RNAblueprint: Flexible and universal multiple target nucleic acid sequence design

Stefan Hammer[1,2,*], Birgit Tschiatschek[2],
Christoph Flamm[1,3], Ivo L. Hofacker[1,2,4,*] and Sven Findeiß[1,2,*]

August 2016

[1]University of Vienna, Faculty of Chemistry, Department of Theoretical Chemistry, 1090 Vienna, Austria,
[2]University of Vienna, Faculty of Computer Science, Research Group Bioinformatics and Computational Biology, 1090 Vienna, Austria,
[3]University of Vienna, Research Network Chemistry Meets Microbiology, 1090 Vienna, Austria and
[4]University of Copenhagen, Center for Non-coding RNA in Technology and Health, Copenhagen, DK-1870, Denmark.

Abstract

Motivation: Realizing the value of synthetic biology in biotechnology and medicine requires the design of molecules with specialized functions. Due to its close structure to function relationship, and the availability of good structure prediction methods and energy models, RNA is perfectly suited to be synthetically engineered with predefined properties. However, currently available RNA design tools cannot be easily adapted to accommodate new design specifications. Furthermore, complicated sampling and optimization methods are often developed to suit a specific RNA design goal, adding to their inflexibility.
Results: We developed a `C++` library implementing a graph coloring approach to uniformly sample sequences compatible with structural and sequence constraints from the typically very large solution space. Uniform sampling from the solution space not only makes optimization runs much more performant, but also raises the probability of finding better solutions for long optimization runs. We show that our software can be combined with any other software package to allow diverse RNA design applications. Scripting interfaces allow the easy adaption of existing code to accommodate new scenarios, making the whole design process universal and flexible. We implemented example design approaches written in `Python` to demonstrate the advantages of a scripting language in conjunction with the `RNAblueprint` library for sequence sampling.
Availability: `RNAblueprint`, `Python` implementations and benchmark data sets are available at github: `https://github.com/ribonets/`
Contact: s.hammer@univie.ac.at
Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 Introduction

RNA molecules are omnipresent in all domains of life. They execute diverse functions including small molecule sensing, signal transduction and gene regulation. RNA is a molecule well-suited for designing with predefined functionality. This is mainly due to its close structure to function relationship and the physio-chemically grounded energy models for straightforward *in silico* calculations at the level of secondary structure. In recent years, due to the advent of synthetic biology, more researchers are focusing on the design of synthetic RNAs. There has been increasing success in modifying existing systems and incorporating novel functionality in RNAs within a cellular context (Chappell *et al.*, 2015; Espah-Borujeni *et al.*, 2015; Green *et al.*, 2014; Rodrigo *et al.*, 2012)

To produce an RNA molecule with a prescribed function, the close structure to function relationship must be incorporated into the design process, along with a rationally defined specification of the structure that performs that function. A solution can then be obtained by generating an RNA sequence that complies with the structural constraints, i.e., is able to fold into the defined structure. This is known as the "inverse folding problem" (Hofacker *et al.*, 1994). Biologically active RNA molecules such as aptamers or ribozymes frequently require specific nucleotide patterns in binding or catalytic domains. Therefore the designed RNA must also comply with certain sequence constraints. Several computational tools capable of solving this hard combinatorial optimization problem have been published. These tools differ mainly in how the initial sequence is selected and which search strategy, e.g. stochastic local or global search, is applied. Both algorithmic characteristics have a big impact on the success of the optimization (see Supplementary Table 1)).

A variety of RNA molecules, natural as well as artificial, have been described, which exploit structural change as their functional mechanism. Usually, the structural switching of these RNAs between an inactive and the active conformation is induced by an external trigger, which can be as diverse as temperature, small organic molecules, or other small RNAs (Berens and Suess, 2015). The design of such RNA devices requires finding a sequence compatible with two or more structural inputs. Designing a bi-stable RNA was first solved by Flamm et al. (Flamm *et al.*, 2001) using a graph coloring approach. Recent tools can now also design multi-state (three or more) RNA molecules (Höner zu Siederdissen *et al.*, 2013; Lyngso *et al.*, 2012; Taneda, 2015; Wolfe and Pierce, 2015). If the trigger is another RNA molecule, this requires algorithms that can handle multi-sequence folding and/or multi-state as well as pseudoknotted structures. Such capabilities are for instance implemented in the `NUPACK` design and analysis framework (Zadeh *et al.*, 2011b).

Sampling sequences, compatible with multiple structural constraints can be achieved using a complex graph coloring algorithm (Höner zu Siederdissen *et al.*, 2013), which guarantees that each solution is drawn statistically fairly with equal probability. In contrast, other approaches use ad hoc sampling heuristics that introduce biases. Thus, good solutions may be missed because the solution space is not fully explored. Furthermore, frequent re-evaluation of already discovered solutions due to biased sampling leads to inefficient optimization, especially if the calculation of the objective involves demanding calculations such as pseudoknot structure prediction.

A review of the literature revealed that published RNA designs were either achieved by manual ad hoc approaches or very specific software implementations, which can handle only restricted design problems on a case-by-case basis (Isaacs *et al.*, 2004; Neupert *et al.*, 2008; Qi *et al.*, 2012; Rodrigo and Jaramillo, 2014; Wachsmuth *et al.*, 2013). Very recent publications focus on the flexibility of the design approach and provide methods and interfaces to allow the specification of broader objectives (Höner zu Siederdissen *et al.*, 2013; Taneda, 2015). However, the diversity of the objectives is still limited and introducing a new feature in the objective function requires changes in the program code (some of which are closed source). Furthermore, the mechanisms of optimization in existing tools are always predefined and very rigid.

To address these limitations, we developed `RNAblueprint`which enables the fair sampling of RNA sequences compatible with multiple structural and sequence constraints. The library can be easily integrated into existing tools. It is therefore now possible to focus on the formulation of the objective function as the most crucial part of the design process. Until now this aspect was largely neglected, even though the objective describes best how the design should function. The actual optimization process is swapped into the scripting interface, where we offer predefined solutions but also give the user the opportunity to easily implement new ideas without having to change the source code of the core library. This flexibility is a major advantage of our approach.

With our framework, in addition to predicting RNA structure and RNA-RNA interactions, and allowing for pseudoknot incorporation (Janssen and Giegerich, 2015; Lorenz *et al.*, 2011; Zadeh *et al.*, 2011b,a) recent methods for the calculation of RNA-ligand interactions can also be incorporated (Lorenz *et al.*, 2016). Using `RNAblueprint` and its scripting interface we here implemented a classic multi-state design, which we used to analyze and benchmark our software. The developed software allows us to effectively solve problems including the design of translational and transcriptional on/off switches, triggered by diverse inputs such as small RNAs, ligands, temperature, salt concentration or proteins. `RNAblueprint` can also be used to specify the design of RNA or DNA scaffolds in synthetic biology, and to construct RNA/DNA origami.

2 Approach

An RNA sequence $x = \{x_1, x_2, x_3, \ldots, x_n\}$ is constructed from a set of monomers $x_i \in \mathcal{A} = \{A, U, G, C\}$ that can interact by forming base pairs (i, j), $1 \leq i < j \leq n$ where i and j are positions separated by at least three bases and $(x_i, x_j) \in \mathcal{B} \subset \mathcal{A} \times \mathcal{A} = \{AU, UA, GC, CG, GU, UG\}$ the set of allowed base pairs. A set of base pairs of a sequence x is called secondary structure Θ.

`RNAblueprint` implements a method to sample RNA sequences compatible with all structures of a given set $\{\Theta_1, \Theta_2, \ldots, \Theta_M\}$ and a sequence constraint $y = \{y_1, y_2, y_3, \ldots, y_n\}$ where $y_i \subseteq \mathcal{A}$ is the set of allowed nucleotides at position i. To be able to uniformly sample from the entire solution space $\mathcal{C}$ (which is the set of all x compatible to all $\Theta_i, 1 \leq i \leq M$, given the sequence constraint y), we implemented the graph-theoretical coloring approach developed in (Höner zu Siederdissen *et al.*, 2013) which is depicted in Figure 1. The goal is to generate sequences that are compatible with a sequence constraint and a set of target structures. Such a design problem is represented as a *dependency graph* $G = (V, E)$ constructed as the union of the circle plot representations of the structural constraints (Supplementary Figure 1). Each

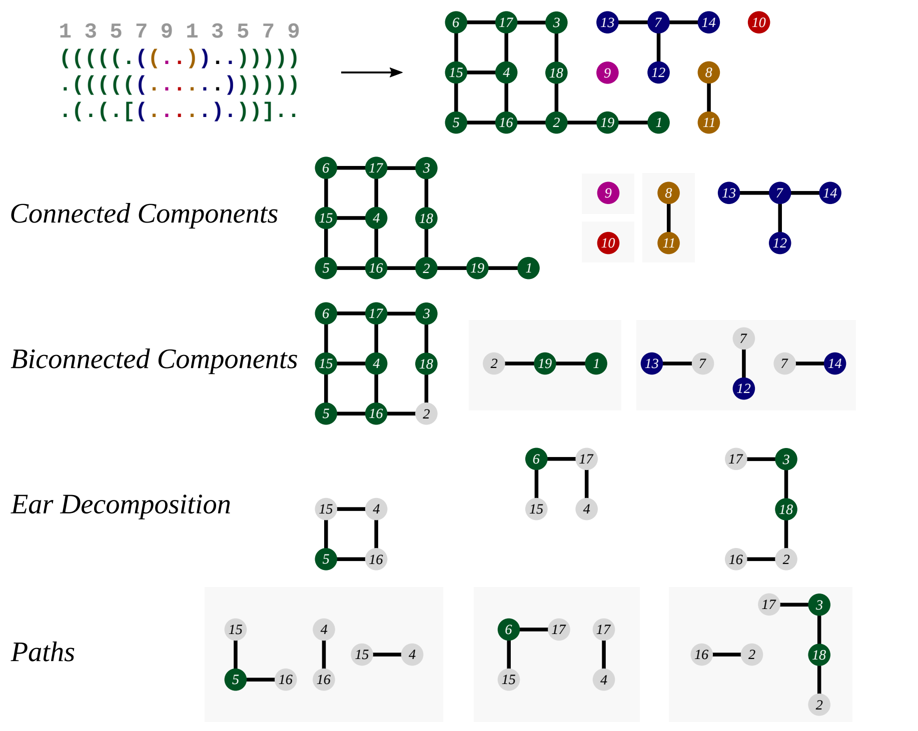


Figure 1: A dependency graph is hierarchically decomposed starting from the top and moving down through four levels to generate a decomposition tree. The dot-bracket strings (top left) denote three structural input constraints which are converted into a dependency graph (top right) by intersecting their circle representations, see Supplementary Figure 1. Gray boxed subgraphs are not decomposed further as their number of possible colorings can be obtained with the path coloring approach. ● nodes represent special vertices.

vertex $v_i \in V$ of the graph corresponds to a position $1 \leq i \leq n$ in the sequence to be designed, and the edges E represent base pairs (i, j) that are formed between them. Each base pair occurs in at least one of the input structures. The resulting graph needs to be bipartite to allow for a solution for the given structural constraints and $\mathcal{C} = \emptyset$ (Höner zu Siederdissen *et al.*, 2013). A coloring or base assignment on a vertex v_i is a single nucleotide $x_i \in \mathcal{A}$ assigned to the position i. Note that a sequence constraint y_i restricts the number of possible assignments of the corresponding vertex and can result in an unsolvable design problem if it contradicts the base pairing pattern enforced by the structural constraints. This might happen even if the dependency graph is bipartite but it can already be detected during the graph construction process.

Flamm *et al.* (2001) showed that paths and circles can be colored fairly with respect to the RNA alphabet $\mathcal{A}$ and the set of possible base pairs $\mathcal{B}$. To strip down the coloring problem, it is therefore desirable to split the dependency graph at vertices with degrees greater than two, denoted as a set of special vertices $\mathcal{S}$. These are usually called cut points for biconnected components or attachment points during the ear decomposition. An ear decomposition of graph G starting with a path P_0 is a decomposition of its edge set $E = P_0 \cup \ P_1 \cup \cdots \cup P_k$ where P_{i+1} is a simple path or ear whose endpoints belong to $P_0 \cup \cdots \cup P_i$, but its internal vertices do not (Maon *et al.*, 1986). For a guaranteed uniform sampling from $\mathcal{C}$, the remaining problem is the correct sampling of all vertices in $\mathcal{S}$, followed by the sampling of the adjacent paths. To solve this, we used a dynamic programming approach where we enumerate all possible combinations of colorings $\{y_i \ \forall \ V_i \in \mathcal{S}\}$ of $\mathcal{S}$ and calculate the number of solutions for each combination. Using stochastic sampling we can draw from this set weighted by the amount of solutions. As this approach is very memory and CPU demanding, it is important to follow a specific order of how to calculate and later sample base assignments for special vertices. We therefore decomposed the dependency graph step-wise into paths, see Figure 1. Complex connected components containing special vertices are decomposed into biconnected components and blocks, further following the ear decomposition described in (Maon *et al.*, 1986). As soon as the maximal degree of a subgraph H is two, either a path or a circle is reached and its decomposition is terminated. Using this decomposition approach, a tree of subgraphs is generated where the complete dependence graph sits at the root and each step of decomposition leads to a fixed order of subgraphs. This results in a decomposition tree with a maximum depth of four, where the different subgraphs of the decomposition populate specific levels of the tree (see Figure 1). For implementation purposes we check that each circle has at least two special vertices, and if not, introduce them at random. As a last step, all paths and circles are once more split at special vertices to ensure that only specials occur at path-ends.

Graph decomposition is done in a deterministic way, except for the ear decomposition (Maon *et al.*, 1986) step. This algorithm follows one of the many possible spanning trees of the corresponding graph. The memory and CPU requirements of the decomposition scale as $\mathcal{O}^{\alpha}$ and $\mathcal{O}^{\beta}$, respectively. As investigated in (Höner zu Siederdissen *et al.*, 2013), the values of α and β depend strongly on the spanning tree chosen. Since the exact analytic relationship between the shape of the spanning tree and these parameters is unknown, we generate a set of random instances of spanning trees and select the one with lowest α and β values.

Finding a compatible sequence proceeds in reverse order of the dependency graph decomposition, by step wise assembly of colored paths, i.e. paths where specific nucleotides have been assigned to the corresponding sequence positions, at special vertices into larger graphs until the dependency graph (the starting point of the original decomposition) is reached. In order to sample sequence assignments during the assembly process in a fair manner, we would need to memorize the number of possible colorings for every partially assembled intermediate graph. The number of colorings of unbranched paths of length l can easily be looked up in the l-th power of the pairing matrix $\mathcal{P}$. Since this information can be easily computed from the coloring of the end points and the path length, there is no need to exhaustively memorize path colorings, a source of combinatorial explosion, explicitly.

The coloring problem therefore reduces to the determination of the possible colorings of the special vertices (⚪ and 🔴 vertices in Figure 2) of all partially assembled intermediate graphs. This information can be efficiently calculated by a dynamic programming procedure that traverses the decomposition tree from the bottom up. The possible colorings of a set of special vertices of a subgraph $\mathcal{S}_{\mathcal{H}}$ are stored in a memorization table during the dynamic

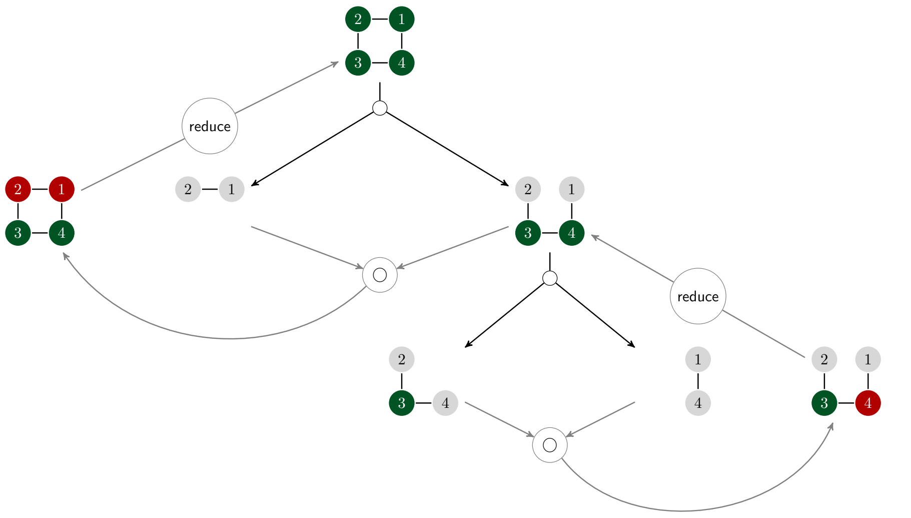


Figure 2: Algorithmic implementation of the decomposition (black arrows) and the reassembly (gray arrows) of a biconnected component. ● nodes are ordinary nodes and ● nodes indicate special vertices. ● nodes are internalized special vertices which can be converted to ordinary nodes with the `reduce` function. The matrix concatenation operator $\circ$ calculates the number of possible colorings of the combined subgraphs.

programming procedure. The dimension of such a table is determined by the number of special vertices a graph possesses. Since this number differs during the recursive traversal of the graphs in the decomposition tree (smaller graphs are connected at special vertices to larger units) the dimension of the memorization tables also varies. A table dimension itself is indexed by color (A, U, G, C), i.e. the elements of $\mathcal{A}$.

The memorization table of a larger graph (parent node in the decomposition tree) is calculated from the memorization tables of its two smaller constituting graphs (child nodes in the decomposition tree) in a type of concatenation procedure. The corresponding entries of the special vertices (table dimensions) are first multiplied component wise and then inserted into the new table. The memorization tables are sparse objects and the above construction procedure only increase dimensionality of the tables. The result would be a very large, sparse memorization table at the root node of the decomposition tree.

To avoid this wasting of memory resources, we introduced a dimension reduction step during the successive construction of the memorization tables. The reduction step rests on the observation that whenever the vertex degree of a special vertex in a partially assembled graph is equal to the vertex degree of the corresponding node in the intersection graph (root node of the decomposition tree) no further subgraph will be "attached" to this particular vertex in subsequent memorization table concatenation operations. Hence the corresponding dimension of the memorization table is collapsed via summing up the values over that particular special vertex, which shrinks the memorization table and internalizes the special vertex.

This implies that memorization tables for connected components have dimension zero since all special vertices have been internalized and removed via summation. In other words a memorization table with zero dimensions stores the total number of possible colorings for the respective subgraph. The memorization table for the root graph (i.e. the original intersection graph) therefore stores the size of the solution space, $|\mathcal{C}|$, which is equal to the total number of sequences compatible with the design constraints. With the help of the total number of sequences, the coloring count entries of the memorization tables can be re-interpreted as probabilities, paving the way for fair sampling approaches.

The sampling procedure works exactly in the opposite order of the memorization table calculation, as is usual in dynamic programming approaches. For each subgraph, special vertices are colored by stochastic sampling from the probability matrix, which corresponds to the re-interpreted memorization table, followed by the sampling of the graph itself, if it is a path. Otherwise the next hierarchical level of subgraphs is processed. If a special vertex has a base assigned already, this information is used during the stochastic sampling. Finally, when the last child has been processed, all bases are assigned and a solution was fairly drawn from the complete solution space.

Besides the *complete sampling* of a new sequence, there are two more procedures available for how to mutate or resample parts of the sequence. *Global sampling* resets the base assignments of all vertices of a random or specified connected component and draws new colors, i.e. nucleotide assignments, for these vertices. *Local sampling* randomly selects one path at the leaves of the decomposition tree and resamples only non-special vertices. This way we ensure the compatibility within a connected component. For both global and local sampling it can be useful to restrict the random selection of subgraphs by minimal and maximal size constraints. The possibility to resample a specific position in the sequence also exists. This either involves a local sampling of the path containing the position or, in cases where the selected position corresponds to a special vertex, a global sampling of the corresponding connected component. In this way, the ranges of positions to be sampled can be specified. A history of previous sampled sequences is stored, making it convenient to revert to those previous sequences if necessary.

The implementation was written in `C++` using the boost graph library and other parts of the boost library available at `http://www.boost.org/`. Using the `SWIG` framework, we offer an easy to use `Perl` and `Python` scripting interface to the library. Additionally, we developed a `Python` module so that code can be reused for many central components.

3 Methods

3.1 Objective function

The original objective function $f(x)$ proposed by Flamm *et al.* (2001) for two target designs was extended for multi-target case (Höner zu Siederdissen *et al.*, 2013) and is

$$f(x) = \underbrace{\sum_{i}^{M} E(x, \Theta_i) - G(x)}_{\text{dominate ensemble}} + \xi \underbrace{\sum_{i<j}^{M} (E(x, \Theta_i) - E(x, \Theta_j))^2}_{\text{minimize energy difference}} \tag{1}$$

where $G(x)$ is the ensemble free energy, $E(x, \Theta_i)$ is the free energy of the sequence x folded into structure Θ_i and ξ is a weighting factor typically set to 1. The first term is to maximize the frequency of each target structure in the ensemble to achieve dominance whereas the second term is to minimize the energy difference of target structures to get them to the same energy level. In (Taneda, 2015) the latter was changed to $\sum_{i<j}^{M} |E(x, \Theta_i) - E(x, \Theta_j)|$ which brings most of the target structure energies are close to the minimum free energy (MFE) and outliers are possible. In contrast the original version attempts to minimize the number of outliers and therefore the distance to the MFE of all states might be higher. Either way, weighting of the two terms is essential in single objective approaches. Although objective function (1) showed good performance on two-target designs, the straight-forward extension to three or more structures neglects the varying number of target structures. We therefore corrected the objective function to

$$\begin{aligned} f(x) =& \frac{1}{M} \sum_{i}^{M} E(x, \Theta_i) - G(x) \\ &+ \xi \frac{2}{M(M-1)} \sum_{i<j}^{M} |E(x, \Theta_i) - E(x, \Theta_j)| \end{aligned} \tag{2}$$

as we sum up M elements in the first term and build $\binom{M}{2}$ differences in the latter. With this new objective function, the ratio between the two terms is independent of the number of structures M. To resemble the good performance for the two-target structure case and keep the 1:1 ratio between the two two terms in the objective we set ξ to 0.5.

3.2 Benchmark Data sets

The number of target structures is only a rough estimate of the complexity of a given design problem. If the given structural constraints have no conflicting base pairs, the complexity of the connected components are just single vertices or paths of length two. If more overlap between the structural constraints exists, paths get longer, and complex subgraphs such as cycles and blocks occur. Based on a published tri-stable switch (Höner zu Siederdissen *et al.*, 2013), which contains only two cycles and eight paths of length two, we generated more complex examples by adding a fourth and fifth structural constraint, see Supplementary Figure 2 A-C. These three example inputs of increasing complexity were used to evaluate the implemented sampling procedures of `RNAblueprint`. The effect of fair sampling is tested on an extreme example that contains one large and complex connected component and a base pair as well as an unpaired position. To further reduce the solution space size, two sequence constraints were introduced, see Supplementary Figure 2 D.
Comparison with existing approaches was performed on the published data sets containing two-, three- and four-target designs as well as pseudoknotted two-target structure examples (Taneda, 2015). The applied optimization is described in section 3.3.

3.3 Classic Multi-State Design

To be able to benchmark against existing design software, we implemented an optimization approach consisting of `RNAblueprint` for fair sampling, the weighted objective function (2), and an adaptive walk. The latter works as follows: Consecutive sequence candidates are generated by randomly applying one of the three sampling methods, i.e. local, global or complete, and calculating the score of the objective function. The new sequence is only kept if the score is better than the current best solution. Depending on the chosen method, one randomly selected subgraph (local and global sampling) or all subgraphs (complete sampling) are redrawn. An exit value of 1000, being the maximum number of optimization trails with no score improvement, was used. To compare this approach to existing multi-target design tools we created 100 solutions for each member of the two-, three- and four-target design data sets described in (Taneda, 2015). Energy calculations for these data sets were made using the scripting bindings of the `ViennaRNA package` v2.2.5 (Lorenz *et al.*, 2011). As we are not restricted to nested base pairs in the structural input, the pseudoknotted two-target data sets described in (Taneda, 2015) were also used with exit value 100. This exit value is set to be much smaller because the runtime dramatically increases when using the `Nupack package` v3.0.4 (Zadeh *et al.*, 2011a) for pseudoknotted structure prediction. Furthermore, only 30 solutions were generated for each of the latter benchmark tasks.

4 Results and Discussion

Effect of fair sampling
A simplified version of the graph decomposition was implemented in `MODENA` (Taneda, 2015). Therein a naïve nucleotide assignment algorithm is used that is able to generate solutions of a design problem but not uniform sampling of the solution space. Furthermore, during the assignment of paired nucleotides without a sequence constraint, the G-U base pair is neglected. This generates a biased initial population of sequences that are subsequently optimized by applying a genetic algorithm. Although the Haskell prototype implementation in (Höner zu Siederdissen *et al.*, 2013) used lazy enumeration of all solutions and therefore allowed fair sampling, this was only for sufficiently small problems. Implementing the complete graph coloring algorithm (Höner zu Siederdissen *et al.*, 2013) and assigning all possible base pairs, `RNAblueprint` guarantees to fairly sample the complete solution space. Unfortunately, `MODENA` is available as binary only, of which the maximum population size is restricted to 1000 and at least one iteration of the genetic algorithm optimization is enforced. Therefore, we could not compare the effect of the implemented nucleotide assignment algorithm alone. However, to compare fair and unfair sampling we customized `RNAblueprint` by replacing the actual number of possible solutions stored in the probability matrix of each subgraph by one. While sampling with the fair approach led to an extreme value distributed frequency of uniquely found solutions, sampling the unfair way distorted the distribution and a few solutions were

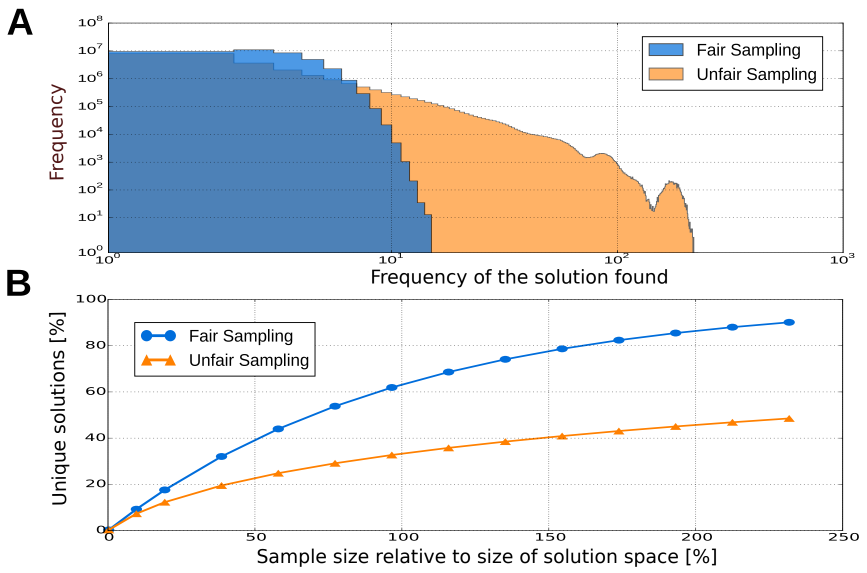


Figure 3: Differences in fair and unfair stochastic sampling shown on a small example with a rather complex dependency graph, see Supplementary Figure 2 D. (A) The histogram shows how frequent unique solutions were found when sampling completely new sequences using fair and unfair sampling. In total $9.6 \cdot 10^9$ solutions were sampled from $4.1 \cdot 10^7$ possible unique sequences (size of solution space). While fair sampling led to an extreme value distribution with the mean (2.57) count being slightly above the relative sample size and the maximum number of times a solution is rediscovered being 15, unfair sampling led to a distorted distribution where a solution is found 4.78 times on average and 227 times maximal. (B) When the sample size was chosen to be much bigger than the solution space ($\sim 230\%$), only about 50% of all possible solutions with unfair sampling was obtained for this example, while the fair method sampled about 90%. The performance of the fair sampling is independent of the underlying problem whereas the curve of the unfair approach heavily depends on the properties of the dependency graph.

generated with high frequency, Figure 3A. It followed that the solution space, by means of unique solutions generated, was explored much faster when applying the fair sample approach, Figure 3B. We expect that the naïve sampling approach of `MODENA` performs similarly to the shown unfair sampling method. It is worth noting that for small sample sizes, i.e. a few percentage of the complete solution space, both fair and unfair approaches performed equally. Our method capable of uniform sequence generation, implemented in `RNAblueprint`, could be used in any multi-state design software such as `MODENA` in order to explore the complete solution space of complex multi-state design problems in an unbiased way.

Sequence sampling

In a typical RNA design scenario, sequences compatible to the structural constraints are scored using an objective function. Thereby the sequence space is transformed into a landscape of complex and typically unknown structure that needs to be explored. Sampling completely new sequences generates solutions distributed over the complete landscape. This way, for an infinite sampling time the global optimum is always found. However, the optimization is rather slow because in each sampling step the reachable neighborhood is the complete solution space. The solution space of small examples is already of size $4.1 \cdot 10^7$ to $1.4 \cdot 10^{14}$ (Supplementary Figure 2) and therefore only a small fraction of all solutions is evaluated during a typical optimization run. The other sampling methods described in section 2 dramatically reduced the size of the reachable neighborhood. An adaptive walk using these move steps led to the solution ending up in local minima. The quality of these minima and how fast they were reached depended on the number of nucleotides changed in each step, Supplementary Figure 3.

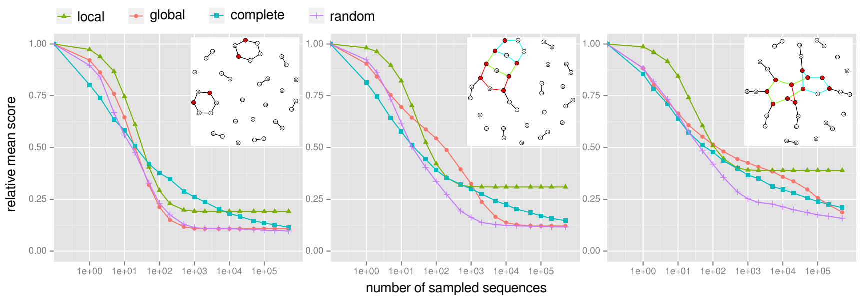


Figure 4: Score change during the optimization procedure using different move steps and dependency graphs. The x-axis shows the number of sampled sequences while the y-axis resembles the mean score from 100 optimization runs, normalized to the mean score of the initial randomly chosen sequences. Three different move steps (local, global and complete) and an additional run, where one of these moves was randomly picked at every step (random), are compared. At the most left plot a very simple dependency graph was generated, only consisting of paths and two circles, in the middle plot the graph already contains a block and on the right hand side many vertices are captured in one big connected component. The slope of the score change mainly results from two aspects, the rejection rate and the quality of the newly found solutions. Both are heavily dependent on the size of the move step, therefore we see a change from the left to the right plot, as the move steps of global, complete and random become bigger, Supplementary Figure 3.

In Figure 4, the published three state design example (Höner zu Siederdissen *et al.*, 2013) was extended to four and five input structures. The extension was made in a way that the complexity of the dependency graph from short paths and circles in the three state example was increased to larger connected components, Supplementary Figure 2. We compared the performance of different sampling methods that differ in the size of their largest move step. One method, called *complete*, always generates a completely new sequence. When sample *global* is applied, the assignments of a randomly selected connected component are redrawn. The random selection is weighted by the number of possible solutions associated to the connected components. In contrast, sample *local* resamples only non special vertices of a randomly selected path.

If the dependency graph contained only short paths and circles (three state example), the global sampling approach was similar to the local sampling, i.e. reached a local minimum relatively fast and the score converged. The relative mean score difference between local and global sampling minima results from the fact that special vertices were redrawn by the latter only. This allowed a maximum step size of up to six nucleotides (complete circle) compared to three nucleotides (longest path), Supplementary Figure 3. The more complex the dependency graph, i.e. the more special nodes and larger connected components exist, the more pronounced this difference between local and global sampling, Supplementary Figure 3. If one large connected component contained most of the bases (five state example), performing a global sampling where all assignments of the large component are most likely reassigned (Supplementary Figure 3), was similar to a completely new sampled sequence, i.e. the slope of the corresponding curves in Figure 4 are similar. However, the hamming distance to reachable neighbors was different for global and complete sampling, Supplementary Figure 3. Reaching a local minimum indicates that most likely no further score improvement can be made using the same sampling method. Changing the method and thereby changing the move step allows other local minima with better solutions to be reached. Interestingly, our analyses showed that randomly changing the sampling method in each step, *random* in Figure 4, gave significantly better results faster in most cases. We investigated the reachable neighborhood of selected time points during optimization of the four state design example in more detail, Supplementary Figure 4. After 1000 sampling steps, the mean score of sequences optimized with the random approach was significantly lower than the score reached with complete sampling (student's t-test p-value: 10^{-55}). Furthermore, the number of neighbors with a score below the current best solution was similar, Supplementary Figure 4. At the end point of the trend curves (after 500,000 sampling trails), global and random sampling reached the same mean scores and within their analyzed neighborhood of size 350,600 no better solution was found, Supplementary Figure 4. Interestingly, the sequences optimized with complete sampling did not reach the same mean score and the likelihood of generating a better solution was very low, Supplementary Figure 4. We stress again that these observations are highly dependent on the design problem, e.g. the complexity of the dependency graph and the length of the sequence to be designed. However, we show in the following that applying the random sampling method to a diverse benchmark data set of nested and pseudoknotted structural input gives reasonably good results.

Impact of normalization and weighting

To analyze the effect of the corrected objective function (2) and the applied optimization procedure we used the recently published benchmark data set (Taneda, 2015), which consists of two-, three- and four-target design problems as well as three pseudoknotted two-target sets. These examples were either taken from naturally occurring RNAs that are able to switch between structural states or were generated in a way that reachable, sub-optimal structures are taken as input constraints for the design process. `RNAblueprint` itself does no optimization on fairly sampled sequences. We implemented an adaptive walk that, given a start sequence, randomly selects one of the three sampling methods and applies it to generate the next sequence candidate. The generated sequence is retained if its score is better than the best prior solution. On the small examples evaluated in Figure 4, this approach adapted best to the varying complexity of the underlying dependency graphs. To score sequences, we applied an objective function that ensures on the one hand that the target structures of a good solution dominate the ensemble while on the other hand the energy difference between the target structures is minimized. In its original version (1), proposed for the two state design case in (Flamm *et al.*, 2001), the corresponding two terms were summed up without any weighting. Designs for two states gave reasonable results compared to other approaches, see Table 1. However, a systematic extension to three or even more states needs individual normalization of both terms. Therefore, we proposed a corrected objective function (2), which is adjusted to the good performing original two state objective. Especially for the four structure designs this yielded a significant improvement over the original one, see Table 1. Note, when using a multi-objective approach it is assumed that the weighting is implicitly found during optimization (Taneda, 2015).

Comparing the results of our naïve optimization procedure with multi-objective approaches that implement complex genetic algorithms to optimize sequences we performed similar or even better on the benchmark data set as measured by δe_1, i.e. the difference of the lowest energy target structure to the ground state and δe_2, i.e. the difference between the ground state and the highest energy target structure, on the benchmark data set. Furthermore, we also compared how often the desired target structures are energetically equal to the predicted MFE structure, see Supplementary Table 2-7. These values are termed n_i, i being the number of target structures with equal energy to the MFE. Given this benchmark measure, `MODENA` and `RNAblueprint` performed similarly. A notable difference between our approach and `MODENA` is that the latter uses a genetic algorithm to optimize a population of 500 individuals of which the best 100 are taken, while we performed 100 independent optimizations. We expect to get similar sequences from a population-based approach while the solutions generated with our approach are extremely diverse.

Although δe_1, δe_2 and n_i together are a good measure of the solution quality of this specific design problem, they do not describe the actual functionality of the switch *in vitro* or *in vivo*. An objective function describing every aspect necessary to create a functional switch might contain many more features, some of which cannot easily be calculated. Furthermore, it is questionable whether the creation of 100 solutions is even useful. It might be better to run the optimization longer and retrieve 10-20 heterogeneous solutions, as this is a more realistic number of solutions for verification in the laboratory.

Flexibility matters

Three example objective functions were proposed by Flamm and coworkers to design two-state temperature and structural switches (Flamm *et al.*, 2001). Those objectives have been adapted to multi-state design and are still used to benchmark new software (Höner zu Siederdissen *et al.*, 2013; Taneda, 2015). `MODENA` enables the user for the first time to choose from a catalog

Table 1: Comparison of currently available approaches to solve multi-target designs. Results of two-, three- and four-target designs are shown. For `RNAblueprint` and `MODENA` two-target designs of pseudoknotted structures are also presented. For each resulting sequence, we evaluated the difference between the most stable target structure to the ground state (δe_1) and the highest energy target structure to the ground state (δe_2). The mean (μ) and median ($\tilde{x}$) energy difference for 100 and 30 generated sequences is presented for the nested and pseudoknotted structure input, respectively. Performance of the individual sequences is listed in Supplementary Table 2-7. Boldface values highlight the best performing approach on a specific data set. For `RNAblueprint` the values for the *original* (1) and *corrected* (2) objective functions are listed.

	Nested Structure Input												Pseudoknotted Structure Input					
	`RNAblueprint`						`MODENA`[a]			`Frnakenstein`[a]			`RNAblueprint`			`MODENA`[a]		
	original			*corrected*														
	2str	3str	4str	2str	3str	4str	2str	3str	4str	2str	3str	4str	LE80	PK60	PK80	LE80	PK60	PK80
$\mu(\delta e_1)$	**0.28**	0.22	1.46	0.31	**0.10**	**0.48**	0.38	0.27	0.84	0.35	0.39	0.92	**0.82**	**0.03**	**0.15**	0.89	0.12	0.29
$\tilde{x}(\delta e_1)$	**0.00**	**0.00**	0.70	**0.00**	**0.00**	**0.05**	0.10	**0.00**	0.39	0.10	0.10	0.55	0.30	**0.00**	**0.00**	**0.20**	**0.00**	**0.00**
$\mu(\delta e_2)$	**0.34**	0.43	1.96	0.36	**0.26**	**1.21**	0.76	0.54	1.78	1.09	0.96	1.89	**1.09**	**0.08**	**0.17**	1.22	0.32	0.56
$\tilde{x}(\delta e_2)$	**0.00**	0.20	1.30	**0.00**	**0.10**	**0.80**	0.50	0.30	1.40	0.60	0.80	1.60	**0.55**	**0.00**	**0.00**	**0.55**	**0.00**	0.05

[a] Values taken from the original publication (Taneda, 2015).

of different structure prediction methods to calculate features of a given sequence and derive new objectives. However, this catalog is fixed and therefore the complete functionality of the applied software might not be available. This is especially true for recent developments, such as the soft constraint framework implemented in the `ViennaRNA package` (Lorenz *et al.*, 2016) and the test tube ensemble defect available in `NUPACK` (Wolfe and Pierce, 2015). Furthermore, the methods to optimize sequences, in the case of `MODENA` by applying a genetic algorithm, cannot be changed. Therefore, we implemented `RNAblueprint` as a library and equipped this sequence generator with a flexible scripting interface where the user can easily implement its own optimization procedures and come up with new objective functions.

5 Conclusion

We have developed a software solution which makes it possible to uniformly sample RNA sequences compatible with structural and sequence constraints. This enables efficient sampling from the entire solution space and avoides heavy re-evaluation of repeatedly generated solutions. Therefore, it is possible to review many more solutions, which leads to better results. Scripting interfaces make it easy to freely combine different optimization algorithms and to incorporate evaluations of different software packages into the objective function. We used the `NUPACK` and the `ViennaRNA` package to design multi-stable RNA structures with and without pseudoknots, respectively. With the scripting interface, any software such as the recently published `RNA shapes studio` (Janssen and Giegerich, 2015) and the approach by Wolfe and Pierce to reduce the amount of unwanted complexes when designing interacting molecules (Wolfe and Pierce, 2015), can be easily integrated. As the correct sequence generation problem is now efficiently solved, further research can focus on the challenging task of finding objective functions that better describe the goals and functions of RNA molecules. Using `RNAblueprint` it is now feasible to explore a much broader range of objectives and it is easy to adapt and recombine existing software and optimization techniques to generate an RNA molecule that perfectly suits the specific needs and goals of the task.

We illustrated the usefulness of our approach with typical but small sample applications. A general solution for solving all the diverse RNA design problems does not exist and there is also no universal way how to benchmark existing tools or novel approaches against each other. Applied measurements heavily depend on the goal and the objective of the design and therefore user knowledge is always necessary to choose an appropriate optimization method, move set and objective function.

Acknowledgments

Thanks to Christian Höner zu Siederdissen for assistance with the prototype Haskell implementation, Peter F. Stadler and Daniel Merkle for fruitful discussion and our private `boost` help desk Jakob L. Andersen. We thank Life Science Editors for proofreading and editing. Computational results have been achieved in part using the Vienna Scientific Cluster (VSC).

Funding

This work was supported by the European Commission under the Environment Theme of the 7th Framework Program for Research and Technological Development (GANr 323987), the COST-Action CM1304 “Systems Chemistry”, the FWF projects SFB F43 “RNA regulation of the transcriptome” and “Doktoratskolleg RNA Biology W1207-B09”.

References

Berens, C. and Suess, B. (2015). Riboswitch engineering — making the all-important second and third steps. *Current Opinion in Biotechnology*, **31**, 10–15.

Chappell, J., Takahashi, M. K., and Lucks, J. B. (2015). Creating small transcription activating RNAs. *Nat Chem Biol*, **11**(3), 214–220.

Espah-Borujeni, A., Mishler, D. M., Wang, J., Huso, W., and Salis, H. M. (2015). Automated physics-based design of synthetic riboswitches from diverse RNA aptamers. *Nucleic Acids Research*, page gkv1289.

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

Green, A. A., Silver, P. A., Collins, J. J., and Yin, P. (2014). Toehold Switches: De-Novo-Designed Regulators of Gene Expression. *Cell*, **159**(4), 925–939.

Hofacker, I. L., Fontana, W., Stadler, P. F., Bonhoeffer, L. S., Tacker, M., and Schuster, P. (1994). Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie / Chemical Monthly*, **125**(2), 167–188.

Höner zu Siederdissen, C., Hammer, S., Abfalter, I., Hofacker, I. L., Flamm, C., and Stadler, P. F. (2013). Computational design of RNAs with complex energy landscapes. *Biopolymers*, **99**(12), 1124–1136.

Isaacs, F. J., Dwyer, D. J., Ding, C., Pervouchine, D. D., Cantor, C. R., and Collins, J. J. (2004). Engineered riboregulators enable post-transcriptional control of gene expression. *Nat Biotechnol*, **22**(7), 841–847.

Janssen, S. and Giegerich, R. (2015). The RNA shapes studio. *Bioinformatics*, **31**(3), 423–425.

Lorenz, R., Bernhart, S. H., Siederdissen, C. H. z., Tafer, H., Flamm, C., Stadler, P. F., and Hofacker, I. L. (2011). ViennaRNA Package 2.0. *Algorithms for Molecular Biology*, **6**(1), 26.

Lorenz, R., Hofacker, I. L., and Stadler, P. F. (2016). RNA folding with hard and soft constraints. *Algorithms for Molecular Biology*, **11**, 8.

Lyngso, R. B., Anderson, J. W., Sizikova, E., Badugu, A., Hyland, T., and Hein, J. (2012). Frnakenstein: multiple target inverse RNA folding. *BMC Bioinformatics*, **13**(1), 260.

Maon, Y., Schieber, B., and Vishkin, U. (1986). Parallel ear decomposition search (EDS) and ST-numbering in graphs. *Theor. Comp. Sci.*, **47**, 277–298.

Neupert, J., Karcher, D., and Bock, R. (2008). Design of simple synthetic RNA thermometers for temperature-controlled gene expression in Escherichia coli. *Nucleic Acids Res*, **36**(19), e124.

Qi, L., Lucks, J. B., Liu, C. C., Mutalik, V. K., and Arkin, A. P. (2012). Engineering naturally occurring trans-acting non-coding RNAs to sense molecular signals. *Nucleic Acids Res*.

Rodrigo, G. and Jaramillo, A. (2014). RiboMaker: computational design of conformation-based riboregulation. *Bioinformatics*, **30**(17), 2508–2510.

Rodrigo, G., Landrain, T. E., and Jaramillo, A. (2012). De novo automated design of small RNA circuits for engineering synthetic riboregulation in living cells. *Proc Natl Acad Sci USA*, **109**(38), 15271–15276.

Taneda, A. (2015). Multi-objective optimization for RNA design with multiple target secondary structures. *BMC Bioinformatics*, **16**(1), 280.

Wachsmuth, M., Findeiß, S., Weissheimer, N., Stadler, P. F., and Mörl, M. (2013). De novo design of a synthetic riboswitch that regulates transcription termination. *Nucleic Acids Research*, **41**(4), 2541–2551.

Wolfe, B. R. and Pierce, N. A. (2015). Sequence Design for a Test Tube of Interacting Nucleic Acid Strands. *ACS Synthetic Biology*, **4**(10), 1086–1100.

Zadeh, J. N., Wolfe, B. R., and Pierce, N. A. (2011a). Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of Computational Chemistry*, **32**(3), 439–452.

Zadeh, J. N., Steenberg, C. D., Bois, J. S., Wolfe, B. R., Pierce, M. B., Khan, A. R., Dirks, R. M., and Pierce, N. A. (2011b). NUPACK: Analysis and design of nucleic acid systems. *Journal of Computational Chemistry*, **32**(1), 170–173.

Supplementary Material
RNAblueprint: Flexible and universal multiple target nucleic acid sequence design

Stefan Hammer, Birgit Tschiatschek, Christoph Flamm, Ivo L. Hofacker and Sven Findeiß

Contents

1 Supplementary Text

1.1 Local neighborhood of various move steps

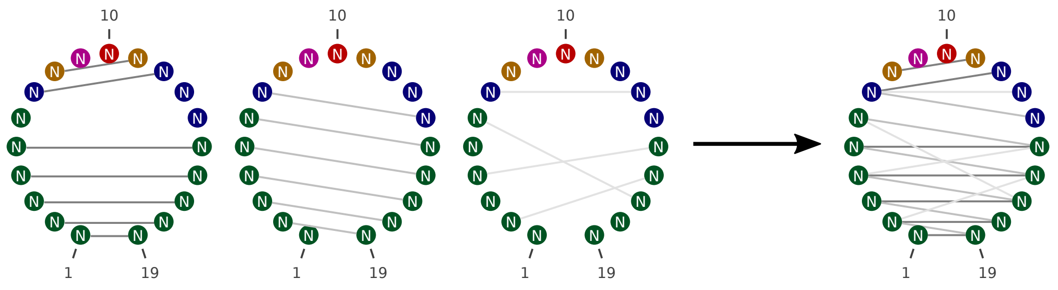
To get a better understanding of the solution landscape based on the introduced move steps, we analyzed the local neighborhood of three small examples with dependency graphs of varying complexity shown in Supplementary Figure 2. Using one of the introduced sampling methods (*complete*, *global* and *local*, see section 2 in the main text), the local neighborhood was explored by stochastic sampling. The analysis includes the actual hamming distance to the start sequence (Supplementary Figure 3), and the score change (Supplementary Figure 4) for the two parts of the multi-state objective function (formula (2) in the main text). Additionally the *random* move, where one of the four sampling methods is chosen randomly at each step was investigated. For global, 85% of the reachable neighborhood, i.e. 3506 neighbors, was generated for each of the 100 sequences. The same absolute number was used for the complete and random approach. With the local move, only very few neighbors can be reached, therefore we sampled as many solutions as possible using an exit condition.

The hamming distance describes the size of the move step in terms of actually changed nucleotides, see Supplementary Figure 3. The distribution of these distances for any move step was very much dependent on the structure of the dependency graph. For the four and five structure example the global move always showed a flat distribution at smaller hamming distances and one defined peak at a certain distance. This results from the fact that the corresponding dependency graphs consisted of one bigger connected component in addition to several small ones (see Supplementary Figure 2). The connected components could be divided into two disjoint sets due to the bipartite property of the base assignments. If the coloring pattern of the disjoint sets of the bigger component was changed in a way that the coloring switches, all nucleotides of this big connected component are changed, resulting in the defined peak with a hamming distance of exactly the size of this component. If the sets maintained the coloring pattern, we obtained a flat distribution of several smaller distances. Complete sampling of the full sequence resulted in similar peaks, however with a shift towards higher distances, as all the smaller connected components are also resampled at every move. The peaks at higher distances show a more even distribution for the same reason. In the analyzed examples, no decomposed path was longer than three nucleotides, excluding special vertices. Therefore, we only obtained hamming distances between 0 and 3 with the local approach. Sampling with a randomly picked move step resulted in a very nice superimposition of all the hamming distance distributions, see Supplementary Figure 3.

We further investigated the score change from a start sequence to its local neighborhood reachable by applying the described sampling methods, Supplementary Figure 4. This is depicted in two-dimensional density plots as score changes for the two parts of the multi state objective function (objective 1 and objective 2) at the x- and y-axis. The weighted overall score change can be obtained by following the inclined lines. The purple line indicates neighbors with a constant overall score, the scale of the actual improvement or decline can be read from the x-axis. From left to right, plots with further optimized sequences obtained from different time steps of Figure 4 were used as start sequences for the analysis. The degree of optimization is therefore measured as “number of sampled sequences”.

In the most left plot (number of sampled sequences = 0), the local neighborhood showed a quite similar distribution in terms of score improvements on objective 1 and objective 2 for any sampling method. After 100 iterations of optimization, global showed the highest number of neighbors with better scores (number in purple box), furthermore the score improvement possible for individual neighbors was highest for the global approach. Both might result from the low quality of the optimized sequences compared to the other approaches. When analyzing even more optimized sequences after 1000 steps, the global move still showed highest number of better neighbors. However, the score improvement possible was quite similar between the various methods. Only with the local approach, the score could not be substantially improved as the local minimum had almost been reached. After $5 \cdot 10^5$ iterations no better solutions could be obtained for the global and local approach as the optimization appeared to have reached a local minimum. Overall, sample local behaved similar to sample global, but reached the local minimum of the optimization much earlier due to the smaller size of the reachable neighborhood. In a local minimum no better solution could be obtained with the same move step. Only the complete approach could not reach a optimization minimum, as we sampled from the whole solution space with this move. However, better solutions could only rarely be found (see Supplementary Figure 4).

2 Supplementary Figures

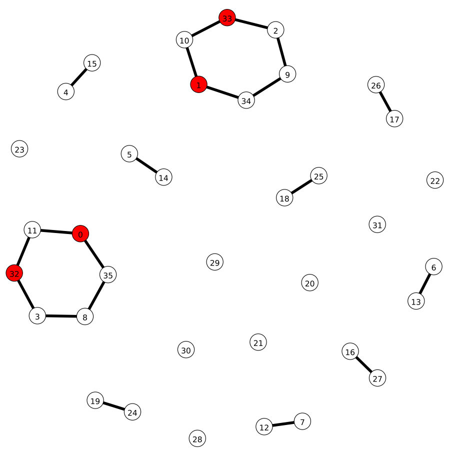


Supplementary Figure 1: The dependency graph G can be generated by the intersection of the circle plot representations of the given structural constraints. The three input structures in dot-bracket notation are first converted to the circle representation depicted on the left hand side and then the intersection is formed as shown on the right hand side. Vertices represent bases in a given order along the backbone of the molecule. Edges in different shades of gray represent the base pairs of the three input structures. Colors on the vertices show the different connected components into which the graph can be decomposed. The further decomposition and graph coloring approach of this example is shown in Figure 1 of the main text. Layouting by `VARNA`[4]

A)

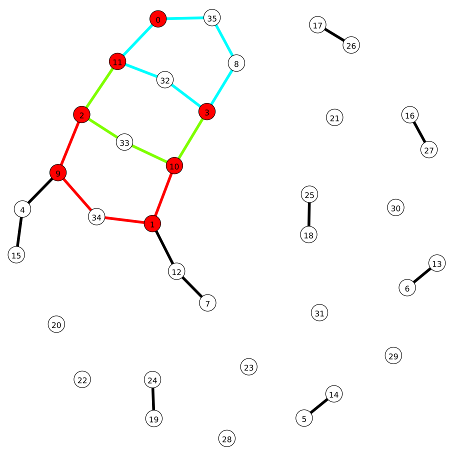
```
((((....))))....((((....))))........
........((((....((((....))))....))))
((((((((....))))((((....))))....))))

Size of solution space: 1.42658e+14
```



B)

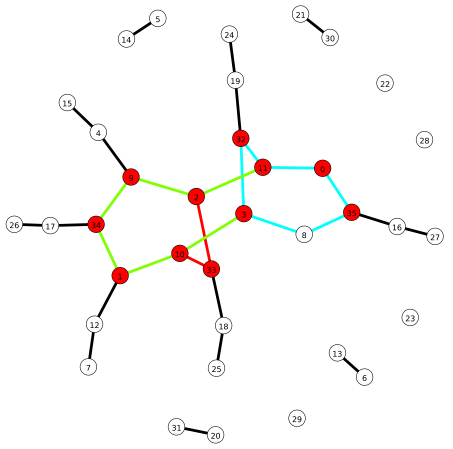
```
((((....))))....((((....))))........
........((((....((((....))))....))))
((((((((....))))((((....))))....))))
.((((....))))...((((....))))........
Size of solution space: 1.24018e+13
```



C)

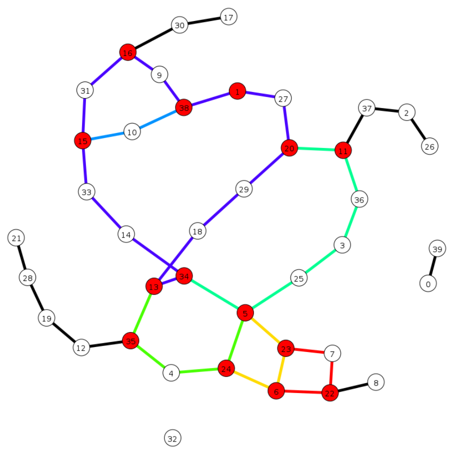
```
((((....))))....((((....))))........
........((((....((((....))))....))))
((((((((....))))((((....))))....))))
.((((....))))...((((....))))........
.((((....))))...((((((........))))))


Size of solution space: 7.08853e+10
```

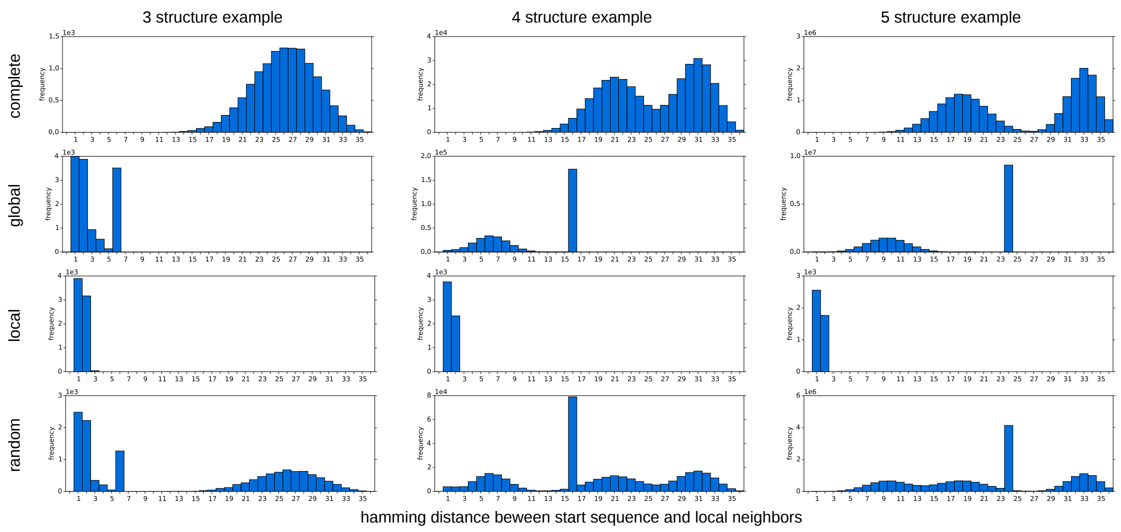


D)

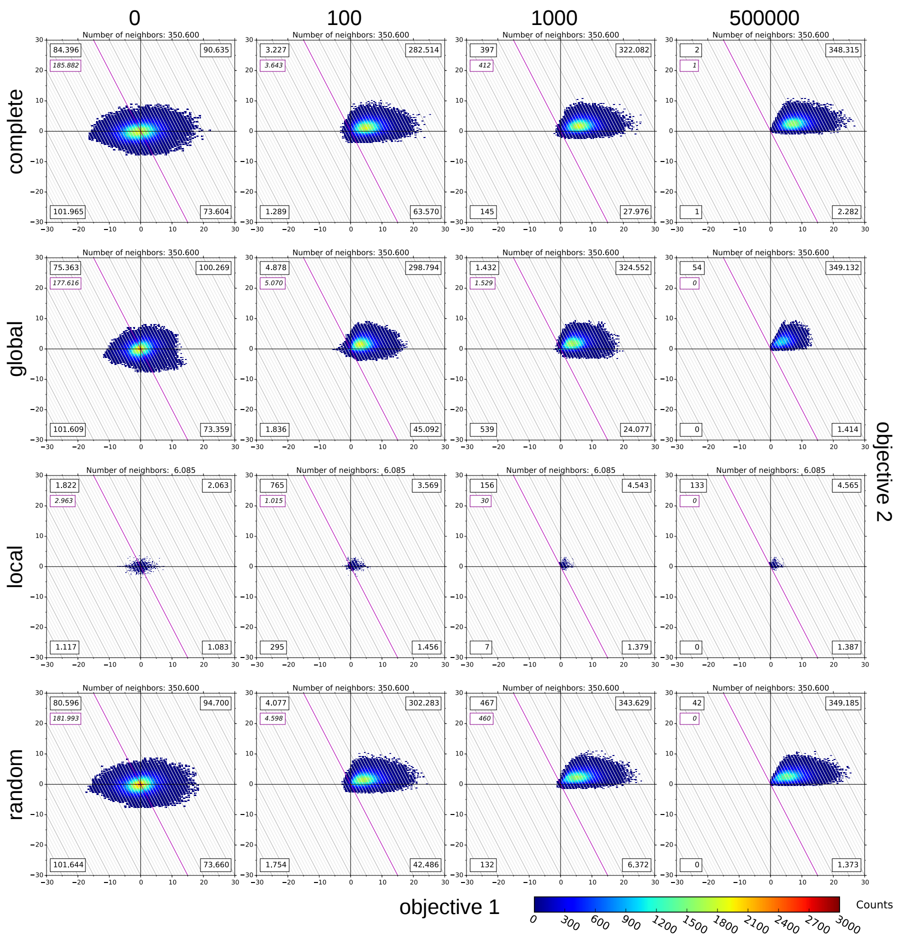
```
.((((((..((....)).....))))))............
..........((.(((....((......))...))).)).
((((((..........(((((......)))))..))))))
.........(.((((((.............)).)))).).
.....((((..(((....))).))))..............
.....(((...............)))..............
ANNNNNNNNNNNNNNNNNNNNNNNNCNNNNNNNNNNNNNN
Size of solution space: 4.14142e+07
```



Supplementary Figure 2: Example design inputs and the corresponding graph representations. The depicted examples were used to evaluate different sampling methods provided in `RNAblueprint` (A-C) as well as the fair and unfair sampling (D). The structure and sequence constraint input is shown at the top and the corresponding dependency graph at the bottom in each example. Special vertices are highlighted in red. Edges of circles and paths resulting from the ear decomposition of complex sub-graphs are colored individually.



Supplementary Figure 3: Size of different move steps measured as hamming distance between an initial start sequence and most of the reachable neighbors. Rows depict the used sampling method/move steps while columns show the three different small design examples with various complexity (see Supplementary Figure 2A-C). For global, 85% of the reachable neighborhood was sampled, while for complete and random the same absolute number was used (3str: $1.3 \cdot 10^4$, 4str: $\approx 3.5 \cdot 10^5$, 5str: $\approx 1.6 \cdot 10^7$ sequences). The local approach had a much smaller neighborhood, which was sampled with an exit condition to reach most of the neighbors (3str: 7108, 4str: 6076, 5str: $1.3 \cdot 10^4$ sequences).



Supplementary Figure 4: Relative score change to local neighborhood with various move steps on the 4 structure example Supplementary Figure 2B. Each row corresponds to a different sampling method, columns represent the neighborhood of differently optimized sequences obtained from Figure 4 in the main text. The degree of optimization is measured in “number of sampled sequences” during the optimization procedure. The density plots depict the score change to local neighbors reachable with the used move step. The score difference is split into changes of the two parts objective 1 and objective 2 on the x-axis and y-axis, respectively. The weighted overall score change to the start sequence can be obtained by the inclined lines, the purple line indicating unchanged score. The size of the neighborhood varies, for global we sampled 85% of unique neighbors and used the same absolute number for complete and random. For the local move we sampled as many unique sequences as possible in a reasonable time using an exit condition as this neighborhood is very small. The numbers in the boxes display the count of solutions in this quadrant, the purple box the absolute number of neighbors with a score change smaller than zero, meaning better solutions than the initial sequence.

3 Supplementary Tables

Supplementary Table 1: Published software to solve the inverse folding problem with single-, two- and multi-target structural input.

Name	**Initial Sequence Selection**	**Search Strategy**	**Reference**
single-target input			
`RNAinverse`	random	stochastic local search	[11]
`RNA-SSD`	random	stochastic local search	[1]
`INFO-RNA`	energy optimized	stochastic local search	[3]
`RNAexinv`	from `RNAinverse`	stochastic local search	[2]
`RNA-ensign`	random	global sampling	[16]
`IncaRNAtion`	seedless	local/global sampling	[19]
`RNAiFold`	seedless	local or global sampling	[8, 9, 10]
`DSS-Opt`	seedless	Newtonian dynamics simulation and simulated annealing	[18]
`EteRNABot`	random	stochastic local search	[15]
`NUPACK:Design`	random	stochastic local search	[5, 25, 24]
`ERD`	RNA sub-sequences of different structural elements are sampled from natural occurring RNA sequences	evolutionary algorithm	[6]
`antaRNA`	a graph that represents all possible paths to generate compatible sequences is used	ant colony based optimization	[13, 14]
two-target input			
`switch.pl`	random	stochastic local search	[7]
`RiboMaker`	random	stochastic local search	[20]
multi-target input			
`ARDesigner`	random	stochastic local search	[21]
`Frnakenstein`	random or from `RNAinverse`	genetic algorithm	[17]
`MODENA`	random	multi objective genetic algorithm	[22, 23]
`RNAdesign`	random	stochastic local search	[12]

The following tables show the benchmark results summarized in Table 1 in the manuscript. The benchmarks were adapted from Taneda [23] and calculated using the classic multi-stable design optimization approach and the weighted objective function. For the two-, three- and four-structure inputs (see Methods section in main text) we generated 100 independent solutions using the `ViennaRNA` with exit value set to 1000. For the pseudoknotted structure data sets only 30 solutions were generated using the `NUPACK` package with the exit value 100.
We used the same measures as in [23] and expanded the table by a probability value. δe_1 and δe_2 are the minimal and maximal energy difference between the evaluated energies of the target structures and the minimum free energy. The values shown in each row of the table are for the solution with the lowest δe_2 and, if multiple solutions existed, also with the the lowest δe_1. Furthermore, $n1, n2, n3, ...nM$ are the number of solutions such that $1, 2, 3, ...M$ of the target structures have the lowest free energy. We also introduced a new measure called "sum prob", which is the sum of the probabilities of all target structures in the Boltzmann ensemble for the solution picked using the δe_1 and δe_2 values.

Supplementary Table 2: Detailed results of [23] two-target design inputs (SV11 & RNAtabupath dataset)

RNA	l	n1	n2	d1	d2	μ d1	$\tilde{x}$ d1	μ d2	$\tilde{x}$ d2	μ nom	$\tilde{x}$ nom	sum prob
alpha operon	130	73	2	0.00	0.00	0.16	0.00	0.37	0.20	5759.73	5794.00	0.46
amv	145	0	0	0.30	0.40	1.10	0.80	1.31	1.00	6615.67	6256.50	0.22
attenuator	73	78	65	0.00	0.00	0.08	0.00	0.10	0.00	4217.42	4121.00	0.33
dsrA	85	0	0	0.10	0.40	2.06	2.00	2.10	2.00	3779.06	3594.50	0.12
hdv	153	64	51	0.00	0.00	0.24	0.00	0.28	0.00	6324.78	6052.00	0.12
hiv	280	10	5	0.00	0.00	1.30	1.15	1.36	1.20	17882.20	17524.00	0.01
ms2	73	53	33	0.00	0.00	0.37	0.00	0.43	0.10	3610.93	3567.00	0.12
rb1	148	81	70	0.00	0.00	0.10	0.00	0.12	0.00	7073.28	6760.50	0.07
rb2	113	24	19	0.00	0.00	0.77	0.70	0.79	0.70	5041.69	5004.00	0.12
rb3	141	61	45	0.00	0.00	0.19	0.00	0.22	0.10	7202.14	6925.00	0.07
rb4	146	0	0	2.70	2.70	5.30	5.30	5.41	5.50	7668.44	7207.50	0.00
rb5	201	67	56	0.00	0.00	0.20	0.00	0.24	0.00	10246.07	10093.00	0.12
ribD	304	0	0	1.10	1.10	2.79	2.85	2.83	2.85	15578.16	15223.50	0.04
s15	74	41	33	0.00	0.00	0.45	0.30	0.49	0.30	3760.65	3732.50	0.18
sbox	247	0	0	0.80	0.90	1.55	1.20	1.59	1.20	10602.62	10138.50	0.26
spliced	56	1	0	0.10	0.40	1.14	1.20	1.22	1.30	2548.00	2278.00	0.07
sv11	115	5	5	0.00	0.00	1.63	1.60	1.66	1.65	4195.06	4210.00	0.02
thim	165	0	0	0.50	0.60	2.13	1.95	2.18	2.00	8238.90	8046.50	0.02
μ				0.31	0.36							0.13
$\tilde{x}$				0.00	0.00							0.12

Supplementary Table 3: Detailed results of the three-target design inputs (RNAdesign dataset [3str]).

RNA	l	n1	n2	n3	d1	d2	μ d1	$\tilde{x}$ d1	μ d2	$\tilde{x}$ d2	μ nom	$\tilde{x}$ nom	sum prob
sq1	100	99	4	0	0.00	0.20	0.00	0.00	0.57	0.30	4273.57	4182.50	0.40
sq2	100	3	0	0	0.90	1.00	3.18	3.00	3.89	3.55	5388.68	5343.50	0.01
sq3	100	25	5	2	0.00	0.00	0.58	0.60	0.99	0.90	5626.87	5299.50	0.14
sq4	100	88	15	4	0.00	0.00	0.07	0.00	0.49	0.40	5046.61	4929.00	0.41
sq5	100	98	31	14	0.00	0.00	0.00	0.00	0.40	0.30	4222.47	4117.50	0.67
sq6	100	37	3	0	0.00	0.20	0.45	0.30	1.56	1.30	6197.26	5751.00	0.11
sq7	100	20	6	0	0.00	0.10	1.02	0.70	1.33	1.15	6176.17	5838.00	0.20
sq8	100	55	10	4	0.00	0.00	0.30	0.00	0.69	0.60	5055.05	4799.00	0.24
sq9	100	84	29	13	0.00	0.00	0.05	0.00	0.31	0.20	4662.04	4517.00	0.48
sq10	100	26	3	1	0.00	0.00	0.63	0.50	1.07	1.00	4737.67	4537.50	0.22
sq11	100	63	3	0	0.00	0.10	0.17	0.00	0.62	0.50	5250.60	5119.50	0.29
sq12	100	93	13	0	0.00	0.10	0.02	0.00	0.47	0.30	4683.74	4630.50	0.16
sq13	100	9	0	0	0.20	0.20	1.40	1.40	2.10	1.90	5890.70	5594.50	0.18
sq14	100	27	0	0	0.00	0.20	0.91	0.60	1.82	1.65	5956.64	5615.00	0.04
sq15	100	2	0	0	0.00	1.20	2.99	2.90	3.90	3.75	6830.74	6464.50	0.07
sq16	100	62	20	7	0.00	0.00	0.14	0.00	0.42	0.40	5479.82	5331.50	0.26
sq17	100	15	0	0	0.00	0.80	0.75	0.70	1.82	1.70	5418.19	5177.00	0.09
sq18	100	46	4	1	0.00	0.00	0.49	0.15	1.33	1.15	5303.81	4982.50	0.25
sq19	100	5	0	0	0.40	0.50	0.89	0.80	1.35	1.30	5099.27	5102.50	0.07
sq20	100	54	12	2	0.00	0.00	0.29	0.00	0.61	0.45	5031.58	4930.00	0.16
sq21	100	37	4	0	0.00	0.10	0.66	0.30	1.67	1.60	5862.30	5632.50	0.06
sq22	100	8	0	0	0.10	0.10	1.00	1.00	1.31	1.30	7393.43	7211.50	0.15
sq23	100	0	0	0	1.00	1.60	3.33	3.20	4.56	4.50	6832.19	6520.50	0.01
sq24	100	36	9	1	0.00	0.00	0.41	0.30	0.74	0.60	5401.19	5415.00	0.18
sq25	100	98	35	15	0.00	0.00	0.00	0.00	0.29	0.20	4286.62	4164.50	0.62
sq26	100	3	0	0	0.10	0.30	0.93	0.90	1.64	1.50	4320.00	4173.00	0.10
sq27	100	3	0	0	0.00	0.30	1.20	1.00	1.56	1.45	6695.16	6579.50	0.09
sq28	100	100	0	0	0.00	0.60	0.00	0.00	0.89	0.80	5375.78	5326.50	0.29
sq29	100	19	4	0	0.00	0.20	0.78	0.75	1.22	1.20	4736.90	4452.00	0.08
sq30	100	31	6	1	0.00	0.00	0.53	0.40	0.85	0.70	5877.83	5473.50	0.08
sq31	100	24	5	2	0.00	0.00	0.54	0.40	1.11	1.00	4747.35	4483.00	0.11
sq32	100	57	12	7	0.00	0.00	0.35	0.00	0.76	0.60	5585.37	5378.50	0.05
sq33	100	79	27	16	0.00	0.00	0.06	0.00	0.30	0.30	4467.61	4247.50	0.37
sq34	100	0	0	0	1.40	1.70	3.72	3.50	4.47	4.30	6378.13	6175.50	0.00
sq35	100	33	4	0	0.00	0.10	0.62	0.35	1.10	1.00	6327.89	6326.50	0.06
sq36	100	91	4	0	0.00	0.20	0.03	0.00	0.87	0.70	5083.06	4925.50	0.29
sq37	100	70	15	4	0.00	0.00	0.12	0.00	0.50	0.40	5409.80	4857.00	0.19
sq38	100	95	9	1	0.00	0.00	0.01	0.00	0.69	0.70	5543.16	5301.50	0.48
sq39	100	11	1	0	0.00	0.10	1.49	1.45	2.21	2.15	5843.19	5736.00	0.06
sq40	100	77	1	0	0.00	1.60	0.11	0.00	1.81	1.70	5184.63	4827.50	0.12
sq41	100	23	6	1	0.00	0.00	0.65	0.60	1.09	1.05	4780.51	4789.50	0.21
sq42	100	11	0	0	0.10	1.10	1.01	0.90	2.01	1.70	5824.38	5596.50	0.18
sq43	100	73	2	0	0.00	0.10	0.15	0.00	1.23	1.20	5224.71	4969.50	0.34
sq44	100	7	3	1	0.00	0.00	1.10	1.10	1.59	1.50	5762.74	5563.00	0.11
sq45	100	65	17	2	0.00	0.00	0.22	0.00	0.57	0.40	4865.60	4726.00	0.35
sq46	100	5	2	0	0.00	0.10	0.97	0.90	1.33	1.30	5104.81	4920.50	0.11
sq47	100	98	1	0	0.00	0.70	0.01	0.00	1.80	1.70	5334.45	5198.00	0.49
sq48	100	43	2	0	0.00	0.20	0.44	0.20	1.58	1.40	5584.79	5310.00	0.17
sq49	100	42	0	0	0.00	0.20	0.36	0.10	1.00	0.90	4986.80	4666.50	0.14
sq50	100	37	0	0	0.00	0.20	0.24	0.10	1.01	0.85	5530.41	5439.00	0.35
sq51	100	3	0	0	0.00	0.60	1.86	1.70	2.51	2.30	6996.33	6806.00	0.12
sq52	100	11	4	1	0.00	0.00	1.48	1.30	2.05	1.95	6403.56	5964.50	0.21
sq53	100	95	22	14	0.00	0.00	0.00	0.00	0.22	0.20	4610.51	4421.00	0.12
sq54	100	4	0	0	0.10	0.20	1.30	1.20	1.76	1.60	7710.90	7420.00	0.11
sq55	100	2	0	0	0.20	0.50	0.96	0.90	1.60	1.50	5135.46	5055.50	0.07
sq56	100	2	1	0	0.20	0.40	1.34	1.25	1.73	1.70	5395.15	5214.50	0.11
sq57	100	100	1	0	0.00	0.20	0.00	0.00	0.57	0.40	5486.52	5293.50	0.76
sq58	100	51	6	0	0.00	0.10	0.29	0.00	1.03	0.90	5603.86	5507.00	0.28
sq59	100	10	1	1	0.00	0.00	1.07	1.10	1.43	1.40	6267.48	5950.00	0.14
sq60	100	25	5	0	0.00	0.10	0.90	0.80	1.38	1.25	6707.52	6497.50	0.04
sq61	100	11	2	0	0.10	0.30	0.49	0.40	0.98	0.85	5828.87	5593.00	0.17
sq62	100	1	0	0	0.30	0.60	2.42	2.20	3.14	3.05	7265.67	7120.00	0.04
sq63	100	100	18	9	0.00	0.00	0.00	0.00	0.31	0.30	5038.84	4908.50	0.49
sq64	100	36	10	2	0.00	0.00	0.61	0.45	1.06	0.90	5973.93	5830.00	0.10
sq65	100	11	2	0	0.00	0.10	1.48	1.25	2.13	1.80	5958.74	5797.00	0.12
sq66	100	35	6	1	0.00	0.00	0.59	0.45	1.30	1.30	5412.14	5253.00	0.22
sq67	100	0	0	0	2.70	2.80	5.82	5.75	6.71	6.60	5062.83	4907.50	0.00
sq68	100	6	0	0	0.10	0.20	1.24	1.15	1.78	1.70	5165.09	4780.50	0.03
sq69	100	0	0	0	0.40	0.70	2.17	2.15	2.76	2.70	5312.29	5054.50	0.02
sq70	100	92	4	0	0.00	0.40	0.03	0.00	0.95	0.60	5617.99	5561.50	0.44
sq71	100	31	0	0	0.80	1.30	0.50	0.40	2.90	2.80	5121.06	4929.50	0.04
sq72	100	96	32	22	0.00	0.00	0.02	0.00	0.21	0.10	4811.61	4764.00	0.23

sq73	100	90	9	2	0.00	0.00	0.06	0.00	0.57	0.50	4794.39	4528.00	0.44
sq74	100	2	0	0	0.20	0.70	1.43	1.40	2.12	1.90	4937.77	4915.50	0.07
sq75	100	15	5	0	0.00	0.10	1.02	0.90	1.38	1.35	5699.49	5654.50	0.04
sq76	100	10	4	1	0.00	0.00	1.30	1.00	1.74	1.50	5527.98	5082.00	0.07
sq77	100	8	0	0	0.20	0.30	1.24	1.00	1.87	1.60	5222.57	4755.50	0.09
sq78	100	37	0	0	0.00	0.50	0.53	0.30	2.29	2.25	5840.44	5586.50	0.13
sq79	100	6	1	0	0.00	0.10	1.33	1.20	1.77	1.60	5358.56	5199.00	0.09
sq80	100	97	7	6	0.00	0.00	0.01	0.00	0.47	0.50	5040.19	5078.50	0.41
sq81	100	72	13	6	0.00	0.00	0.09	0.00	0.53	0.40	5665.89	5465.00	0.13
sq82	100	99	29	17	0.00	0.00	0.00	0.00	0.28	0.30	4704.17	4574.00	0.37
sq83	100	49	13	0	0.00	0.10	0.26	0.10	0.80	0.60	5277.87	5093.00	0.47
sq84	100	11	2	0	0.10	0.10	0.49	0.50	0.76	0.70	4489.22	4219.00	0.20
sq85	100	85	28	16	0.00	0.00	0.05	0.00	0.26	0.20	4566.86	4433.50	0.15
sq86	100	39	12	2	0.00	0.00	0.59	0.20	1.05	1.00	5731.73	5579.50	0.16
sq87	100	13	3	0	0.50	0.70	0.57	0.50	1.89	1.80	5291.04	5183.50	0.15
sq88	100	88	16	3	0.00	0.00	0.03	0.00	0.42	0.40	5146.69	5061.00	0.37
sq89	100	96	37	9	0.00	0.00	0.02	0.00	0.30	0.20	4360.10	4162.00	0.44
sq90	100	86	23	0	0.00	0.10	0.04	0.00	0.41	0.30	5143.78	5037.50	0.40
sq91	100	61	17	3	0.00	0.00	0.22	0.00	0.57	0.50	4734.75	4702.00	0.15
sq92	100	9	0	0	0.10	0.30	0.75	0.80	1.25	1.30	5635.99	5417.50	0.06
sq93	100	60	9	3	0.00	0.00	0.16	0.00	0.50	0.40	5387.46	5207.00	0.19
sq94	100	7	0	0	0.00	0.20	1.06	0.95	1.50	1.40	5614.43	5522.50	0.15
sq95	100	10	1	1	0.00	0.00	1.49	1.40	2.13	2.05	6653.09	6352.00	0.25
sq96	100	11	0	0	0.00	0.10	0.57	0.60	1.07	1.05	5344.93	5185.50	0.35
sq97	100	23	4	0	0.00	0.10	0.73	0.60	1.24	1.10	5232.00	4990.50	0.24
sq98	100	91	0	0	0.00	0.20	0.02	0.00	0.84	0.80	5278.64	5337.50	0.36
sq99	100	29	5	4	0.00	0.00	0.39	0.40	0.65	0.70	6004.32	5776.00	0.19
sq100	100	11	3	0	0.00	0.30	1.08	1.00	1.71	1.65	5842.90	5535.50	0.07
μ					0.10	0.26							0.20
$\tilde{x}$					0.00	0.10							0.15

Supplementary Table 4: Detailed results of the four-target design inputs (RNAdesign dataset [4str]).

RNA	l	n1	n2	n3	n4	d1	d2	μ d1	$\tilde{x}$ d1	μ d2	$\tilde{x}$ d2	μ nom	$\tilde{x}$ nom	sum prob
sq1	100	100	6	0	0	0.00	0.20	0.00	0.00	1.41	1.35	4664.00	4645.50	0.32
sq2	100	0	0	0	0	3.50	4.10	6.47	6.40	8.27	8.25	5631.09	5349.50	0.00
sq3	100	5	0	0	0	0.00	1.00	2.36	2.25	4.33	4.30	8144.79	7797.00	0.03
sq4	100	47	12	3	1	0.00	0.00	0.34	0.10	1.17	1.10	5187.51	4873.00	0.38
sq5	100	90	7	0	0	0.00	0.20	0.03	0.00	0.84	0.90	4284.09	4131.00	0.61
sq6	100	25	0	0	0	0.00	0.40	0.59	0.50	1.49	1.30	5837.08	5439.00	0.02
sq7	100	33	7	1	0	0.00	0.10	0.83	0.50	1.59	1.20	6032.15	5602.00	0.21
sq8	100	3	0	0	0	0.10	1.00	2.12	1.80	3.58	3.50	5275.46	5194.50	0.02
sq9	100	87	7	1	1	0.00	0.00	0.06	0.00	0.70	0.60	5390.52	5133.00	0.16
sq10	100	14	1	0	0	0.00	0.60	1.11	1.10	2.51	2.40	5755.72	5612.50	0.13
sq11	100	13	1	0	0	0.00	0.30	0.97	1.00	1.76	1.65	5237.67	4830.50	0.28
sq12	100	90	0	0	0	0.00	0.50	0.04	0.00	1.74	1.80	4802.36	4512.00	0.11
sq13	100	1	0	0	0	1.90	2.60	4.70	4.50	6.52	6.50	4709.38	4604.00	0.00
sq14	100	19	1	0	0	0.10	2.80	1.95	1.80	5.87	5.75	6728.20	6255.00	0.03
sq15	100	0	0	0	0	0.80	2.00	5.05	5.05	6.87	6.85	6603.52	6285.00	0.01
sq16	100	76	6	0	0	0.00	0.20	0.07	0.00	0.85	0.70	5762.46	5637.50	0.23
sq17	100	4	0	0	0	0.50	1.90	1.97	1.90	4.06	4.00	6474.41	6308.50	0.01
sq18	100	11	0	0	0	0.30	0.90	1.46	1.30	3.39	3.35	6029.45	5881.50	0.04
sq19	100	2	0	0	0	0.60	3.20	3.44	3.60	6.39	6.40	6584.01	6045.00	0.00
sq20	100	10	1	0	0	0.20	0.40	1.41	1.30	2.55	2.50	6055.93	5739.50	0.03
sq21	100	0	0	0	0	2.40	4.20	5.06	4.85	7.55	6.95	6629.80	6202.00	0.00
sq22	100	5	0	0	0	0.00	1.10	1.25	1.10	2.89	2.80	6671.32	6656.00	0.16
sq23	100	0	0	0	0	0.10	1.10	2.50	2.50	4.11	4.05	7139.78	6899.00	0.02
sq24	100	30	3	1	0	0.00	0.30	0.51	0.45	1.53	1.40	6150.44	5871.00	0.11
sq25	100	83	13	3	0	0.00	0.10	0.07	0.00	0.88	0.80	4689.20	4550.50	0.39
sq26	100	29	4	0	0	0.00	0.60	0.29	0.20	1.98	1.70	4508.82	4349.00	0.06
sq27	100	10	0	0	0	0.00	0.40	1.04	0.90	1.86	1.65	6446.55	6113.00	0.15
sq28	100	77	1	0	0	0.00	0.90	0.13	0.00	3.02	3.10	5078.57	4956.50	0.06
sq29	100	58	1	1	0	0.20	0.50	0.29	0.00	2.35	2.40	4995.75	4773.00	0.02
sq30	100	30	5	0	0	0.10	0.80	0.71	0.40	2.77	2.60	6483.36	6471.00	0.03
sq31	100	3	0	0	0	0.50	0.80	1.40	1.20	3.04	2.95	6137.66	5859.50	0.06
sq32	100	50	1	0	0	0.00	0.30	0.35	0.05	1.41	1.35	5898.09	5703.50	0.11
sq33	100	42	0	0	0	0.00	0.20	0.37	0.20	1.44	1.35	5355.95	5207.00	0.34
sq34	100	0	0	0	0	2.00	2.50	4.29	4.20	6.00	6.15	6271.05	5960.00	0.00
sq35	100	12	0	0	0	0.00	1.40	1.30	1.15	3.79	3.55	8251.96	7816.50	0.13
sq36	100	95	1	0	0	0.00	0.30	0.02	0.00	1.62	1.70	5187.96	5114.50	0.20
sq37	100	1	0	0	0	1.60	3.30	3.69	3.60	6.41	5.95	5043.71	4973.00	0.00
sq38	100	65	0	0	0	0.10	2.20	0.13	0.00	2.84	2.65	5697.99	5505.00	0.24
sq39	100	15	1	0	0	0.00	0.50	1.28	1.30	2.99	2.90	5976.42	5625.00	0.07
sq40	100	57	12	0	0	0.00	1.60	0.22	0.00	2.19	2.00	5242.94	5032.00	0.50
sq41	100	35	2	0	0	0.00	1.00	0.50	0.30	3.57	3.30	5653.29	5480.00	0.08
sq42	100	7	0	0	0	0.00	1.60	2.03	1.75	4.62	4.50	8326.71	8098.00	0.03
sq43	100	27	1	0	0	0.00	0.80	0.76	0.60	3.09	2.90	6014.72	5888.00	0.08
sq44	100	9	0	0	0	0.40	0.70	1.26	1.25	2.60	2.60	5983.92	5747.00	0.05
sq45	100	21	2	0	0	0.00	0.20	0.71	0.70	1.62	1.55	4956.77	4721.00	0.41
sq46	100	0	0	0	0	0.80	1.40	2.09	2.00	3.32	3.25	5492.59	5190.50	0.01
sq47	100	75	1	0	0	0.00	0.70	0.10	0.00	2.34	2.50	5413.14	5245.50	0.28
sq48	100	3	0	0	0	0.10	2.20	2.45	2.35	5.59	5.40	7103.42	7001.00	0.02
sq49	100	58	15	0	0	0.00	0.30	0.19	0.00	1.11	0.80	5253.46	5004.50	0.23
sq50	100	31	0	0	0	0.00	0.60	0.52	0.40	2.51	2.50	5530.37	5439.50	0.13
sq51	100	0	0	0	0	4.00	5.40	6.57	6.30	10.69	10.35	5762.02	5631.00	0.00
sq52	100	4	0	0	0	0.80	2.50	1.80	1.60	5.84	5.75	5353.40	5154.50	0.01
sq53	100	76	8	0	0	0.10	0.10	0.09	0.00	1.10	1.00	6392.77	6288.00	0.09
sq54	100	1	0	0	0	0.50	1.90	3.07	3.10	4.86	4.60	6768.51	6372.00	0.02
sq55	100	2	0	0	0	0.70	1.40	1.15	1.10	2.63	2.50	7216.97	7093.00	0.07
sq56	100	1	0	0	0	1.90	2.10	2.90	3.05	4.19	4.20	5908.21	5732.50	0.00
sq57	100	57	1	0	0	0.20	0.80	0.25	0.00	2.22	2.30	5312.35	5193.00	0.27
sq58	100	6	1	0	0	0.00	0.60	1.69	1.45	2.83	2.60	6425.52	6195.00	0.05
sq59	100	7	0	0	0	1.00	1.50	2.39	2.30	4.12	3.95	7793.30	7376.00	0.02
sq60	100	1	0	0	0	0.60	1.50	3.51	3.30	4.89	4.70	7286.09	6800.50	0.03
sq61	100	8	0	0	0	0.50	0.70	1.30	1.20	2.39	2.30	5986.88	5832.50	0.03
sq62	100	9	0	0	0	0.90	1.80	1.68	1.50	4.18	4.10	6856.59	6591.50	0.01
sq63	100	91	10	1	0	0.00	0.10	0.03	0.00	0.98	0.80	7286.67	7182.50	0.27
sq64	100	1	0	0	0	0.10	0.50	2.80	2.50	4.17	4.00	5808.72	5467.00	0.06
sq65	100	18	1	0	0	0.10	1.20	1.27	1.20	3.36	3.30	6390.87	6213.50	0.11

sq66	100	21	1	0	0	0.00	0.60	0.65	0.60	1.99	1.80	6255.07	5895.00	0.07
sq67	100	4	0	0	0	1.30	2.60	3.81	3.30	6.20	6.00	5246.49	5200.00	0.00
sq68	100	4	0	0	0	1.00	1.90	2.54	2.20	5.89	5.80	6006.96	5795.00	0.00
sq69	100	0	0	0	0	1.60	3.70	6.31	6.25	8.99	8.80	4326.40	4039.50	0.00
sq70	100	84	28	1	0	0.00	0.40	0.08	0.00	1.31	1.00	5154.21	4981.50	0.24
sq71	100	83	9	0	0	0.00	1.20	0.07	0.00	3.16	3.20	4920.36	4618.50	0.22
sq72	100	16	2	0	0	0.00	0.20	0.82	0.60	1.62	1.50	5470.21	5229.50	0.13
sq73	100	14	0	0	0	0.30	1.10	1.24	0.90	2.91	2.65	7827.23	7487.50	0.05
sq74	100	5	0	0	0	0.60	1.20	1.26	1.10	3.17	3.10	7526.13	7286.50	0.10
sq75	100	27	0	0	0	0.00	0.20	0.63	0.50	2.20	2.20	6661.48	6398.00	0.09
sq76	100	2	0	0	0	0.70	1.60	2.78	2.55	4.45	4.45	6245.68	6120.50	0.04
sq77	100	2	0	0	0	0.00	0.80	2.42	2.35	3.85	3.90	5459.47	5428.00	0.03
sq78	100	6	0	0	0	2.30	3.90	2.88	2.90	7.39	7.20	5939.76	5733.00	0.00
sq79	100	9	0	0	0	1.10	2.30	1.51	1.45	5.31	5.00	5571.69	5455.50	0.00
sq80	100	99	1	0	0	0.00	0.10	0.00	0.00	0.93	0.80	4971.47	4749.00	0.29
sq81	100	4	0	0	0	1.10	2.70	2.51	2.25	6.86	6.65	6834.38	6613.50	0.01
sq82	100	82	20	5	1	0.00	0.00	0.07	0.00	0.88	0.80	5441.07	4977.00	0.38
sq83	100	29	4	0	0	0.00	0.30	0.52	0.40	1.49	1.30	5529.28	5313.50	0.37
sq84	100	17	2	0	0	0.00	0.30	0.81	0.75	1.87	1.90	5614.13	5365.50	0.17
sq85	100	0	0	0	0	1.90	2.60	4.12	3.80	6.68	6.45	5422.96	5032.00	0.00
sq86	100	11	1	0	0	0.40	0.60	1.44	1.30	2.54	2.40	6893.95	6882.50	0.06
sq87	100	44	2	0	0	0.40	1.20	0.37	0.15	3.07	3.05	5224.05	5011.00	0.08
sq88	100	25	4	0	0	0.10	1.50	0.67	0.45	3.25	3.10	5241.55	4913.00	0.13
sq89	100	61	6	2	0	0.00	0.10	0.19	0.00	1.22	1.15	4720.24	4575.00	0.08
sq90	100	78	5	0	0	0.00	0.50	0.08	0.00	2.49	2.50	5720.69	5516.50	0.16
sq91	100	51	3	1	0	0.00	0.20	0.26	0.00	1.54	1.40	5793.00	5592.00	0.11
sq92	100	0	0	0	0	1.90	2.50	4.22	3.75	7.45	7.25	6379.51	6355.50	0.00
sq93	100	4	0	0	0	0.00	1.20	1.70	1.60	4.54	4.40	7232.49	7114.00	0.04
sq94	100	38	5	0	0	0.00	0.20	0.33	0.20	1.18	1.10	5789.17	5573.00	0.23
sq95	100	1	0	0	0	3.30	3.90	6.13	5.70	8.34	8.10	5024.56	4885.00	0.00
sq96	100	9	1	0	0	0.40	0.80	0.81	0.70	2.28	2.10	5934.80	5825.00	0.08
sq97	100	56	0	0	0	0.00	0.90	0.32	0.00	2.47	2.50	5535.12	5305.50	0.14
sq98	100	92	0	0	0	0.00	0.50	0.03	0.00	2.11	2.10	5873.75	5611.50	0.08
sq99	100	24	11	1	1	0.00	0.00	0.43	0.40	0.71	0.70	5705.84	5430.50	0.25
sq100	100	1	0	0	0	2.20	2.60	2.62	2.45	5.43	5.20	6302.11	5966.00	0.00
μ				0	0	0.48	1.21							0.11
$\tilde{x}$				0	0	0.05	0.80							0.07

Supplementary Table 5: Detailed results of two-target pseudknot design inputs (LE80 dataset).

RNA	l	n1	n2	d1	d2	μ d1	$\tilde{x}$ d1	μ d2	$\tilde{x}$ d2	μ nom	$\tilde{x}$ nom	sum prob
PKB00002 PKB00004 0	50	11	5	0.00	0.00	0.57	0.15	0.71	0.35	635.77	638.50	0.24
PKB00005 PKB00015 0	41	0	0	0.60	0.60	1.42	1.40	1.63	1.70	568.57	534.50	0.10
PKB00008 PKB00031 0	40	0	0	0.20	0.40	1.34	1.10	1.67	1.40	656.43	579.50	0.19
PKB00010 PKB00066 0	40	12	5	0.00	0.00	0.52	0.20	0.70	0.45	593.07	573.50	0.38
PKB00012 PKB00268 0	40	8	4	0.00	0.00	0.85	0.65	0.95	0.70	453.80	399.00	0.05
PKB00030 PKB00045 0	41	0	0	0.70	0.90	1.64	1.40	1.85	1.60	588.83	529.00	0.19
PKB00047 PKB00069 0	61	0	0	3.00	3.30	5.61	5.45	5.80	5.70	487.47	449.00	0.00
PKB00048 PKB00265 0	61	0	0	1.10	1.20	3.40	3.10	6670.27	3.65	517.87	452.00	0.01
PKB00050 PKB00128 0	59	10	4	0.00	0.00	0.65	0.40	0.93	0.75	518.03	481.50	0.15
PKB00052 PKB00107 0	52	4	1	0.00	0.00	6667.73	0.85	6667.99	1.15	437.93	385.50	0.25
PKB00057 PKB00072 0	67	0	0	1.30	2.10	5.41	5.30	5.69	5.50	495.33	436.00	0.00
PKB00068 PKB00129 0	68	0	0	2.60	3.30	5.34	5.10	5.65	5.45	681.17	640.50	0.00
PKB00070 PKB00244 0	55	2	0	0.00	0.10	1.91	1.75	2.32	1.90	481.70	394.50	0.17
PKB00078 PKB00106 0	62	4	0	0.00	0.10	1.17	1.00	2.25	1.90	525.20	466.50	0.24
PKB00080 PKB00132 0	49	10	4	0.00	0.00	0.50	0.40	0.70	0.60	407.60	407.50	0.07
PKB00088 PKB00127 0	62	10	1	0.00	0.00	0.74	0.55	1.54	1.15	676.53	658.00	0.25
PKB00098 PKB00232 0	62	1	0	0.00	0.40	2.53	2.65	2.80	2.80	609.90	580.00	0.04
PKB00131 PKB00205 0	48	0	0	1.70	2.20	3.02	3.00	4.16	3.90	570.73	567.50	0.01
PKB00139 PKB00141 0	70	0	0	1.50	1.60	3.17	3.00	3.27	3.20	745.73	662.00	0.01
PKB00142 PKB00231 0	71	1	0	0.00	0.80	2.86	2.55	3.33	2.90	498.97	468.50	0.05
PKB00143 PKB00233 0	71	0	0	1.40	1.50	13337.10	3.70	13337.27	3.75	603.03	577.00	0.01
PKB00148 PKB00218 0	72	0	0	3.30	3.60	5.52	4.95	5.77	5.10	642.50	571.00	0.00
PKB00175 PKB00259 0	57	0	0	0.30	0.50	1.69	1.65	1.98	1.90	648.77	643.00	0.08
PKB00179 PKB00280 0	68	0	0	0.60	0.60	2.61	2.70	2.90	2.90	565.13	587.50	0.01
PKB00180 PKB00212 0	64	0	0	0.30	0.40	6670.49	3.05	20003.69	3.65	448.53	402.50	0.13
PKB00190 PKB00266 0	47	21	7	0.00	0.00	0.18	0.00	0.34	0.20	534.43	530.00	0.29
PKB00207 PKB00213 0	45	7	1	0.00	0.00	13334.10	0.60	13334.26	0.85	364.37	339.50	0.26
PKB00211 PKB00239 0	80	0	0	0.80	1.10	4.15	3.75	4.62	4.50	486.80	464.00	0.02
PKB00222 PKB00305 0	80	0	0	0.50	0.90	6670.15	3.35	6670.54	3.85	595.60	592.00	0.02
PKB00224 PKB00281 0	43	9	3	0.00	0.00	0.70	0.55	1.01	0.75	513.63	485.00	0.19
PKB00230 PKB00273 0	48	0	0	2.00	2.50	4.11	4.10	6671.29	4.65	374.43	353.00	0.00
PKB00248 PKB00257 0	66	0	0	4.40	6.70	6675.06	8.55	33343.25	10.90	214.17	221.00	0.00
PKB00263 PKB00270 0	62	6	1	0.00	0.00	13334.31	0.95	13334.51	1.15	620.23	629.50	0.16
PKB00269 PKB00272 0	66	0	0	1.50	2.30	6670.63	3.80	20004.32	4.40	444.57	411.00	0.00
μ				0.82	1.09							0.11
$\tilde{x}$				0.30	0.55							0.06

Supplementary Table 6: Detailed results of two-target pseudknot design inputs (PK60 dataset).

RNA	l	n1	n2	d1	d2	μ d1	$\tilde{x}$ d1	μ d2	$\tilde{x}$ d2	μ nom	$\tilde{x}$ nom	sum prob
no1	60	0	0	0.20	0.20	1.64	1.35	1.82	1.60	620.13	659.00	0.11
no2	60	2	0	0.20	0.30	1.48	1.55	1.87	1.70	1015.93	1008.50	0.06
no3	60	17	5	0.00	0.00	0.39	0.00	0.65	0.20	806.70	707.00	0.12
no4	60	11	5	0.00	0.00	0.58	0.20	0.70	0.30	810.63	769.00	0.14
no5	60	3	1	0.00	0.00	1.20	1.30	1.41	1.40	696.63	686.50	0.08
no6	60	2	0	0.00	0.30	1.16	0.95	1.40	1.30	787.03	736.00	0.15
no7	60	29	18	0.00	0.00	0.04	0.00	0.11	0.00	929.33	897.50	0.45
no8	60	0	0	0.30	0.30	2.50	2.30	2.72	2.60	647.67	651.00	0.06
no9	60	27	11	0.00	0.00	0.03	0.00	0.16	0.10	908.17	887.00	0.27
no10	60	21	9	0.00	0.00	0.15	0.00	0.31	0.10	944.97	917.00	0.39
no11	60	27	15	0.00	0.00	0.05	0.00	0.17	0.05	862.27	806.00	0.41
no12	60	17	3	0.00	0.00	0.38	0.00	0.61	0.55	779.47	779.00	0.33
no13	60	8	1	0.00	0.00	0.45	0.25	0.56	0.40	784.07	791.00	0.17
no14	60	28	14	0.00	0.00	0.03	0.00	0.11	0.10	852.70	798.50	0.28

no15	60	3	1	0.00	0.00	1.31	1.25	1.57	1.50	615.13	595.00	0.18
no16	60	7	2	0.00	0.00	0.60	0.45	0.77	0.55	729.23	736.00	0.17
no17	60	0	0	0.20	0.50	1.98	1.90	2.17	2.15	657.07	619.00	0.06
no18	60	25	7	0.00	0.00	0.15	0.00	0.32	0.20	719.77	717.50	0.46
no19	60	14	3	0.00	0.00	0.62	0.10	0.89	0.55	784.93	683.50	0.27
no20	60	4	0	0.00	0.10	1.14	0.85	1.44	1.30	652.77	640.50	0.19
no21	60	3	1	0.00	0.00	1.71	1.70	1.94	1.85	592.00	576.50	0.09
no22	60	23	7	0.00	0.00	0.10	0.00	0.27	0.20	880.83	885.50	0.31
no23	60	30	15	0.00	0.00	0.00	0.00	0.11	0.05	881.77	860.50	0.75
no24	60	3	0	0.00	0.20	1.21	1.20	1.43	1.35	532.30	486.50	0.37
no25	60	20	15	0.00	0.00	0.13	0.00	0.18	0.05	841.13	822.00	0.27
no26	60	28	11	0.00	0.00	0.04	0.00	0.14	0.10	845.93	900.50	0.44
no27	60	7	1	0.00	0.00	1.44	1.30	1.92	1.45	633.67	625.00	0.16
no28	60	27	4	0.00	0.00	0.02	0.00	0.26	0.15	902.00	877.50	0.45
no29	60	17	5	0.00	0.00	0.36	0.00	0.50	0.30	639.13	604.50	0.06
no30	60	22	6	0.00	0.00	0.15	0.00	0.30	0.20	672.63	652.00	0.14
no31	60	1	0	0.00	0.20	1.84	1.25	2.06	1.55	696.03	704.00	0.07
no32	60	23	3	0.00	0.00	0.14	0.00	0.30	0.20	905.87	876.00	0.32
no33	60	12	4	0.00	0.00	0.47	0.40	0.63	0.55	652.57	589.00	0.41
no34	60	9	4	0.00	0.00	0.75	0.50	0.86	0.55	528.40	475.50	0.14
no35	60	21	8	0.00	0.00	0.21	0.00	0.39	0.40	844.40	802.00	0.46
no36	60	9	1	0.00	0.00	0.59	0.40	0.83	0.55	805.07	812.00	0.33
no37	60	0	0	0.60	1.00	1.88	1.80	2.16	2.00	937.03	1000.50	0.03
no38	60	1	1	0.00	0.00	1.55	1.25	1.79	1.55	647.67	640.00	0.24
no39	60	30	18	0.00	0.00	0.00	0.00	0.06	0.00	865.63	862.00	0.48
no40	60	24	15	0.00	0.00	0.13	0.00	0.27	0.05	1026.00	1072.00	0.27
no41	60	22	9	0.00	0.00	0.13	0.00	0.34	0.10	810.83	743.00	0.22
no42	60	26	7	0.00	0.00	0.09	0.00	0.22	0.10	885.23	835.00	0.45
no43	60	2	0	0.00	0.80	2.10	1.90	2.52	2.40	539.30	485.00	0.05
no44	60	29	9	0.00	0.00	0.01	0.00	0.15	0.10	1038.80	1084.00	0.46
no45	60	13	4	0.00	0.00	0.76	0.35	1.01	0.60	828.50	831.00	0.12
no46	60	4	1	0.00	0.00	1.51	1.10	1.78	1.60	437.30	422.50	0.05
no47	60	2	0	0.00	0.10	1.90	1.80	2.14	2.05	645.33	591.00	0.03
no48	60	0	0	0.10	0.10	1.52	1.30	1.72	1.60	695.47	681.00	0.12
no49	60	18	7	0.00	0.00	0.17	0.00	0.29	0.15	878.80	829.00	0.32
no50	60	18	7	0.00	0.00	0.35	0.00	0.53	0.25	666.67	657.50	0.10
μ				0.03	0.08							0.24
$\tilde{x}$				0.00	0.00							0.21

Supplementary Table 7: Detailed results of two-target pseudknot design inputs (PK80 dataset).

RNA	l	n1	n2	d1	d2	μ d1	$\tilde{x}$ d1	μ d2	$\tilde{x}$ d2	μ nom	$\tilde{x}$ nom	sum prob
no1	80	3	2	0.00	0.00	1.13	1.00	1.31	1.30	871.63	915.00	0.09
no2	80	23	15	0.00	0.00	0.15	0.00	0.20	0.05	846.37	799.00	0.17
no3	80	28	4	0.00	0.00	0.05	0.00	0.19	0.10	1178.70	1119.50	0.22
no4	80	22	8	0.00	0.00	0.18	0.00	0.34	0.10	1007.97	940.00	0.17
no5	80	12	7	0.00	0.00	0.64	0.25	0.74	0.40	1079.90	1108.50	0.27
no6	80	25	18	0.00	0.00	0.20	0.00	0.25	0.00	971.53	933.50	0.10
no7	80	5	1	0.00	0.00	0.96	0.80	1.12	0.95	833.23	816.00	0.12
no8	80	1	0	0.00	0.20	1.80	1.45	2.00	1.70	847.50	803.00	0.14
no10	80	20	8	0.00	0.00	0.27	0.00	0.38	0.15	925.50	929.00	0.26
no11	80	29	10	0.00	0.00	0.01	0.00	0.15	0.10	1083.83	1006.00	0.21
no12	80	30	20	0.00	0.00	0.00	0.00	0.06	0.00	1006.93	959.00	0.42
no13	80	30	16	0.00	0.00	0.00	0.00	0.08	0.00	1114.80	1123.00	0.41
no14	80	10	3	0.00	0.00	0.59	0.55	0.77	0.65	1087.57	1057.00	0.22
no15	80	25	5	0.00	0.00	0.08	0.00	0.18	0.10	1098.63	1064.50	0.14
no16	80	27	9	0.00	0.00	0.02	0.00	0.18	0.10	971.37	941.50	0.31
no17	80	8	4	0.00	0.00	0.62	0.55	0.73	0.65	1064.17	1107.50	0.12
no18	80	18	8	0.00	0.00	0.29	0.00	0.39	0.20	842.50	760.50	0.10
no19	80	29	13	0.00	0.00	0.00	0.00	0.08	0.10	1164.00	1130.50	0.24
no20	80	17	7	0.00	0.00	0.31	0.00	0.40	0.20	1178.23	1212.50	0.24
no21	80	1	0	1.10	1.30	6671.48	4.45	6671.74	4.60	707.30	652.00	0.01
no22	80	7	1	0.00	0.00	0.85	0.60	1.09	0.85	1144.57	1137.50	0.25
no23	80	29	13	0.00	0.00	0.04	0.00	0.18	0.10	1181.20	1121.00	0.53
no24	80	7	2	0.00	0.00	0.98	0.60	1.16	0.75	835.43	815.00	0.29
no25	80	7	2	0.00	0.00	1.04	1.10	1.43	1.40	776.83	709.50	0.31
no26	80	0	0	3.30	3.40	5.81	5.55	6.51	5.95	531.07	508.50	0.00
no27	80	1	0	0.00	0.20	2.92	2.65	3.37	3.30	979.33	905.50	0.12
no28	80	24	12	0.00	0.00	0.15	0.00	0.34	0.10	1146.77	1176.50	0.10
no29	80	21	7	0.00	0.00	0.18	0.00	0.31	0.15	989.97	974.50	0.12
no30	80	14	3	0.00	0.00	0.30	0.10	0.58	0.35	965.67	980.00	0.22
np9	80	20	14	0.00	0.00	0.29	0.00	0.34	0.10	931.57	895.50	0.19
μ				0.15	0.17							0.20
$\tilde{x}$				0.00	0.00							0.20

References

[1] Mirela Andronescu, Anthony P Fejes, Frank Hutter, Holger H Hoos, and Anne Condon. A new algorithm for RNA secondary structure design. *J Mol Biol*, 336(3):607–624, Feb 2004.

[2] Assaf Avihoo, Alexander Churkin, and Danny Barash. RNAexinv: An extended inverse rna folding from shape and physical attributes to sequences. *BMC Bioinformatics*, 12(1):319, 2011.

[3] Anke Busch and Rolf Backofen. INFO-RNA – a fast approach to inverse RNA folding. *Bioinformatics*, 22(15), August 2006.

[4] Kvin Darty, Alain Denise, and Yann Ponty. VARNA: Interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15):1974–1975, Aug 2009.

[5] Robert M Dirks, Milo Lin, Erik Winfree, and Niles A Pierce. Paradigms for computational nucleic acid design. *Nucleic Acids Res*, 32(4):1392–1403, 2004.

[6] A. Esmaili-Taheri, M. Ganjtabesh, and M. Mohammad-Noori. Evolutionary solution for the RNA design problem. *Bioinformatics*, 30(9):1250–1258, Jan 2014.

[7] Christoph 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.

[8] Juan Antonio Garcia-Martin, Peter Clote, and Ivan Dotu. RNAiFOLD: a constraint programming algorithm for rna inverse folding and molecular design. *J Bioinform Comput Biol*, 11(2):1350001, Apr 2013.

[9] Juan Antonio Garcia-Martin, Peter Clote, and Ivan Dotu. RNAiFold: a web server for rna inverse folding and molecular design. *Nucleic Acids Res*, 41(Web Server issue):W465–W470, Jul 2013.

[10] Juan Antonio Garcia-Martin, Ivan Dotu, Javier Fernandez-Chamorro, Gloria Lozano, Jorge Ramajo, Encarnacion Martinez-Salas, and Peter Clote. RNAiFold2T: Constraint programming design of thermo-IRES switches. *Bioinformatics*, 32(12):i360–i368, jun 2016.

[11] 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 / Chemical Monthly*, 125(2):167–188, February 1994.

[12] Christian Höner zu Siederdissen, Stefan Hammer, Ingrid Abfalter, Ivo L. Hofacker, Christoph Flamm, and Peter F. Stadler. Computational design of RNAs with complex energy landscapes. *Biopolymers*, 99(12):1124–1136, 2013.

[13] Robert Kleinkauf, Torsten Houwaart, Rolf Backofen, and Martin Mann. antaRNA – Multi-objective inverse folding of pseudoknot RNA using ant-colony optimization. *BMC Bioinformatics*, 16, 2015.

[14] Robert Kleinkauf, Martin Mann, and Rolf Backofen. antaRNA: ant colony-based RNA sequence design. *Bioinformatics*, 31(19):31143121, May 2015.

[15] Jeehyung Lee, Wipapat Kladwang, Minjae Lee, Daniel Cantu, Martin Azizyan, Hanjoo Kim, Alex Limpaecher, Sungroh Yoon, Adrien Treuille, and Rhiju Das. RNA design rules from a massive open laboratory. *Proceedings of the National Academy of Sciences*, pages 2122–2127, Jan 2014.

[16] Alex Levin, Mieszko Lis, Yann Ponty, Charles W. O'Donnell, Srinivas Devadas, Bonnie Berger, and Jérôme Waldispühl. A global sampling approach to designing and reengineering RNA secondary structures. *Nucleic Acids Res*, 40(20):10041–10052, Nov 2012.

[17] Rune B Lyngso, James WJ Anderson, Elena Sizikova, Amarendra Badugu, Tomas Hyland, and Jotun Hein. Frnakenstein: multiple target inverse RNA folding. *BMC Bioinformatics*, 13(1):260, 2012.

[18] Marco C. Matthies, Stefan Bienert, and Andrew E. Torda. Dynamics in sequence space for RNA secondary structure design. *Journal of Chemical Theory and Computation*, 8(10):3663–3670, Oct 2012.

[19] Vladimir Reinharz, Yann Ponty, and Jérôme Waldispühl. A weighted sampling algorithm for the design of RNA sequences with targeted secondary structure and nucleotide distribution. *Bioinformatics*, 29(13), Jul 2013.

[20] G. Rodrigo and A. Jaramillo. RiboMaker: computational design of conformation-based riboregulation. *Bioinformatics*, 30(17):2508–2510, may 2014.

[21] Wenjie Shu, Ming Liu, Hebing Chen, Xiaochen Bo, and Shengqi Wang. ARDesigner: a web-based system for allosteric RNA design. *J Biotechnol*, 150(4):466–473, Dec 2010.

[22] Akito Taneda. MODENA: a multi-objective RNA inverse folding. *Adv Appl Bioinform Chem*, 4:1–12, 2011.

[23] Akito Taneda. Multi-objective optimization for RNA design with multiple target secondary structures. *BMC Bioinformatics*, 16(1):280, September 2015.

[24] Brian R. Wolfe and Niles A. Pierce. Sequence Design for a Test Tube of Interacting Nucleic Acid Strands. *ACS Synthetic Biology*, 4(10):1086–1100, October 2015.

[25] Joseph N. Zadeh, Brian R. Wolfe, and Niles A. Pierce. Nucleic acid sequence design via efficient ensemble defect optimization. *Journal of Computational Chemistry*, 32(3):439–452, 2011.