioned in the Introduction, excluding `Inv`, which concerns 3-noncrossing structures. It should be mentioned that different sets of structures are used in benchmarking studies for different papers [2], [9], [47], [54]. Since we believe that the benchmarking set introduced by Taneda et al. [47] is the most unbiased and biologically relevant set of target structures, we believe the benchmarking results for this data set to be the most representative for the behavior of `RNAiFold` (see Tables 1 and 5). Nevertheless, in the remaining Tables 2 and 3, we benchmark `RNAiFold` against all the other data sets considered in the literature.

The benchmarking set of target secondary structures of Taneda et al. is built in the following manner.

- Download the seed alignment for various families from Rfam [22].
- Select the largest sequence in each seed alignment.
- Extract the annotated structure for the given sequence.
- Remove pseudoknotted pairs.

Since the Rfam database is modified and updated over time, to permit accurate benchmarking, we used the same set of Rfam structures used in the benchmarking from [47].

In order to compare with other approaches (mostly heuristic) we run our algorithms for each instance a certain number of times (usually 50), and report the number of times where the algorithm was able to return a solution, and the average time in which it did. For our LNS algorithm, which is heuristic, this is clearly understood. For our CP algorithm, even though it is complete, since we have added a random component to the variable (and value) ordering heuristic, different runs will explore the search space in a different order, and, thus, yield different results.

All benchmarking was carried out on an Intel Core i72630QM using 4 cores (2GHz, 16GB memory, Linux Ubuntu 10.4), with a cutoff time of 10 minutes for all runs and for all algorithms. MODENA results are reported as in [47], where there is only 1 run with a population size of equal to the number of runs of the rest of the algorithms. Reported time is total time (in seconds) for MODENA to return the final population. All other times are reported also in seconds and are the average over all runs that returned a solution, where a *dash* ('−') corresponds to no solution found and thus no average time available. For all tables, best results are shown in bold face. Note that the algorithm that solves more runs might not be the fastest, since the average time is computed only over solved runs.

`INFO-RNA` 2.0 (newest version) was run, while allowing 0 mismatches in the final sequence (-n 0). `MODENA` was run with the maximum number of iterations allowed (9999) and a population equal to the number of runs. `RNA-SSD` code was modified to avoid premature termination due to the maximum number of tries and keep trying until a solution is found. `RNAinverse` was run with -R 1 (search until one solution

is found).

We will discuss the results separately for CP and LNS.

CP results

Table 1. **Rfam CP Results.**

Parameters		CP		INFO-RNA		MODENA		RNA-SSD		RNAinverse	
RF id	n	sol	time	sol	time	sol	time	sol	time	sol	time
RF00001.121	117	38	21.5	**50**	**0.0**	6	36.8	22	1.0	41	233.1
RF00002.2	151	**44**	29.5	4	62.6	20	39.4	6	**12.2**	0	-
RF00003.94	161	0	-	1	72.1	**29**	**70.2**	0	-	0	-
RF00004.126	193	**50**	1.5	**50**	**0.1**	34	52.9	**50**	2.0	**50**	48.3
RF00005.1	74	**50**	0.2	**50**	**0.0**	33	12.4	**50**	0.1	**50**	0.1
RF00006.1	89	**50**	0.3	**50**	**0.0**	37	15.1	**50**	0.6	**50**	4.3
RF00007.20	154	**50**	5.6	**50**	**0.0**	34	44.4	**50**	1.1	**50**	12.4
RF00008.11	54	**50**	0.1	**50**	**0.0**	26	8.7	**50**	**0.0**	**50**	**0.0**
RF00009.115	348	**48**	**20.8**	0	-	29	214.1	26	48.2	0	-
RF00010.253	357	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-
RF00011.18	382	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-
RF00012.15	215	**50**	**2.7**	15	25.0	27	64.5	28	28.8	1	139.4
RF00013.139	185	**50**	1.6	**50**	**0.8**	12	51.5	49	2.8	**50**	19.8
RF00014.2	87	**50**	0.3	**50**	**0.0**	33	17.5	49	0.1	**50**	**0.0**
RF00015.101	140	49	1.3	**50**	**0.2**	38	29.1	40	0.6	**50**	52.4
RF00016.15	129	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-
RF00017.90	301	**50**	19.3	**50**	**0.0**	28	208.1	50	7.0	**50**	10.0
RF00018.2	360	**47**	**12.1**	1	697.0	28	331.5	0	-	0	-
RF00019.115	83	**50**	0.2	**50**	**0.0**	32	14.9	**50**	0.2	**50**	0.3
RF00020.107	119	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-
RF00021.10	118	**50**	0.3	**50**	**0.0**	37	27.8	49	0.2	**50**	0.2
RF00022.1	148	**50**	0.7	**50**	**0.0**	38	32.6	24	0.9	35	225.5
RF00024.16	451	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-
RF00025.12	210	**50**	**1.4**	9	47.9	33	54.2	29	2.9	0	-
RF00026.1	102	**50**	**0.4**	33	5.5	38	15.2	50	1.4	44	173.2
RF00027.7	79	**50**	0.1	**50**	**0.0**	32	17.4	**50**	0.1	**50**	0.4
RF00028.1	344	**39**	**6.2**	0	-	0	-	4	71.2	0	-
RF00029.107	73	**50**	0.3	**50**	**0.0**	37	10.4	**50**	0.2	**50**	0.3
RF00030.30	340	**46**	**6.8**	1	57.3	22	186.8	34	39.3	0	-
sum	-	**1111**	**133.2**	813	271.5	683	1555.5	860	220.9	771	919.7
avg	-	**38.3**	**5.7**	28.0	12.9	23.6	67.6	29.7	10.0	26.6	54.1

Summary of the experimental results. The first column is the Rfam identifier, the second column is the length of the structure. The rest of the columns are: (sol) number of runs where the algorithm returned a solution out of 50 executions (for MODENA is the number of correct individuals in the final population), and (time) the average time (in seconds) to find a solution (over the runs that did return a solution), for all the algorithms tested. The last two rows show sum and average values.

Tables 1,2,3 show the comparison results for our method against `MODENA`, `RNA-SSD`, `INFO-RNA` and `RNAinverse`. According to results from Table 1, we see that CP is far superior to other methods. There are more runs in which the algorithm returns a solution, and it is only slightly slower than `INFO-RNA` on some of the easiest structures (those that are always solved in less than 1 second). Note that times are averaged over runs that returned a solution, and thus, speed comparison with methods that returned less solutions is not completely fair. In any case, our method is faster overall.

Tables 2 and 3 show a comparison over two sets of biologically relevant structures from [2]. In these cases, CP shows comparable performance, and it is only inferior for some of the larger structures, especially in the set from Table 2, where it is possible

Table 2. **RNA-SSD set 1 CP Results.**

Parameters			CP		INFO-RNA		MODENA		RNA-SSD		RNAinverse	
RF id	n	runs	sol	time	sol	time	sol	time	sol	time	sol	time
Z83250	260	50	**50**	2.6	**50**	**0.0**	14	125.6	**50**	2.1	43	213.9
L11935	264	50	**50**	5.0	**50**	**0.0**	16	121.8	**50**	1.1	**50**	109.1
LIU92530	289	50	**50**	10.0	**50**	**0.0**	0	-	1	354.9	17	351.9
U84629	299	50	**50**	5.5	**50**	**0.0**	9	153.1	35	6.4	1	554.6
AF107506	337	50	**50**	9.5	**50**	**0.0**	28	218.2	49	6.6	7	347.6
AF106618	350	50	**50**	20.8	**50**	**0.0**	5	131.9	50	2.1	38	265
AJ011149	376	50	47	140.5	**49**	**0.0**	0	-	26	62.5	1	463.5
S70838	389	50	**50**	27.9	**50**	**0.0**	3	275.4	47	7.1	10	295.2
U63350	418	25	**25**	11.7	**25**	**1.2**	17	191.3	21	2.8	6	346.3
AF141485	473	25	17	51.4	**25**	**0.1**	13	266.6	22	65.1	0	-
U81771	491	25	**25**	28.8	**25**	**0.1**	10	221.6	23	26.2	0	-
AJ130779	506	25	22	70.1	**25**	**0.1**	12	227	23	11	2	507.2
AF096836	646	25	**25**	48.2	24	**0.3**	4	440.4	18	15.5	0	-
X61771	659	25	8	67.0	**18**	**0.3**	0	-	18	129.6	0	-
AJ236455	751	25	0	-	0	-	0	-	**19**	**39.2**	0	-
AJ132572	780	25	23	158.2	**24**	**0.3**	0	-	20	30	0	-
AB015827	856	10	4	245.2	**10**	**5.2**	0	-	9	49.7	0	-
D38777	858	10	1	173.3	**10**	**1.5**	0	-	**10**	17.3	0	-
AF029195	1053	10	7	321.0	**10**	**2.7**	0	-	**10**	42.2	0	-
X81949	1200	10	**6**	197.1	5	**15.7**	0	-	6	48.5	0	-
AJ133622	1296	10	0	-	**8**	**7.8**	0	-	4	128.6	0	-
AF056938	1398	10	5	477.9	**10**	**2.5**	4	319.7	7	58.5	0	-
X99676	1442	10	2	569.2	**8**	**9.8**	1	510.1	7	156.5	0	-
L77117	1475	10	0	-	**5**	**20.4**	0	-	**5**	90.4	0	-
sum	-	680	567	2640.9	**631**	**68**	136	3202.7	530	1353.9	175	3454.3
avg	-	28.3	23.6	125.8	**26.3**	**3.0**	5.7	246.4	22.1	56.4	7.3	345.4

Summary of the experimental results. The first column is the Rfam identifier, the second column is the length of the structure and the third the number of runs executed for all the algorithms. The rest of the columns are: (sol) number of runs where the algorithm returned a solution out of *runs* (for MODENA is the number of correct individuals in the final population), and (time) the average time (in seconds) to find a solution (over the runs that did return a solution), for all the algorithms tested. The last two rows show sum and average values.

that, given a larger cutoff time, CP would find solutions as well. The newest version of `INFO-RNA` performs extremely well, especially in the benchmarks of Table 2. Our algorithm is slightly slower than both `RNA-SSD` and `INFO-RNA`.

Table 4 shows a summary of all the datasets. Our algorithm finds, overall, a solution in a greater amount of runs; solves a similar amount of structures when compared to `RNA-SSD` and `INFO-RNA`, and it is only slightly slower than these two methods.

We do not claim our approach is faster than previous methods, but it solves more instances more often and it is at least comparable in speed, which can be counterintuitive given the exhaustive nature of our CP approach. We show that the addition of a large number of potentially relevant biological constraints does not jeopardize speed. However, times reported here correspond to finding one solution; finding all solutions or proving that none exists will, of course, require a greater amount of time.

Note that, given the stochastic nature of our algorithm (to prevent helices from being composed entirely of GC pairs), we run `RNAiFold` several times and provide statistics on these multiple runs for comparison. Even though in the long run, each execution of `RNAiFold` will either return a solution or prove that none exists, the speed with which it can find a solution is influenced by the stochastic nature of our

Table 3. **RNA-SSD set 2 CP Results.**

Parameters		CP		INFO-RNA		MODENA		RNA-SSD		RNAinverse	
#	n	sol	time	sol	time	sol	time	sol	time	sol	time
1	100	**100**	0.1	**100**	**0.0**	77	19.3	**100**	0.1	**100**	0.1
2	100	**100**	**0.0**	**100**	**0.0**	73	26.2	**100**	0.1	**100**	0.1
3	100	**100**	2.7	**100**	**0.0**	75	69.4	98	1.5	**100**	4.1
4	100	**100**	0.7	**100**	**0.0**	82	104.5	**100**	0.9	**100**	4.1
5	100	**100**	**0.7**	2	165.7	53	245.7	0	-	2	407.9
6	100	**99**	6.2	93	0.8	62	192.2	**100**	**0.0**	3	362
7	100	**100**	9.8	84	**0.8**	68	405.9	64	12.8	4	254.6
8	100	**99**	**7.0**	22	19.5	57	421.1	76	48.4	0	-
9	100	**0**	-	**0**	-	**0**	-	**0**	-	**0**	-
10	100	92	32.9	**100**	**0.1**	57	397.2	99	6.9	13	287.6
sum	-	**890**	**60.0**	701	186.9	604	1881.5	737	70.7	422	1320.5
avg	-	**89**	**6.7**	70.1	20.8	60.4	209.1	73.7	8.8	42.2	165.1

Description	
1	Minimal catalytic domains of the hairpin ribozyme satellite RNA of the tobacco ringspot virus (Figure 1a) (Fedor, 2000)
2	U3 snoRNA 5'-domain from Chlamydomonas reinhardtii, in vivo probing (Figure 6B) (Antal et al., 2000)
3	H. marismortui 5S rRNA (Figure 2) (Szymanski et al., 2002)
4	VS Ribozyme from Neurospora mitochondria (Figure 1A) (Lafontaine et al., 2001)
5	R180 ribozyme (Figure 2B) (Sun et al., 2002)
6	XS1 ribozyme, Bacillus subtilis P RNA-based ribozyme (Figure 2A) (Mobley and Pan, 1999)
7	Homo Sapiens RNase P RNA (Figure 4) (Pitulle et al., 1998)
8	S20 mRNA from E.coli (Figure 2) (Mackie, 1992)
9	Halobacterium cutirubrum RNAse P RNA (Figure 2) (Haas et al., 1990)
10	Group II intron ribozyme D135 from ai5g (Figure 5) (Swisher et al., 2001)

Summary of the experimental results. The first column is the Rfam identifier, the second column is the length of the structure. The rest of the columns are: (sol) number of runs where the algorithm returned a solution out of 50 executions (for MODENA is the number of correct individuals in the final population), and (time) the average time (in seconds) to find a solution (over the runs that did return a solution), for all the algorithms tested. The last two rows show sum and average values.

algorithm.

Table 4. **Summary of solved structures for sets 1,2,3.**

	CP	INFO-RNA	MODENA	RNA-SSD	RNAinverse
Total solved	**2568**	2145	1423	2127	1368
Σ avg time	2834.1	526.4	6639.7	1645.5	5694.5
Str solved	53	53	45	**54**	35
avg avg time	53.5	**9.9**	147.5	30.5	162.7

Summary table showing: (1) Total number of successful runs, (2) sum of average times, i.e., the sum of all average times in previous tables, (3) number of structures solved, i.e., number of structures for which the algorithm returned at least one solution, and (4) double averaged time, i.e., sum of average times divided by number of structures solved.

LNS results

Table 5 shows a comparison of our LNS algorithm over the Rfam set of structures [c]. Recall that we added different variable and value heuristics with the goal of solving more inverse folding subproblems, and of increasing randomization to escape revisiting the same sequences again and again. We performed this comparison to sort out which combination of heuristics is best. Boldface results signify the best result, i.e. which solves a higher percentage of runs and, in case of a tie, does so with a lower average time.

The results show that LNS with none of these added mechanisms is superior for a larger number of sequences. However, these tables also show that LNS (with added variable and value heuristics) is capable of solving more sequences, more quickly, for target structures that are larger and more complex.

EteRNA results

Lastly, to show the use of introducing design constraints, we selected a set of 12 inverse folding problem instances from the *EteRNA* web site `http://eterna.cmu.edu`. Results for both the CP and LNS programs are shown in Table 6. Note that no other approach in the literature can solve these inverse folding problems given their design constraints.

The EteRNA structures were selected at random, from the vast set of structures available. EteRNA classifies its structures in 6 different levels of difficulty (from 0 to 5) and we selected two structures from each level. The constraints represented in this small data set correspond to:

- **MAX GC:** maximum number allowed of GC base pairs. GC stacked base pairs are the most stable base pairs, limiting the maximum number of base pairs that

[c]We have performed the same comparison for the other datasets but it is not shown here due to space constraints. It is, however, reported in the web server space.

Table 5. **Rfam LNS Results.**

		Levels Bottom-Up				Leaves to root			
Parameters		A-U-C-G UP		variable UP		A-U-C-G UP		variable UP	
RF id	n	sol	time	sol	time	sol	time	sol	time
RF00001.121	117	50	8.86	50	14.11	**50**	**8.38**	50	13.83
RF00002.2	151	50	23.22	48	150.11	**50**	**22.53**	48	152.41
RF00003.94	161	0	-	13	241.69	0	-	10	253.70
RF00004.126	193	50	0.79	50	1.16	**50**	**0.41**	50	0.88
RF00005.1	74	**50**	**0.40**	50	0.86	50	0.46	50	0.51
RF00006.1	89	**50**	**0.39**	50	6.49	50	2.34	50	8.47
RF00007.20	154	50	5.20	50	6.85	**50**	**2.90**	50	6.43
RF00008.11	54	**50**	**0.01**	50	0.03	**50**	**0.01**	50	0.07
RF00009.115	348	**50**	**20.70**	50	185.07	50	25.46	49	181.30
RF00010.253	357	0	-	0	-	0	-	0	-
RF00011.18	382	0	-	0	-	0	-	0	-
RF00012.15	215	50	1.29	50	8.65	**50**	**1.25**	50	11.15
RF00013.139	185	50	0.23	50	2.00	**50**	**0.18**	50	3.13
RF00014.2	87	50	1.34	50	0.66	50	0.90	**50**	**0.10**
RF00015.101	140	**50**	**4.57**	50	7.80	50	4.94	50	10.10
RF00016.15	129	0	-	0	-	0	-	0	-
RF00017.90	301	50	15.94	50	18.11	**50**	**15.73**	50	21.79
RF00018.2	360	50	18.18	30	272.45	**50**	**15.67**	34	252.14
RF00019.115	83	**50**	**0.13**	50	0.70	50	0.19	50	0.61
RF00020.107	119	0	-	0	-	0	-	0	-
RF00021.10	118	50	0.07	50	0.92	**50**	**0.05**	50	0.65
RF00022.1	148	50	2.21	50	4.38	**50**	**1.10**	50	5.13
RF00024.16	451	0	-	0	-	0	-	0	-
RF00025.12	210	50	0.27	50	8.29	**50**	**0.21**	50	5.39
RF00026.1	102	**50**	**3.15**	50	10.92	50	4.47	50	4.89
RF00027.7	79	**50**	**0.03**	50	0.52	**50**	**0.03**	50	0.32
RF00028.1	344	49	56.48	50	101.35	**50**	**43.50**	50	93.38
RF00029.107	73	50	2.63	50	3.67	50	3.94	**50**	**2.34**
RF00030.30	340	48	9.76	49	49.76	**49**	**6.80**	45	34.10

Summary of the experimental results. Computational time (in seconds) was measured on an Intel Core i7-2630QM (2GHz, 16GB memory, Linux Ubuntu 10.4. Time limit for was set to 10 minutes. The first column is the Rfam identifier, the second column is the length of the structure. The rest of the columns are number of runs where the algorithm returned a solution (over a total of 50 runs) and the average time to find a solution (over the runs that did return a solution), for all the algorithms tested. *Levels bottom-up* heuristic is explained in section *CP* and it is the same variable ordering heuristic that the CP model uses; *leaves to root* heuristic is a variant which is introduced in section *LNS*.

can appear in the structure increases the difficulty of finding a sequence, at least, for someone trying to solve it "by hand".

- **MIN GU:** similarly, GU base pairs are less stable, and are penalized when they

Table 6. ***EteRNA* Results.**

Parameters		Constraints			LNS		CP	
description	n	M_GC	m_GU	M_G	sol	time	sol	time
Prion Pseudoknot	36	-	3	-	10	82.18	10	59.41
Human astrovirus	43	-	6	-	1	478.22	0	-
Homo Sapiens 1 Series	83	-	8	-	10	62.72	7	1.69
HIV Primer Binding Site	107	12	8	-	4	243.14	2	32.18
Homo Sapiens 3	109	10	20	-	1	482.54	0	-
Other Ribosomal RNA	112	12	6	2	10	122.03	10	1.05
Bacilus Subtilis sRNA	113	-	11	-	4	294.84	3	311.81
5s Ribosomal RNA	120	-	4	-	10	30.30	10	30.16
Tribolium Castaneum	123	18	13	-	7	224.71	4	83.77
Oryza sativa 4	176	40	20	-	10	215.83	0	-
Symbiotic plasmid	300	55	10	4	2	206.39	0	-
Telomerase RNA	546	-	15	-	6	297.43	0	-

Summary of the experimental results. Computational time (in seconds) were measured on an Intel Core i7-2630QM (2GHz, 16GB memory, Linux Ubuntu 10.4) Time limit was set to 10 minutes. The first column is the description, the second column is the length of the structure, the third column is the maximum number of GC base pairs allowed, the fourth column is the minimum number of GU base pairs and the fifth column is the maximum number of consecutive Gs. The rest of the columns are number of runs where the algorithm returned a solution (over a total of 10 runs) and the average time to find a solution (over the runs that did return a solution), for all the algorithms tested.

close a stem. Fixing a minimum number of GU base pairs increases difficulty as well.

- **MAX G:** maximum number allowed of consecutive Gs in the sequence. For similar reasons as MAX GC, this increases the difficulty of finding a sequence.

See Appendix A for results and comparison of these instances in terms of Ensemble Defect.

4. Availability and Future Work

In order to allow the research community to benefit from our new methods for RNA inverse folding with design constraints, we have created a web server at `http://bioinformatics.bc.edu/clotelab/RNAiFold`. This web site supports both the CP and LNS methods for single molecule RNA inverse folding, as well inverse folding for the hybridization of two RNA molecules. Source code for these programs is also

available at the same location.

Our current algorithms solve the classical RNA inverse folding problem, calculate a given number of (or all) the solutions, and return whether no solution exists. Additionally, our programs incorporate new design constraints. We plan to add new design constraints and to optimize other criteria such as ensemble defect. We also intend to perform experimental validations of our designed RNAs in the near future.

4.1. *Riboswitch Design*

Current and future work of our lab is to extend the current tool, `RNAiFold`, to support *riboswitch* design.[d] In this case, we need to determine an RNA sequence that folds into two different, metastable structures. Pioneering work has been done on the problem by Flamm et al. [19] and Zadeh et al. [54]. The latter group has actually performed both *in vitro* and *in vivo* RNA design. Since the method of Flamm et al. is a generalization of `RNAinverse`, we expect our CP and LNS approach to provide significant improvements.

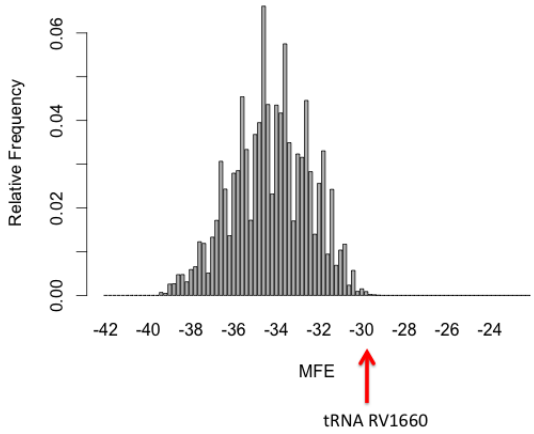
In designing an RNA sequence that folds into two distinct metastable states, S_1, S_2, one might consider the strategy of finding a sequence that folds into each of S_1, S_2 with a certain probability, since clearly both target metastable structures cannot simultaneously be the MFE structure. However, the Boltzmann probability of any given structure, including the MFE structure, may be tiny; hence, we will instead minimize *expected base pair distance* from target metastable structures for structures within a certain basin of attraction. In supplementary information we report preliminary analysis of known riboswitch sequences with respect to expected base pair distance and other measures, including *pointwise entropy*. See Appendix B for a description of relevant structural diversity measures in the context of RNA synthetic design.

4.2. *Structural Diversity, Robustness and RNA Evolution*

Given that our CP approach can return all sequences whose MFE structure is the given target structure, we can analyze the minimum free energy of these structures, as well as their structural diversity (see Appendix B). Such analysis can provide insights into subtle differences between naturally occurring RNA and synthetic RNA whose minimum free energy structures are identical. Such insights may prove important in future work in synthetic biology and molecular evolution theory.

[d]A bacterial riboswitch is a portion of the $5'$ untranslated region (UTR) of messenger RNA, that performs gene regulation by undergoing a conformational change upon binding with a ligand, such as guanine, thiamine pyrophosphate, lysine, etc. [43]. Recently, a eukaryotic riboswitch (the thiamine pyrophosphate, TPP, riboswitch (the most common bacterial riboswitch) has been found that resides in an intronic region and controls alternative messenger RNA splicing by conformational change [11].

Fig. 4. Minimum Free Energy distribution of tRNA



Minimum Free Energy distribution for over 4 million sequences returned by our algorithm, where RV1660 is the only tRNA which `RNAiFold` found among all sequences from the seed alignment of Rfam family RF00005.

As proof of concept, we computed the free energy of all sequences that `RNAiFold` determined,[e] which fold into the following tRNA consensus secondary structure (consensus structure taken from the Rfam RF00005 seed alignment):

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

Figure 4 plots the distribution of minimum free energy for all sequences output by our program.

[e]Available computer memory was exhausted, after `RNAiFold` returned over 4 million sequences, whose minimum free energy structure is the target structure, (taken to be the consensus secondary structure for Rfam family RF00005).

Acknowledgments

We would like to thank Dr. Taneda for his generosity in providing us with the set of target structures, obtained from Rfam data [47], and used in this study for benchmarking purposes. We would also like to thank the authors of all methods for making their software available to us. Funding for the research of P. Clote and I. Dotu was provided by the National Science Foundation with grants DMS-1016618 and DMS-0817971, with additional funding to P.C. by Digiteo Foundation. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Appendix A. Ensemble Defect and NUPACK comparison

Design constraints can improve the odds of the designed sequences to actually fold into the target structure (*in vivo* or *in vitro*) if they are specified as a result of some biologically relevant insights. In this case, however, we use them as a test for our algorithms, since they reduce the number of sequence solutions and thus, increase the difficulty of the problem.

On the other hand, even though our algorithms do not optimize ensemble effect, to demonstrate the quality of our solutions obtained when some design constraints are added and when the randomized value ordering heuristic is utilized, we present a comparison in terms of average ensemble defect with NUPACK for the EteRNA instances. This comparison is shown in table 7. Note that in some cases our sequences have better ensemble defect, although NUPACK is superior in most (this is not surprising, since NUPACK uses average ensemble defect in its search criteria).

However, NUPACK does not take into account the aforementioned constraints, which might or might not impact the resulting ensemble defect values. Moreover, our CP approach opens the possibility of calculating a large number of solutions (or all solutions, if desired), and subsequently filtering the solutions by other criteria, such as ensemble defect, structural diversity, etc.

Appendix B. Structural Diversity Measures

In this appendix, we define measures of structural diversity, all of which depend only on the computation of the base pairing probabilities

$$p_{i,j} = \sum_{\{S:(i,j)\in S\}} P(S) = \frac{\sum_{\{S:(i,j)\in S\}} \exp(-E(S)/RT)}{Z} \tag{B.1}$$

where $P(S)$ is the Boltzmann probability of structure S of a given RNA sequence $a = a_1, \ldots, a_n,$, $E(S)$ is the Turner energy of secondary structure S [38, 52], $R \approx 0.001987$ kcal/mol.K is the universal gas constant, T is absolute temperature, and the *partition function* $Z = \sum_S \exp(-E(S)/RT)$, where the sum is taken over all secondary structures S of a. As explained in [40, 55], probability $p_{i,j}$ of base pair

Table 7. ***EteRNA* Ensemble Defect Results.**

Parameters		Constraints			Avg Ensemble Defect	
description	n	Max GC	Min GU	Max G	NUPACK	CP
Prion Pseudoknot	36	-	3	-	**0.85%**	0.94%
Human astrovirus	43	-	6	-	**5.50%**	22.40%
Homo Sapiens 1 Series	83	-	8	-	**0.64%**	0.64%
HIV Primer Binding Site	107	12	8	-	**0.89%**	5.53%
Homo Sapiens 3	109	10	20	-	**0.45%**	11.85%
Other Ribosomal RNA	112	12	6	2	**0.86%**	3.09%
Bacilus Subtilis sRNA	113	-	11	-	**0.50%**	3.76%
5s Ribosomal RNA	120	-	4	-	4.10%	**1.27%**
Tribolium Castaneum	123	18	13	-	**3.20%**	9.68%
Oryza sativa 4	176	40	20	-	**0.50%**	1.06%
Symbiotic plasmid	300	55	10	4	**3.39%**	11.72%
Telomerase RNA	546	-	15	-	8.79%	**2.61%**

Comparison against NUPACK. Average Ensemble Defect for 1 sequence found with NUPACK (no constraints) and 1 sequence found with CP (with constraints). The first column is the description, the second column is the length of the structure, the third column is the maximum number of GC base pairs allowed, the fourth column is the minimum number of GU base pairs and the fifth column is the maximum number of consecutive Gs. The last two columns show average ensemble defect for NUPACK and CP sequences.

(i, j), where $1 \leq i < j \leq n$, can be computed in cubic time and quadratic space. For each fixed position $1 \leq i \leq n$, we define the probability distribution $p^*_{i,j}$, for j varying in $[1, n+1]$, by symmetrizing p for values $1 \leq i, j \leq n$, and then define $p^*_{i,n+1} = 1 - \sum_{j>i} p_{i,j} - \sum_{j<i} p_{j,i}$ [16, 41].

Expected pointwise entropy. For a given RNA sequence $a = a_1, \ldots, a_n$ and fixed position $1 \leq i \leq n$, the (Shannon) entropy of the probability distribution $p_{i,j}$, as j varies in $[1, n+1]$ is defined by $H_i(a) = -\sum_{j=1}^{n+1} p_{i,j} \cdot \ln p_{i,j}$. Given an RNA sequence $a = a_1, \ldots, a_n$, we define the *expected pointwise entropy* $\langle H(a) \rangle$ by $\langle H(a) \rangle = -\sum_{i=1}^{n} \sum_{j=1}^{n+1} \frac{p_{i,j} \cdot \ln p_{i,j}}{n}$. Clearly, if all low energy secondary structure of the RNA sequence $a = a_1, \ldots, a_n$ closely resemble the minimum free energy (MFE) structure, then the expected pointwise entropy is close to 0.

Expected base pair distance from a structure. Let S_0 be an arbitrary secondary structure of the RNA sequence $a_1, \ldots, a_n$. The expected base pair distance

to S_0 is defined by

$$E[\{d_{\mathrm{BP}}(S, S_0) : S \in \mathbb{S}(a_1, \ldots, a_n)\}] = \sum_S P(S) \cdot d_{\mathrm{BP}}(S, S_0). \tag{B.2}$$

For brevity, we will write $E[\text{BP-distance to } S_0]$, or even $E[d_{\mathrm{BP}}(S_0)]$, to abbreviate $E[\{d_{\mathrm{BP}}(S, S_0) : S \in \mathbb{S}(a_1, \ldots, a_n)\}]$, defined in equation (B.2). We have the following.[f]

$$\begin{aligned} E[d_{\mathrm{BP}}(S_0)] &= \sum_S P(S) \cdot d_{\mathrm{BP}}(S, S_0) = \sum_S P(S) \cdot \left[\sum_{(i,j) \in S - S_0} 1 + \sum_{(i,j) \in S_0 - S} 1 \right] \\ &= \sum_{1 \le i < j \le n} I[(i,j) \notin S_0] \cdot \sum_S P(S) + \sum_{1 \le i < j \le n} I[(i,j) \in S_0] \cdot \sum_{\{S : (i,j) \notin S\}} P(S) \\ &= \sum_{1 \le i < j \le n} I[(i,j) \notin S_0] p_{i,j} + \sum_{1 \le i < j \le n} I[(i,j) \in S_0] \cdot (1 - p_{i,j}) \\ &= \sum_{1 \le i < j \le n} I[(i,j) \notin S_0] \cdot p_{i,j} + I[(i,j) \in S_0] \cdot (1 - p_{i,j}) \end{aligned} \tag{B.3}$$

In this derivation, $I[(i,j) \notin S_0]$ denotes the indicator function for whether the base pair (i,j) does *not* belong to S_0. Although this notion, and the derivation (B.3) both appear to be new, there is a clear relation to the notion of *structural diversity*, $\langle D_v \rangle$, defined in the source code of Vienna RNA Package [25,28] as follows: $\langle D_v \rangle = \sum_{S,T} P(S) \cdot P(T) \cdot d_{\mathrm{BP}}(S,T) = \sum_{i=1}^n \sum_{j=1}^n p_{i,j} \cdot (1 - p_{i,j})$.

Ensemble defect. Given RNA sequence $a = a_1, \ldots, a_n$ and target structure S_0, Dirks et al. [16] define the *ensemble defect*, denoted by $n(a, S_0)$, to be the expected number of nucleotides whose base pairing status differs from target structure S_0, taken over the ensemble of secondary structures of a. Formally, we recall that

$$n(a, S_0) = n - \sum_{1 \le i,j \le n} p^*_{i,j} \cdot I[(i,j) \in S_0] - \sum_{1 \le i \le n} p^*_{i,n+1} \cdot I[i \text{ unpaired in } S_0]$$

where p^* is defined above, and I is the indicator function. This distance measure is clearly motivated by the notion of *structural diversity*, $\langle D_{mh} \rangle$, defined by Morgan and Higgs [41] and computed by Lorenz and Clote [33] in the context of the ensemble of locally optimal (kinetically trapped) secondary structures. Following Morgan and Higgs, we have $\langle D_{mh} \rangle = n - \sum_{i=1}^n \sum_{j=1}^{n+1} (p^*_{i,j})^2$.

For a related statistical mechanics study of RNA folding see [31].

A study of these measures for known Riboswitches is shown in table 8. Our next step is to implement an algorithm where a sequence has to fold into two distinct metastable secondary structures, using some of the previously mentioned measures as a metric.

[f]To the best of our knowledge, the observation in equation (B.3), that expected base pair distance to a target structure S_0 can be computed in $O(n^3)$ time, seems to be new.

Table 8. **Different Probability measures.**

Measures	Flamm	Guanine	xpt-pbuX	S	TPP	A
Length	45	148	202	141	146	113
BPdist	25	42	48	25	9	24
$E[S1]$	-12.2	-55.7	-66.7	-40.6	-33.92	-26.5
$E[S2]$	-10.8	-38.6	-42.16	-23.9	-19.32	-23.8
$E_{BPdist}(S_1)$	11.07	3.85	18.55	79.91	22.63	25.26
$E_{BPdist}(S_2)$	14.91	42.8	60.91	64.17	28.79	31.39
μ-H	0.66	0.17	0.5	0.46	0.31	0.84
σ-H	0.26	0.19	0.39	0.36	0.34	0.4
$\langle D_v \rangle$	12.87	6.85	30.23	24.73	16.33	33.71
barrier	11.8	27.89	25.64	18.2	16.9	10.6
$n(S_1)$	16.84	6.85	29.45	106.93	36.02	37.22
$n(S_2)$	22.93	59.17	82.97	90.36	49.32	48
$\langle D_{mh} \rangle$	25.51	51.98	90.54	69.56	44.98	63.94

Type means riboswitch type, where F is the engineered bistable switch of Flamm et al. [20]; guanine is *Bacillus subtilis* guanine riboswitch [48]; xpt-pbuX is *Bacillus subtilis* xpt-pbuX riboswitch [35]; S is *Thermoanaerobacter tencongensis* S-adenosylmethionine riboswitch [48]; TPP is *T. tencongensis* TPP riboswitch [48]; A is *Vibrio vulnificus* adenine riboswitch [48]. Len is sequence length. BP dist is the base pair distance between metastable structures S_1, S_2. $E[S1]$ and $E[S2]$ are resp. free energies of S_1, S_2. Exp BP dist S_1 is expected base pair distance to S_1, computed by (B.2), and similarly for Exp BP dist S_2. The mean, μ-H and standard deviation σ-H of pointwise entropy are explained in the text, as well as $\langle D_v \rangle$, lthe Vienna structural diversity. Barrier energy is computed by our algorithm [17]. $n(S_1)$ and $n(S_2)$ denote ensemble defect, as defined in (B.4). $\langle D_{mh} \rangle$ is the Morgan-Higgs structural diversity.

References

1. E. S. Andersen. Prediction and design of DNA and RNA structures. *N. Biotechnol.*, 27(3):184–193, July 2010.
2. M. Andronescu, AP. Fejes, F. Hutter, HH. Hoos, and A. Condon. A new algorithm for rna secondary structure design. *J Mol Biol.*, 336:607–624, 2004.
3. C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181:223–230, 1973.
4. W.C. Babcock. Intermodulation interference in radio systems/frequency of occurrence and control by channel selection. *Bell System Technical Journal*, 31:63–73, 1953.
5. M. H. Bailor, X. Sun, and H. M. Al-Hashimi. Topology links RNA secondary structure with global conformation, dynamics, and adaptation. *Science*, 327(5962):202–206, January 2010.
6. A.R. Banerjee, J.A. Jaeger, and D.H. Turner. Thermal unfolding of a group I ribozyme: The low-temperature transition is primarily disruption of tertiary structure. *Biochemistry*, 32:153–163, 1993.
7. R. Bellman. On the approximation of curves by line segments using dynamic pro-

gramming. *Communications of the ACM*, 4(6):284, 1961.
8. S. J. Brouns, M. M. Jore, M. Lundgren, E. R. Westra, R. J. Slijkhuis, A. P. Snijders, M. J. Dickman, K. S. Makarova, E. V. Koonin, and J. Van der Oost. Small CRISPR RNAs guide antiviral defense in prokaryotes. *Science*, 321(5891):960–964, August 2008.
9. A. Busch and R. Backofen. Info-rna, a fast approach to inverse rna folding. *Bioinformatics*, 22(15):1823–1831, 2006.
10. T.-H. Chang, H.-D. Huang, L.-C. Wu, C.-T. Yeh, B.-J. Liu, and J.-T. Horng. Computational identification of riboswitches based on RNA conserved functional sequences and conformations. *RNA*, 15(7), 2009.
11. M. T. Cheah, A. Wachter, N. Sudarsan, and R. R. Breaker. Control of alternative RNA splicing and gene expression by eukaryotic riboswitches. *Nature*, 447(7143):497–500, May 2007.
12. Y. Y. Chen, M. C. Jensen, and C. D. Smolke. Genetic control of mammalian T-cell proliferation with synthetic RNA regulatory systems. *Proc. Natl. Acad. Sci. U.S.A.*, 107(19):8531–8536, May 2010.
13. S. S. Cho, D. L. Pincus, and D. Thirumalai. Assembly mechanisms of RNA pseudoknots are determined by the stabilities of constituent secondary structures. *Proc. Natl. Acad. Sci. U.S.A.*, 106(41):17349–17354, October 2009.
14. K. Darty, A. Denise, and Y. Ponty. VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15):1974–1975, August 2009.
15. K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley and Sons, 2001.
16. R.M. Dirks, M. Lin, E. Winfree, and N.A. Pierce. Paradigms for computational nucleic acid design. *Nucleic. Acids. Res.*, 32(4):1392–1403, 2004.
17. I. Dotu, W. A. Lorenz, P. Van Hentenryck, and P. Clote. Computing folding pathways between RNA secondary structures. *Nucleic. Acids. Res.*, 38(5):1711–1722, 2010.
18. J.A. Doudna and T.R. Cech. The chemical repertoire of natural ribozymes. *Nature*, 418(6894):222–228, 2002.
19. C. Flamm, I. L. Hofacker, S. Maurer-Stroh, P. F. Stadler, and M. Zehl. Design of multistable RNA molecules. *RNA.*, 7(2):254–265, February 2001.
20. Christoph Flamm, Ivo L. Hofacker, Sebastian Maurer-Stroh, Peter F. Stadler, and Martin Zehl. Design of multi-stable RNA molecules. *RNA*, 7:254–265, 2001.
21. 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.
22. 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. Res.*, 39(Database):D141–D145, January 2011.
23. J. Gorodkin, L. J. Heyer, and G. D. Stormo. Finding common sequence and structure motifs in a set of RNA sequences. *Proc Int Conf Intell Syst Mol Biol*, 5:120–123, 1997.
24. J. Gorodkin, L. J. Heyer, and G. D. Stormo. Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Res.*, 25(18):3724–3732, 1997.
25. AR. Gruber, R. Lorenz, SH. Bernhart, R. Neubock, and IL. Hofacker. The Vienna RNA websuite. *Nucleic Acids Research*, 36:70–74, 2008.
26. C. Hammann and E. Westhof. Searching genomes for ribozymes and riboswitches. *Genome Biol*, 8:210, 2007.
27. Pascal Van Hentenryck and Laurent Michel. *Constraint-Based Local Search*. MIT Press, 2005. ISBN-10: 0-262-22077-6 ISBN-13: 978-0-262-22077-4.

28. I.L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Res.*, 31:3429–3431, 2003.
29. I.L. Hofacker, W. Fontana, P.F. Stadler, L.S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatsch. Chem.*, 125:167–188, 1994.
30. F. W. Huang, W. W. Peng, and C. M. Reidys. Folding 3-noncrossing RNA pseudoknot structures. *J. Comput. Biol.*, 16(11):1549–1575, November 2009.
31. M. Huynen, R. Gutell, and D. Konings. Assessing the reliability of RNA folding using statistical mechanics. *J. Mol. Biol.*, 267:1104–1112, 1997.
32. L.P. Lim, M.E. Glasner, S. Yekta, C.B. Burge, and D.P. Bartel. Vertebrate microRNA genes. *Science*, 299(5612):1540, 2003.
33. W. A. Lorenz and P. Clote. Computing the partition function for kinetically trapped RNA secondary structures. *PLoS. One.*, 6(1):e16178, 2011.
34. R. B. Lyngso and C. N. Pedersen. RNA pseudoknot prediction in energy-based models. *J. Comput. Biol.*, 7(3-4):409–427, 2000.
35. M. Mandal, B. Boese, J.E. Barrick, W.C. Winkler, and R.R. Breaker. Riboswitches control fundamental biochemical pathways in Bacillus subtilis and other bacteria. *Cell*, 113(5):577–586, 2003.
36. N. R. Markham and M. Zuker. UNAFold: software for nucleic acid folding and hybridization. *Methods Mol. Biol.*, 453:3–31, 2008.
37. D.H. Mathews, M.D. Disney, J.L. Childs, S.J. Schroeder, M. Zuker, and D.H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. Natl. Acad. Sci. USA*, 101:7287–7292, 2004.
38. D.H. Mathews, J. Sabina, M. Zuker, and D.H. Turner. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, 288:911–940, 1999.
39. D.H. Mathews and D.H. Turner. Dynalign: An algorithm for finding the secondary structure common to two RNA sequences. *J. Mol. Biol.*, 317:191–203, 2002.
40. J.S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29:1105–1119, 1990.
41. S.R. Morgan and P.G. Higgs. Barrier heights between ground states in a model of RNA secondary structure. *J. Phys. A: Math. Gen.*, 31:3153–3170, 1998.
42. A. M. Poole, D. C. Jeffares, and D. Penny. The path from the RNA world. *Journal of Molecular Evolution*, 46:1–17, 1998.
43. A. Serganov, Y.R. Yuan, O. Pikovskaya, A. Polonskaia, L. Malinina, A.T. Phan, C. Hobartner, R. Micura, R.R. Breaker, and D.J. Patel. Structural basis for discriminative regulation of gene expression by adenine- and guanine-sensing mRNAs. *Chem. Biol.*, 11(12):1729–1741, 2004.
44. R. P. Shetty, D. Endy, and T. F. Knight, Jr. Engineering BioBrick vectors from BioBrick parts. *J. Biol. Eng.*, 2(1):5, 2008.
45. S. Sidon. Ein Satz über trigonometrische Polynome und seine Anwendungen in der Theorie der Fourier-Reihen. *Mathematische Annalen*, 106:536–539, 1932.
46. G.A. Soukup and R.R. Breaker. Relationship between internucleotide linkage geometry and the stability of RNA. *RNA*, 5:1308–1325, 1999.
47. A. Taneda. Modena: a multi-objective rna inverse folding. *Advances and Applications in Bioinformatics and Chemistry*, 4(1), 2011.
48. C. A. Wakeman, W. C. Winkler, and CE Dann. Structural features of metabolite-sensing riboswitches. *Trends Biochem. Sci.*, 32(9):415–424, September 2007.
49. S. Washietl and I. L. Hofacker. Identifying structural noncoding RNAs using RNAz.

Curr Protoc Bioinformatics, 0(O):O, September 2007.

50. K.A. Wilkinson, E.J. Merino, and K.M. Weeks. RNA SHAPE chemistry reveals nonhierarchical interactions dominate equilibrium structural transitions in $tRNA^{Asp}$. *J. Am. Chem. Soc.*, 127:4659–4667, 2005.
51. M. Wu and I. Tinoco Jr. RNA folding causes secondary structure rearrangement. *PNAS*, 95(20):11555–11560, 1998.
52. T. Xia, J. SantaLucia Jr., M.E. Burkard, R. Kierzek, S.J. Schroeder, X. Jiao, C. Cox, and D.H. Turner. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemistry*, 37:14719–35, 1999.
53. C. Xue, F. Li, T. He, G. P. Liu, Y. Li, and X. Zhang. Classification of real and pseudo microRNA precursors using local structure-sequence features and support vector machine. *BMC. Bioinformatics*, 6:310, 2005.
54. J. N. Zadeh, B. R. Wolfe, and N. A. Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *J. Comput. Chem.*, 32(3):439–452, February 2011.
55. M. Zuker. On finding all suboptimal foldings of an RNA molecule. *Science*, 244(4900):48–52, April 1989.
56. M. Zuker. Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.*, 31(13):3406–3415, 2003.
57. M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, 9:133–148, 1981.
58. M. Zytnicki, C. Gaspin, , and T. Schiex. Darn a weighted constraint solver for RNA motif localization. *Constraints*, 13:91–109, 2008.