A New Algorithm for RNA Secondary Structure Design ⋆

Mirela Andronescu [a], Anthony P. Fejes [b], Frank Hutter [a],
Anne Condon [a], Holger H. Hoos [a,∗]

[a]*Department of Computer Science*
University of British Columbia
Vancouver, BC, Canada

[b]*Department of Microbiology and Immunology*
University of British Columbia
Vancouver, BC, Canada

Abstract

The function of many RNAs crucially depends on their structure. Therefore, the design of RNA molecules with specific structural properties has many potential applications, *e.g.*, in the context of investigating the function of biological RNAs, of creating new ribozymes, or of designing artificial RNA nanostructures. Here, we present a new algorithm for solving the following RNA secondary structure design problem: Given a secondary structure, find an RNA sequence (if any) that is predicted to fold to that structure. Unlike the (pseudoknot-free) secondary structure prediction problem, this problem appears to be computationally hard. Our new algorithm, "RNA Secondary Structure Designer (RNA-SSD)", is based on stochastic local search, a prominent general approach for solving hard combinatorial problems. A thorough empirical evaluation on computationally predicted structures of biological sequences and artificially generated RNA structures as well as on empirically modelled structures from the biological literature shows that RNA-SSD substantially outperforms the best known algorithm for this problem, RNAinverse from the Vienna RNA Package. In particular, the new algorithm is consistently able to solve structures for which RNAinverse is unable to find solutions. The RNA-SSD software is publicly available under the name of RNA Designer at www.rnasoft.ca [1].

Key words: RNA structure, RNA design, RNA inverse folding, stochastic local search

⋆ This material is based upon work supported by the U.S. National Science Foundation under Grant No. 0130108, and by the National Sciences and the Engineering Research Council of Canada.

∗ Corresponding author. Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver, BC, V6T 1Z4, Canada.

Email addresses: `andrones@cs.ubc.ca` (Mirela Andronescu),

1 Introduction

RNA molecules play many roles in the cell that go far beyond acting as intermediaries in the translation of genomic DNA into proteins. For example, ribosomal and transfer RNAs are crucial components of the translation machinery, and various types of small RNAs have important functions in gene expression. Catalytic RNA molecules, called ribozymes, can cleave other RNA molecules; these and other RNAs may have played a role in early evolution before the more complex proteins were evolved. The structure of RNA molecules is crucial to their function in these and many other cellular processes, and indeed computational approaches for predicting RNA secondary and tertiary structure are widely used by biologists.

In this paper, we focus on the inverse problem to that of predicting RNA secondary structure, namely design of RNA molecules with a desired secondary structure. One motivation for this work is that novel ribozymes may provide new paths to the design of drugs [2], or have industrial uses [3]. Another motivation stems from the use of carefully designed DNA structures (which are in many ways analogous to RNA structures) in DNA self-assembly computation [4]. Finally, the ability to design RNA molecules with specific structural features can play a crucial role in research on the function of natural RNAs.

Before describing our problem and approach in more detail, we note briefly that a secondary structure for an RNA strand is simply a set of pairing interactions between bases in the strand. Each base can be paired with at most one other base. Most base pairings occur between Watson-Crick complementary bases C and G or A and U, respectively (canonical pairs). Other pairings, such as G-U can be found occasionally.

Computational approaches for prediction of RNA secondary structure are based on a thermodynamic model that associates a free energy value with each possible secondary structure for a strand. The secondary structure with the lowest possible free energy value, the so-called minimum free energy (MFE) structure, is predicted to be the most stable secondary structure for the strand. (For an example of an RNA secondary structure and the calculation of its free energy, see Figure 1.) Although for a given RNA strand, the number of secondary structures can be exponential in the length of the strand, RNA secondary structure prediction appears to be easier than protein secondary structure prediction, at least for the class of so-called pseudoknot-free secondary structures (see Section 2 for the definition of pseudoknot-free structures). There are widely used dynamic programming algorithms that, given an RNA strand of length n, find in $\Theta(n^3)$ time the secondary structure with the lowest free energy, from the class of pseudoknot-free secondary structures [5]. Throughout the paper, all references to secondary structures refer to

fejes@interchange.ubc.ca (Anthony P. Fejes), mail@fhutter.de (Frank Hutter), condon@cs.ubc.ca (Anne Condon), hoos@cs.ubc.ca (Holger H. Hoos).

pseudoknot-free secondary structures.

The RNA design problem that we consider here is as follows: given a secondary structure, find an RNA strand (if any), that folds to that structure. This can be seen as a discrete constraint satisfaction problem where the constraint variables are the positions in the desired RNA strand, the values assigned to these variables correspond to the bases at the respective positions, and the constraints capture the base pairings that define the given secondary structure. Although its complexity is unknown, this RNA secondary structure design problem appears to be computationally hard, hence a heuristic approach is appropriate. We note that for evaluating a candidate base assignment with respect to the given secondary structure constraints, the MFE secondary structure for the respective candidate RNA strand needs to be determined. The fact that this operation requires $\Theta(n^3)$ time poses a challenge for any heuristic search approach.

We present a new algorithm, RNA-SSD (RNA Secondary Structure Designer), that designs RNA strands for input secondary structures. At the core of our algorithm is a stochastic local search (SLS) procedure that iteratively modifies single unpaired bases or base pairs of a candidate strand in order to obtain a strand that folds into the target structure.

Since each step of the SLS algorithm requires a call to a $\Theta(n^3)$-time evaluation function, a key component of our approach is a hierarchical decomposition of the input secondary structure into small substructures. The core SLS algorithm is only applied to the smallest substructures, and the corresponding partial solutions are combined into candidate solutions for larger subproblems guided by the decomposition tree. Since the subproblems are not independent, this does not always result in valid designs for the corresponding substructure. Consequently, multiple attempts (involving additional calls to the core SLS procedure) are often required before partial solutions can be successfully combined.

The other key component of the algorithm is a method for generating a good initial design for the RNA strand. This initialisation procedure assigns bases probabilistically to the strand, using different probabilistic models for base positions that are paired and unpaired in the target structure. In addition, the algorithm ensures that complementary stretches of bases are avoided across the design, except where desired along two sides of a stem.

We empirically evaluated our RNA-SSD algorithm on randomly generated structures, on computationally predicted structures (pseudoknot-free) of naturally occurring sequences from the Ribosomal Database Project (RDP), and on biological structures reported in the literature. In addition, we compared the performance of our algorithm with RNAinverse, a previous algorithm of Hofacker et al. [6] for RNA strand design which is part of the widely used Vienna RNA Secondary Structure Package. The results of this empirical evaluation show that RNA-SSD substantially outperforms RNAinverse on a broad range of structures. RNA-SSD found solutions to all 24 computationally predicted structures of naturally occurring sequences (length: 260–1475 bases) and 8 out of 10 biological structures (length: 65–583 bases), while the Vienna algorithm, RNAinverse, solved only 12 of these 34 structures (these 12 are all of length less than 500 bases). RNA-SSD solved 419 of the 420 artificial structures, while RNAinverse solved only 378. For all but 8 out of 378 structures designed by both algorithms, RNA-SSD is substantially faster (at least one order of magnitude in about 79% of the cases, at least two orders of magnitude in about 48% of the cases). Furthermore, we observed that the solutions found by RNA-SSD for the biological structures are typically more stable (in terms of their minimum free energy and partition function calculation) than the original biological sequences.

The rest of this paper is organised as follows. In Section 2, we give some background on RNA secondary structure prediction, define the problem of RNA strand design formally, and describe some previous work on this problem. Our algorithm is described in detail in Section 3. In Sections 4 and 5, we present our empirical analysis and performance results, and provide a discussion of these, along with some further results that refine and support our analysis, in Section 6. Conclusions and future work are described in Section 7.

2 The RNA Secondary Structure Design Problem

An RNA strand consists of a sugar phosphate backbone to which the four bases cytosine (C), guanine (G), adenine (A), and uracil (U) are attached. Each strand has two chemically distinct ends, known as the $5'$ and $3'$ ends. Simple Watson-Crick base pairing involves the bonding of C with G and A with U via three or two hydrogen bonds, respectively. Additionally, so-called wobble pairs can form between G and U. For a given RNA strand, a secondary structure describes which bases are paired. Specifically, the secondary structure of a strand of length n is a set of pairs (i, j), where i and j are in the range $[1, \ldots, n]$, and (i, j) represents a pairing between the ith and jth bases in the strand where the bases in the strand are indexed from 1 to n starting at the 5' end. In a secondary structure, each base has at most one partner. Base pairs are most often found stacked onto other base pairs in substructures called *stems* or *helices*. Sometimes, unpaired bases are interspersed in stems; these are known as *internal loops* or *bulges*. Loops occurring at the ends of stems are called *hairpins*, and loops from which more than two stems originate are known as *multi-branched loops*, or simply *multiloops*. Figure 1 shows a standard display of a secondary structure for an RNA strand, in which the stems and loops are apparent. This particular secondary structure is *pseudoknot-free*; that is, it does not have any two base pairs (i, j) and (i', j') where $i < i' < j < j'$.

Associated with a secondary structure for a strand is its *free energy*. For pseudoknot-

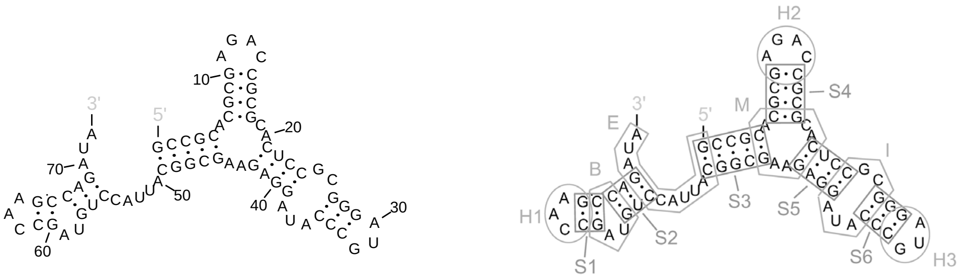

Fig. 1. Left: Graphical depiction of the predicted minimum free energy secondary structure S for sequence GCCGCACGCGAGACCGCGCACUCCGCGGGAUGCCCAUAGGAGAAGCGGCAUUACCUGUAGCCAAGCCAGAUA. Right: The free energy for structure S is computed from contributions of the labelled stems ($S1$–$S6$) and loops (hairpins $H1$–$H3$, bulge B, internal loop I, external loop E, and multiloop M) as $\Delta G(S) = \Delta G(\mathrm{H1}) + \Delta G(\mathrm{S1}) + \Delta G(\mathrm{B}) + \Delta G(\mathrm{S2}) + \Delta G(\mathrm{E}) + \Delta G(\mathrm{S3}) + \ldots + \Delta G(\mathrm{S6})$ $+\Delta G(\mathrm{H3}) = 4.1 - 3.4 + 3.1 - 3.5 - 3.1 - 11.7 + \ldots - 5.8 + 4.1 = -22.5$, which for this strand is the predicted minimum free energy, calculated using mfold [15].

free secondary structures, this is typically calculated as the sum of the free energies of each stacked pair and each loop [7], and estimates for these values have been experimentally determined. Let $E(X, S)$ denote the free energy of an RNA sequence X when folded into the secondary structure S. Furthermore, let Φ denote a function that assigns to each RNA sequence X a secondary structure S^* that minimises free energy $E(X, S)$ over all possible secondary structures S of X.

The **RNA secondary structure prediction problem** can be stated as follows: Given an RNA sequence X, determine $\Phi(X)$. When the energy $E(X, S)$ of an RNA sequence X folded into the secondary structure S is evaluated by the nearest neighbor thermodynamical model (not allowing for pseudoknots), the RNA secondary structure prediction problem is efficiently solvable using a dynamic programming approach that runs in time $\Theta(n^3)$, due to Lyngsø, Zuker, and Pedersen [5], a refinement of an earlier algorithm of Zuker and Stiegler. Throughout this paper, we refer to this as Zuker's algorithm.

Analogously, we can state the **RNA secondary structure design problem**: Given an RNA secondary structure S^*, find a sequence X^* s.t. $\Phi(X^*) = S^*$. In the optimisation variant of this problem, we determine the quality of a candidate solution X by comparison of the structure $S = \Phi(X)$ with the desired structure S^*; given a distance metric d, we attempt to minimise $d(S, S^*)$. In our algorithm, the distance metric will merely measure the number of bases that bond incorrectly, *i.e.*, number of bases in X whose pairing status is different between S and S^*. It is also possible to include information on the stability of folds in the distance metric.

Hofacker et al. [6] have already developed an algorithm for the RNA secondary structure design problem, RNAinverse, which is included in the Vienna RNA Secondary Structure Package[1] . The algorithm performs an “adaptive” walk search on so-called *compatible* sequences, *i.e.*, sequences that can possibly form a base-pair at the required positions in the desired structure. These sequences are candidates only; there is no guarantee that they will fold into the target structure. Starting from a randomly chosen sequence x_0, in each step the algorithm induces a mutation and accepts it if and only if the cost function decreases. These search steps are iterated until either a solution is found or a certain number of mutations has been carried out.

A drawback of this approach is that calculating the distance for each mutation involves running a folding algorithm on the sequence under observation, for which generally Zuker’s $\Theta(n^3)$ algorithm is used, where n is the length of the sequence to be folded. To counter this, the RNAinverse algorithm of Hofacker et al. applies the previously described basic RNA design procedure iteratively to substructures, and then produces a full sequence by concatenating the subsequences obtained from solving these smaller secondary structure design problems. The rationale is that it is likely (but not assured) that the substructures which are optimal for subsequences will also occur for the full sequence (see [6] for a more detailed explanation). This same idea is also underlying our new algorithm, described in the following; however, RNA-SSD differs from RNAinverse in many important aspects, including sequence initialisation, structure decomposition and sequence assembly, as well as substructure search.

3 The RNA-SSD Algorithm

Our new algorithm for the RNA secondary structure design problem is based on a stochastic local search approach that uses a probabilistic sequence initialisation heuristic, hierarchical decomposition of the given structure, and a randomised iterative improvement method for finding sequences for the resulting substructures. The space searched by our algorithm consists of RNA sequences that correspond to complete assignments of bases $x_i \in \{$A,C,G,U$\}$ to all positions i of the given RNA secondary structure. We restrict this space to sequences in which positions that are paired in the desired structure are assigned complementary bases, such as C-G or A-U.

Different from many other constraint satisfaction problems, evaluating the quality of candidate solutions for the RNA Secondary Structure Design Problem is computationally quite expensive, having time complexity $\Theta(n^3)$, where n is the length of the given sequence [5] (see Section 2). We use the *fold* function from the Vienna Package, the most efficient implementation of Zuker’s algorithm of which we are aware. Unfortunately, even a single local reassignment of a base in the se-

[1] `http://www.tbi.univie.ac.at/~ivo/RNA/`

```
procedure RNA-SSD
    input: target RNA secondary structure $S$, parameters
    output: RNA sequence $X$
    initialise sequence $X$;
    hierarchically decompose $S$ and $X$;
    recursively search for sequence with MFE structure $S$;
    return $X$;
end RNA-SSD.
```

Fig. 2. Pseudocode for the new RNA secondary structure design algorithm; details are discussed in the text.

quence can result in a completely different MFE secondary structure. Hence, there seems to be little hope for reducing the complexity of evaluating candidate solutions by incremental updating. Consequently, an SLS algorithm for the RNA Secondary Structure Design Problem should keep the number of candidate solution evaluations minimal. In this respect, simple variants of straight-forward SLS algorithms, such as the Min-Conflicts Heuristic [8], can be expected to perform poorly on this problem, particularly when applied to larger problem instances.

Based on these considerations, our algorithm takes a different approach: After constructing an initial candidate sequence for the entire given RNA structure S, that structure and the initial sequence are hierarchically decomposed into smaller components corresponding to substructures of S. At the lowest level, these subproblems are independently solved using a conventional SLS algorithm. Solutions to these subproblems are then combined into candidate solutions for larger subproblems. There is no guarantee that valid solutions to subproblems can be combined into a valid solution of a larger subproblem; hence, at this stage, each combination attempt has to be evaluated using Zuker's algorithm. If at any stage the respective combined candidate sequence does not fold into the required structure, new candidate solutions to the subproblems are determined using the same mechanism as described before.

Following this approach, the expensive evaluation of candidate solution happens primarily at the level of substructures which can be made small enough (by iterated decomposition) to render the $\Theta(n^3)$ complexity of Zuker's algorithm manageable. Larger candidate sequences are only evaluated after merging partial solutions, a process that happens much more rarely. It may be noted that this approach is based on the intuition that although there can be complicated dependencies between subproblems, there is a reasonable chance that solutions to subproblems can be successfully combined into solutions of the entire problem. The empirical performance of our algorithm on biological and artificial RNA structures supports this intuition.

An outline of the algorithm is shown in Figure 2; in the following, we will discuss its components (and parameters) in more detail.

The easiest way of initialising the RNA sequence X would be to randomly and independently assign bases to the positions of X according to a uniform distribution over A,C,G,U. (This is the initialisation method used in RNAinverse.) However, given our goal of minimising the number of candidate solution evaluations, it would be preferable to initialise the sequence in such a way that its MFE structure is as close as possible to the target structure S. To achieve this, we exploit three insights.

Firstly, the assignment should be done in such a way that paired bases in S are assigned complementary bases. This ensures that X can fold into the desired structure S, but there might be alternate conformations with lower free energy. Based on the same intuition, we make sure that the unpaired sequence positions directly following helix regions (*e.g.*, in the loop section of a hairpin loop) are assigned non-complementary bases; this prevents energetically favourable but undesired helix extensions. [2]

Secondly, the fact that C-G base pairings are energetically more favourable than A-U pairings suggests that, by preferentially assigning C-G pairs to helix regions and other paired positions of S, sequences can be obtained whose MFE structures tend to contain these (desired) pairings. Furthermore, by preferentially assigning A and U bases to unpaired sequence positions in X the chances of erroneous pairings should intuitively be decreased. These heuristic choices are closely related to the conditions underlying recent theoretical work on the RNA secondary structure design problem [9]. They are also in agreement with the observation that the average fraction of C-G pairs in helices appears to be relatively high for most types of RNAs.

Finally, we use a tabu mechanism to further minimise the potential for undesired but energetically favourable interactions between subsequences of X. This mechanism is based on assigning short sequences of bases (sequence motifs) to contiguous segments of the target structure S. A sequence motif m is only admissible if it does not form more than d_{min} consecutive base pairs with any previously used motif or with itself. (A similar mechanism is used by Seeman in the design of DNA structures [10].) The maximal segment size l_{max} (*i.e.*, motif length) and the motif distance threshold d_{min} are parameters of our algorithm; obviously, d_{min} and l_{max} need to be set in such a way that sufficiently large sets of admissible motifs exist. In our experiments we somewhat arbitrarily set them as $l_{max} = 10$ and $d_{min} = 5$, but there is likely room for improvement.

Overall, our initialisation algorithm works as follows: the target structure S is par-

[2] Since we are considering non-canonical G-U pairings, it is possible (for bulge loops with one free base) that such an assignment does not exist, in which case this constraint is ignored.

titioned into segments of successive paired or unpaired bases of maximal size l_{max}. Then, the sequence of segments is traversed from the $5'$ to the $3'$ end of the structure. For each segment of unpaired bases we generate a corresponding sequence motif m of the same length by choosing bases A,C,G,U independently and randomly for each position with probabilities p_A, p_C, p_G, p_U with $p_G = p_C$, $p_A = p_U = 1/2 - p_C$, respectively. If base pairing between the first or last base of this segment with the last or first base of a previously assigned unpaired segment could lead to an undesired helix extension (as described above), the probabilistic base generation is conditioned on not producing the problematic bases. If the motif m is admissible in terms of the tabu mechanism described above, it is assigned to the respective segment. Chunks of paired bases are handled analogously, only that in this case probabilities p'_A, p'_C, p'_G, p'_U are used for generating the bases with $p'_G = p'_C$, $p'_A = p'_U = 1/2 - p'_C$ and both the motif m as well as its complement $\bar{m}$ are checked for admissibility and assigned to the two segments corresponding to both sides of the respective helix region.

The probabilities p_C and p'_C are parameters of our algorithm (the other probabilities can be derived from these based on the equalities stated above). For the experiments reported in Sections 4 and 5, we used $p_C = 0.17$ and $p'_C = 0.33$; these values merely reflect the underlying biological intuition and have not been tuned to optimise performance.

Empirical evidence suggests that especially for larger and more complex target structures, the probabilistic base assignment as well as the tabu mechanism contribute significantly towards the strong performance of our algorithm (see Section 6).

Hierarchical Decomposition

The purpose of the hierarchical decomposition procedure is to divide the given target structure S into small structure components; these components correspond directly to subsequences of the candidate sequence X. Similar to Hofacker et al. [6], we split the structure at multiloops; the split points we consider are located at the innermost base pairs of multiloops and stems, which results in $k - 1$ split points for a multiloop with k arms (see Figure 3). Unlike their method, however, we recursively split the structure into two substructures in each decomposition step; thus we obtain a binary decomposition tree whose root is formed by the full target structure X and whose leaves correspond to small substructures of X. Each non-leaf node of the tree represents a substructure of X that can be obtained by merging the two substructures corresponding to its children. More precisely, the hierarchical decomposition is performed as follows: Starting with the entire structure, decomposition steps are performed as long as (1) the structure to be split is not smaller than *MaxSplit* bases, and (2) both resulting substructures are not smaller than *Min-*

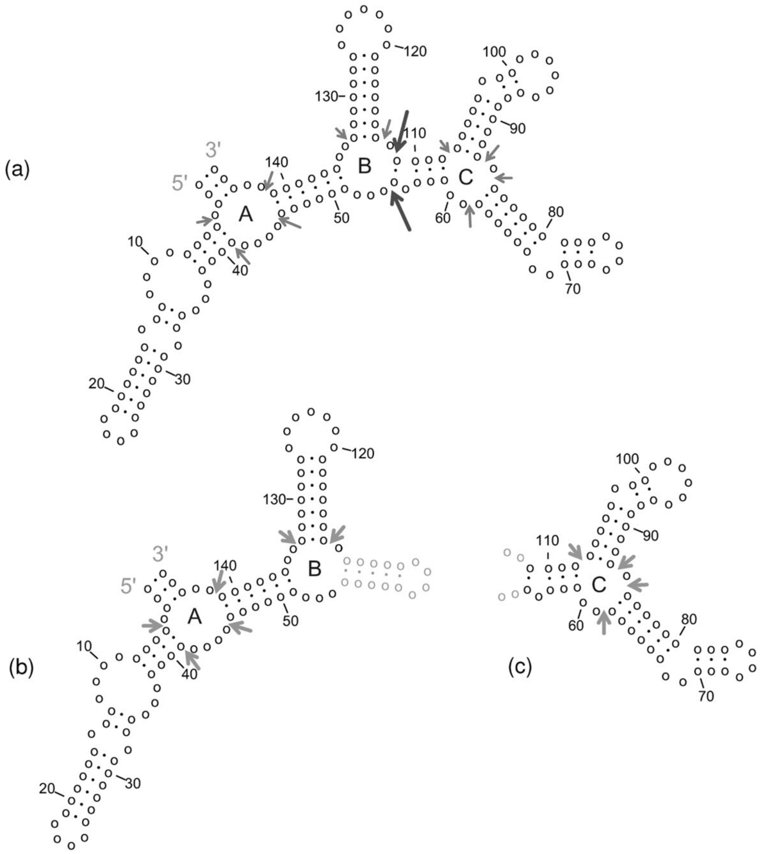

Fig. 3. An example of a structure (a) divided into two substructures (b),(c) using the decomposition method described in the text. The actual split point is indicated by long arrows in (a), alternative split points are indicated by short arrows; bases shown in light grey are added to enforce boundary conditions. The five possible split points for subsequent decomposition steps are indicated by short arrows in (b) and (c).

Split bases. In preliminary experiments, we determined that good performance is obtained by using *MaxSplit*=70 and *MinSplit*=30, and consequently these values were used for all experiments reported in the following.

In contrast to Hofacker et al.'s algorithm [6], our procedure attempts to construct decomposition trees that are as balanced as possible. This is done by selecting a multiloop and split point within that multiloop such that the two substructures obtained by splitting at that point results are as close as possible in size (where the size of a substructure corresponds to the length of the corresponding subsequence).

An example of a split as performed by our hierarchical decomposition method is shown in Figure 3; note that there are only six split points in this structure, five of which would lead to a less balanced decomposition into two substructures (these are shown in Figure 3(b) and (c); using the *MaxSplit* and *MinSplit* settings mentioned above, only three of these are admissible).

In general, such a split creates two new free ends of the underlying RNA sequence (in addition to its original $5'$ and $3'$ ends). Subsequently, we will use Zuker's RNA secondary structure prediction algorithm for evaluating candidate sequences assigned to these substructures; but this algorithm can only be applied to single RNA strands (*i.e.*, structures with two free ends). Therefore, we always split by separating the structure such that the split point never falls within a segment of consecutive free or paired bases, and by connecting the two free ends created by the split, both resulting substructures have exactly two free ends. To create structural boundary conditions at the split points that are similar to those of the original structure, this connection is achieved by merging the free ends of the substructure whose strand has become disconnected as a result of the split with those of a small hairpin loop and stem structure of size 14 (five paired, four unpaired, five paired bases); furthermore, if the other substructure contains a bulge directly after its first base pair, we add two unpaired bases to each of its free ends. The added bases, shown in grey in Figure 3, are generated using the same probabilistic model as in the sequence initialisation procedure described above.

Recursive Stochastic Local Search

The final component of our algorithm is a recursive SLS procedure. Starting at the leaves of the decomposition tree, corresponding to the smallest substructures ${S_k}^*$ of the given target structure S^*, this procedure iteratively modifies single bases of the corresponding subsequences X_k that are in conflict with ${S_k}^*$, *i.e.*, that are paired in the MFE structure S_k of X_k but not in ${S_k}^*$ or vice versa. This is done until either a subsequence X_k with MFE structure ${S_k}^*$ is obtained, or until a maximal number n_L of base modifications have been performed without finding such a sequence. In our experiments, we used $n_L = 5000$; furthermore, as long as no solution is found, every 1000 steps the search is restarted from the initial subsequence. Note that Zuker's RNA structure prediction algorithm has to be called in each iteration of this procedure.

In its recursion step, the SLS procedure is first used to determine valid sequences X_i and X_j for the two substructures corresponding to the children of a given non-leaf node k of the decomposition tree. (If no valid sequence is found for X_i, the algorithm still continues with the previous best candidate sequence for X_i as if it were valid, and similarly for X_j.) A candidate sequence X_k for ${S_k}^*$, the substructure associated with node k, is then determined by merging X_i and X_j (after removing any bases newly introduced in the hierarchical decomposition process). Next, the MFE structure S_k of X_k (determined using Zuker's algorithm) is compared to ${S_k}^*$. If S_k and ${S_k}^*$ are identical, the recursive step has been successfully completed. Otherwise, the conflicting bases in X_i or X_j are memorised and the SLS procedure is recursively called on the subsequence with the higher relative fraction of conflicting bases, resulting in a new sequence X_i or X_j. This process is iterated until either a valid sequence for ${S_k}^*$ has been found, or until a maximal number n_M of attempts to merge subsequences X_i and X_j into a valid sequence for ${S_k}^*$ have been performed. For our experiments, we used $n_M = 5$.

The single-base modifications at the level of smallest substructures are determined based on a randomised first-improvement strategy: With a fixed probability p_{rnd}, an arbitrary base position i in the given subsequence is selected uniformly at random; otherwise, this uniform random choice is restricted to base positions that are currently in conflict with the target structure. For our experiments we used $p_{rnd} = 0.2$. An alternate base assignment is then generated for the selected position using the same probabilistic model for that position as in the previously described sequence initialisation procedure. (In particular, the probabilities for generating particular base assignments depend on whether that base position is paired or free in the target structure.) If necessary, this process of proposing alternate bases for position i is repeated until a base x is found that is not identical with the present, conflicting base assignment to position i. If base assignments for this position previously resulted in a mispairing on higher levels in the recursion, x is also not allowed to be the base assigned on the lowest conflicting level. Replacing the current, conflicting base at position i with x leads to a new candidate sequence X'_k for this substructure. At this stage, the MFE structure S'_k for X'_k is determined (using Zuker's algorithm); if S'_k is closer to ${S_k}^*$ than the MFE substructure S_k of the previous candidate sequence X_k in terms of the number of conflicting bases, the search is continued from X'_k; if the number of conflicting bases is higher for X'_k than for X_k, the search is still continued from X'_k with probability p_{acc}, and from X_k in the remaining cases. For our experiments we used $p_{acc} = 0.2$.

Note that using this first-improvement strategy helps to keep the number of (expensive) candidate solution evaluations per search step small. The randomisation helps to ensure that this search does not get stuck in local optima of the evaluation function.

4 Experimental Results for Artificial Data

The performance of RNA-SSD was empirically evaluated and compared against that of RNAinverse, the Vienna inverse folding algorithm [6], using artificially generated RNA structures as well as biological data. Our results clearly indicate that RNA-SSD shows substantially improved performance over RNAinverse; furthermore, the observed differences in performance increase with problem hardness. In this section we describe the experiments and results for artificial data, while the results for biological data are presented in the next section.

The use of artificial data in the evaluation of RNA-SSD is motivated by several factors. Firstly, this approach allows us to use a large number of similar RNA structures for our empirical analysis, and hence reduces the chance of drawing erroneous conclusions from a small set of atypical results. Secondly, it facilitates the study of extreme cases of RNA structures which may rarely or never occur in nature, but could, for example, have applications in the design of artificial RNA nanostructures; knowledge of an algorithm's behaviour in such extreme cases is often also important for understanding the limits of its applicability. Finally, artificially generated structures with controlled properties provide a means of studying the impact of certain features, such as size of a given structure, or the relative prevalence of bulges, on the performance of RNA design algorithms such as RNA-SSD or RNAinverse.

Artificially generated structures have been previously used by Hofacker et al. [6] for evaluating the performance of the RNAinverse algorithm. However, different from their approach, we decided to implement a simple RNA structure generator which allows us to directly control salient properties of the structures being generated, including the overall size as well as the number and size of bulge, internal, and multiloops, and the length of stems. For these and a number of other structural properties, the generator allows the specification of a random distribution of values (including the special case of only one fixed value); hence, for a given parameter setting, the generator effectively samples from a parameterised random distribution of RNA structures. The generator is described in detail in the Appendix.

Our empirical performance analysis is based on five sets of artificial RNA structures with different structural properties (see Table 1). The parameter settings for the generator were chosen such that different classes of biologically plausible structures were obtained and a rough sense of scaling of the performance of the two algorithms with the size of the input RNA structures could be obtained. (An example structure obtained from our generator and a computationally predicted structure of a biological RNA sequence are shown in Figures 4 and 5.)

All computational experiments were carried out on PCs with dual 1GHz Pentium III processors (only one of which was used by the algorithms we tested), 256KB cache, and 1GB RAM running Red Hat Linux, Version 2.4.9-6smp. Both, RNA-SSD and RNAinverse are highly randomised; consequently, we performed multiple runs of each algorithm on all problem instances and measured the distribution of run-times (RTDs) over all runs on a given instance [11]. In the following, we mainly report mean CPU times from multiple runs on each RNA structure; however, we give more detailed insights into the underlying RTDs in the discussion section. If not explicitly mentioned otherwise, 10 independent runs were performed for each RNA structure. In many cases, one or both algorithms did not solve each given structure in every run performed on it. In cases where the fraction of successful runs, f_s, was higher than 0 but lower than 1, we estimated the expected time required for finding a solution as

$$E_s + (1/f_s - 1) \cdot E_u \tag{1}$$

(a)

Name	# Insts	Size	Bulges	# Multiloops	# Branches per Multiloop	# Internal loops
A_1	100	178–447	No	2–5	3–5	2–5
A_2	100	164–465	Yes	2–5	3–5	2–5
A_3	100	172–409	Yes	1–3	4–6	4–7
A_4	100	203–402	Yes	4–7	2–4	2–4
A_5	20	329–775	Yes	4–10	3–5	4–10

(b)

Stem length	Hairpin size	Internal loop size	Multiloop size	Bulge size	Fr. bulges per stem
				(sets A2-A5 only)	
5–10	3–10	3–10	5–10	1–3	0.1–0.2

Table 1
Characteristics of five sets of artificial RNA structures. (a) For each set, successive columns give the set name, number of instances, size, whether there are bulges or not, and ranges for the number of multiloops, number of branches per multiloop, and number of internal loops. (b) Additional parameters common to all five sets of structures give ranges for the stem length, hairpin size, internal loop size, multiloop size, fraction of bulges per stem, and bulge size (the last two do not apply to set A_1 but do apply to the remaining sets). Exactly how these parameters are used to generate random instances is described in the Appendix.

Name	SR(RNA-SSD)=1 (0)		SR(RNAinv)=1 (0)		t(RNA-SSD) $<$ t(RNAinv)	t(RNA-SSD) $<$ t(RNAinv)/10
A_1	100/100	(0/100)	63/100	(0/100)	100/100	57/100
A_2	85/100	(1/100)	5/100	(13/100)	83/86	73/86
A_3	88/100	(0/100)	3/100	(8/100)	90/92	81/92
A_4	88/100	(0/100)	2/100	(16/100)	81/84	72/84
A_5	19/20	(0/20)	0/20	(5/20)	15/15	14/15

Table 2
Performance results for RNA-SSD and RNAinverse on sets of artificial RNA structures (test-sets $A1$–$A5$). The columns SR($\cdot$)=1 (0) show the fraction of structures for which the respective algorithm found solutions in all (none) of the runs (SR stands for success rate); the columns t(RNA-SSD) $<$ t(RNAinverse) and t(RNA-SSD) $<$ t(RNAinverse)/10 show the fraction of structures which RNA-SSD solves faster and at least 10 times faster, respectively (t stands for run-time).

where E_s and E_u denote the average time for successful and unsuccessful runs, respectively. (Note that $E_s + (1/f_s - 1) \cdot E_u$ is the expected time of a successful run, E_s, plus the expected number of unsuccessful runs prior to the first successful run times the expected time of an unsuccessful run, E_u; unsuccessful runs of both algorithms, RNA-SSD and RNAinverse, can vary substantially w.r.t. their run time.) In the following, for each test-set we indicate the number of RNA structures that were not always or never solved by either algorithm.

Table 2 summarises the performance results for RNA-SSD and RNAinverse on the sets A_1 through A_5 described in Table 1. Each algorithm performed 10 runs

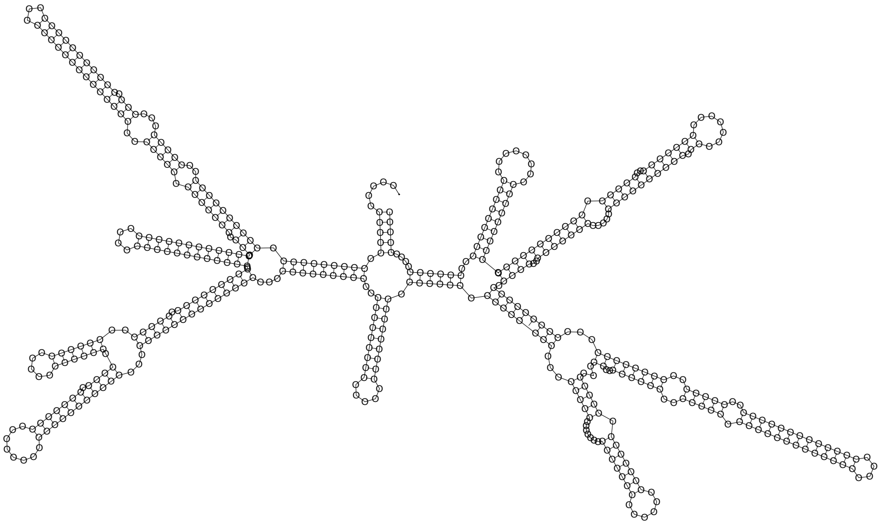

Fig. 4. Example of an artificially created RNA structure from test-set $A5$, length 490 bases.

per structure; unsuccessful runs were terminated after 3600 CPU seconds. Overall, RNA-SSD performed substantially better than RNAinverse. Remarkably, RNA-SSD solved all but one of the instances, while RNAinverse failed to find any solutions for 42 of the given 420 structures within the given time limits. On the remaining instances, RNA-SSD found solutions up to $10\,000$ times faster than RNAinverse, and substantially more consistently throughout multiple runs and across each test-set (see also Figure 6). RNAinverse was found to be slightly faster on only 8 of 420 tested structures. The performance differences in favour of RNA-SSD were found to be particularly pronounced for more complex and biologically plausible structures with bulges. Generally, these structures appear to be more challenging for both RNA secondary structure design algorithms. There is also limited evidence that with increasing size of the structures, the performance advantage of RNA-SSD over RNAinverse becomes more pronounced (further, more substantial evidence to that effect will be presented in the next section).

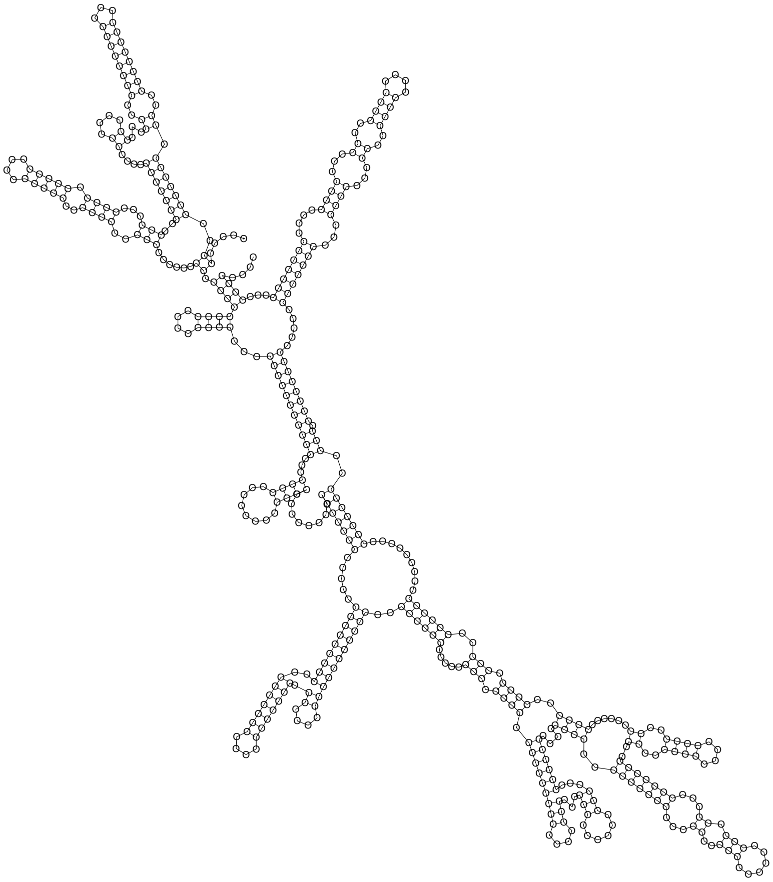

Fig. 5. The predicted structure of the 16S rRNA (small subunit ribosomal RNA) from an unidentified eubacterium, GenBank accession number U81771 (No 11 in Table 3), length 491 bases.

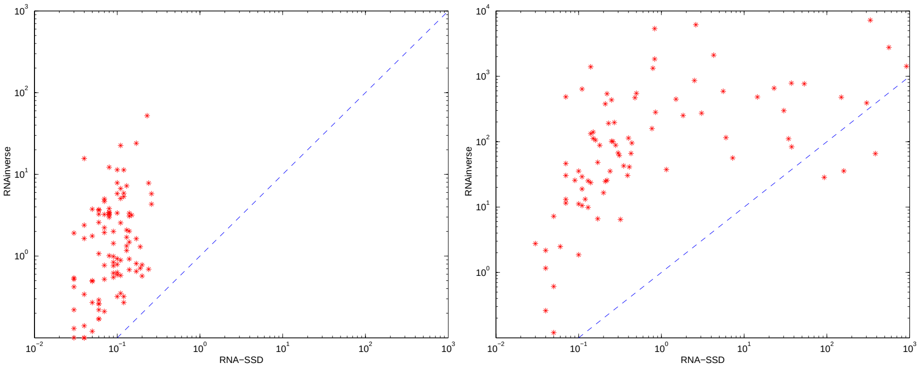

Fig. 6. Performance of RNA-SSD vs. RNAinverse across two test-sets of artificially generated structures. Left: Set $A1$ (structures without bulges), right: Set $A2$ (structures with bulges). Each test-set is comprised of 100 structures, each of which is represented by one data point in the respective correlation plot, with the exception of instances that were never solved by either of the two algorithms. The reported run-times are mean values (in CPU seconds) determined based on 10 runs per structure.

5 Experimental Results for Biological Data

In order to test whether and to which extent the excellent performance results for RNA-SSD would carry over to biological structures, we created a set of ribosomal RNA sequences (rRNAs) obtained from the Ribosomal Database Project[3]. The sequences were chosen in an arbitrary and unbiased way; we note that some are partial sequences, according to their corresponding GenBank entries. For each of these, an MFE structure was computed using the same RNA structure prediction algorithm as the one used within RNA-SSD and RNAinverse for evaluating candidate sequences; this ensures that at least one solution exists for each structure. These structures, although typically not identical to the experimentally determined secondary structures of the respective RNAs, share many important features with these and can hence be considered as biologically realistic test cases for RNA design algorithms such as RNA-SSD or RNAinverse. Our test set is comprised of 24 structures of size 260–1475 bases; we refer to this as test set B (for biological data).

Computational experiments on this test-set were performed analogously to those for the artificial test-sets described above. For each structure, between 10 and 50 independent runs of both algorithms were performed; runs were terminated unsuccessfully if after 3600 CPU seconds on our reference machine no solution had been found. For some input structures, while all runs of RNAinverse on a given input structure were unsuccessful, in some runs, a sequence was found whose MFE structure was close to the desired structure. In these cases we measured the minimal distance between the structures found and the desired structure (where the distance is defined as the number of incorrectly paired or unpaired bases) as well as the time required for finding this best approximate solution.

As can be seen in Table 3, RNA-SSD performed dramatically better than RNAinverse on this test set. While RNA-SSD found solutions for all 24 structures, RNAinverse failed to solve 17 structures. In particular, RNAinverse did not find a correct solution for any structure larger than 500 bases, and for the 7 structures it did solve correctly, its performance in terms of success rate and run-time is substantially inferior to that of RNA-SSD. These results confirm the earlier observation from our experiments with artificial structures that RNA-SSD performs substantially better than RNAinverse, especially on large and biologically realistic structures, which are in many cases beyond the reach of RNAinverse.

It is worth noting that, as can be seen from the table, although larger structures tend to be harder to solve for RNA-SSD, the correlation between size and hardness (*i.e.*, run-time) is not very strong. In particular, there are several cases of small and medium size structures that appear to be atypically hard to design. Based on the

[3] `http://rdp.cme.msu.edu/download/SSU_rRNA/unaligned/SSU_Unal.gb`

	source name	size	RNA-SSD		RNAinverse		
no	organism (GenBank accession no.)	(bases)	succ rate	exp time	succ rate		exp time
1	Rhizobiaceae group bacterium NR64 (Z83250)	260	50/50	4.15	2/50		4639.38
2	Bacterial sp. from marine plankton (L11935)	264	47/50	2.91	6/50		1195.98
3	Leptospira interrogans strain 94-7997013 (LIU92530)	289	6/50	2517.57	(2/50)	(2)	(15129.52)
4	Unidentified marine eubacterium (U84629)	299	8/50	189.12	(2/50)	(2)	(15993.48)
5	Anabaena uncultured bacterium SY2-21 (AF107506)	337	39/50	23.96	3/50		13330.04
6	Ochrobactrum sp. BL200-8 (AF106618)	350	47/50	10.19	6/50		5446.07
7	uncultured eubacterium 3-25 (AJ011149)	376	41/50	70.84	(2/50)	(2)	(24975.68)
8	Prevotella ruminicola M384 (S70838)	389	49/50	16.23	1/50		46591.08
9	Unidentified crenarchaeote (U63350)	418	21/25	7.44	1/25		25117.46
10	Uncultured eubacterium clone CRE-FL72 (AF141485)	473	4/25	441.59	(3/25)	(6)	(21106.95)
11	Unidentified eubacterium clone vadinIA59 (U81771)	491	25/25	81.43	1/25		46118.47
12	Stenotrophomonas isolate P-26-14 (AJ130779)	506	25/25	18.66	(2/25)	(2)	(14728.98)
13	Nitrobacter sp. Nb4 (AF096836)	646	21/25	65.85	(4/25)	(4)	(17403.46)
14	Wolbachia pipientis (X61771)	659	18/25	342.45	(2/25)	(8)	(41877.73)
15	Uncultured archaeon ST1-4 (AJ236455)	751	23/25	1483.60	0/25	(-)	(-)
16	Bradyrhizobium isolate 283A (AJ132572)	780	19/25	92.36	0/25	(-)	(-)
17	Spirochaeta sp. clone Hs33 (AB015827)	856	10/10	95.54	0/10	(-)	(-)
18	Sulfolobus acidocaldarius (D38777)	858	3/10	376.31	0/10	(-)	(-)
19	Pseudomonas sp. Y1000 (X99676)	950	10/10	168.45	0/10	(-)	(-)
20	Unidentified methanogen ARC21 (AF029195)	1053	10/10	37.06	0/10	(-)	(-)
21	Planctomyces brasiliensis DSM 5305 (X81949)	1150	10/10	382.79	0/10	(-)	(-)
22	Uncultured archaeon KTK 9A (AJ133622)	1296	10/10	152.89	0/10	(-)	(-)
23	Methanococcus fervens (AF056938)	1398	6/10	313.40	0/10	(-)	(-)
24	Methanococcus jannaschii (L77117 bases 157084-159459)	1475	10/10	151.76	0/10	(-)	(-)

Table 3

Performance results for RNA-SSD and RNAinverse on computationally predicted structures for a set of rRNA sequences obtained from the RDP database (test-set B). For each of RNA-SSD and RNAinverse, for structures where correct solutions were found, the *succ rate* column gives, for each structure, the number of runs performed in which a correct solution was found, divided by the total number of runs performed, and the *exp time* column gives the expected time for finding the correct solution, estimated by formula (1). For structures where only approximate solutions were found by RNAinverse, the *succ rate* column gives the fraction of runs in which the best approximate (rather than correct) solution was found and the distance to the desired structure is given in parentheses, and the *exp time* column gives the expected time for finding this approximate solution, also in parentheses. in 2 of 50 runs, Dashes (–) indicate cases where RNAinverse did not return any solution within the cutoff time.

no	size	RNA-SSD						RNAinverse	
		SeqInit-a		SeqInit-b		SeqInit			
		succ rate	exp time	succ rate	exp time	succ rate	exp time	succ rate	exp time
2	264	46/50	5.99	47/50	5.42	47/50	2.91	6/50	1195.98
5	337	17/50	133.28	15/50	153.76	39/50	23.96	3/50	13330.04
8	389	48/50	35.41	47/50	28.73	49/50	16.23	1/50	46591.08
12	506	23/25	56.57	24/25	55.90	25/25	18.66	2/25 (2)	(14728.98)
13	646	13/25	255.70	12/25	271.48	21/25	65.85	4/25 (4)	(17403.46)

Table 4
Comparison of the performance of RNA-SSD with different type of sequence initialization procedures. The leftmost column gives the numbers of the sequences used, consistent with Table 3. *SeqInit* is the default procedure of RNA-SSD. *SeqInit-a* uses probabilistically biased base assignment but does not use the tabu mechanism. *SeqInit-b* uses unbiased base assignment but does use the tabu mechanism. For each algorithm variant, the *succ rate* and *exp time* column entries are defined as in Table 3.

common characteristics of these structures, we believe that the difficulty is caused by short and unstable stems or other small substructures attached to multi-loops. In general, such substructures would severly restrict the number of sequences that can fold into the desired structure, *i.e.*, reduce solution density, which is known to correlate with the hardness of the RNA design problem and other combinatorial problems for local search search algorithms [12–14]. Differences in solution density are also very likely responsible for the extreme variations in hardness of artificial structures observed in Section 4.

6 Discussion

While the results presented in the previous two sections clearly illustrate the substantial performance advantage of RNA-SSD over the earlier RNAinverse algorithm, a number of questions about the characteristics of the algorithm and the solutions to the design problems provided by it remain to be answered.

One such question concerns the reasons for the performance differences between the two algorithms, and particularly, the role of the sequence initialisation procedure. Recall that our method of sequence initialisation introduces a probabilistic bias for more stable GC-rich paired (stem) regions and GC-poor unpaired (loop) regions. Furthermore, it uses a tabu mechanism for preventing incorrect pairing between short sequence motifs. To find out to which extent the performance of RNA-SSD depends on these two initialisation mechanisms, we performed computational experiments on selected structures from our biological test-set B. In addition to RNA-SSD with the standard sequence initialisation procedure, which uses both mechanisms (here referred to as *SeqInit*), we studied two variants of RNA-SSD that use *SeqInit-a*, a modified initialisation routine that uses the probabilistically biased base assignment but not the tabu mechanism, and *SeqInit-b*, a modified ini-

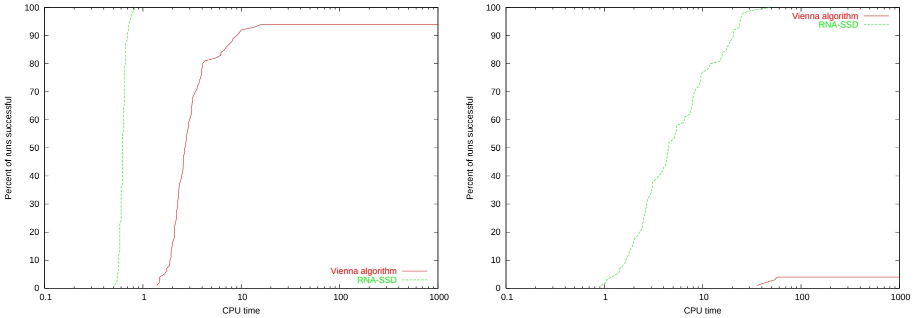

Fig. 7. Empirical run-time distributions (RTDs) for RNA-SSD and RNAinverse for a small randomly generated structure from test-set A_1 without bulges (left, length 119) and a small rRNA fragment (right, length 118). The RTDs are based on 100 runs of each algorithm in all of which a solution was found. CPU time is measured in seconds.

tialisation routine that does not use probabilistically biased base assignment, but does use the tabu mechanism. The performance of these variants was measured as described in Section 5, and compared to RNA-SSD with the standard initialisation and RNAinverse, which initialises the sequence completely at random, choosing each of the bases A,C,G,U with uniform probability for each sequence position.

As can be seen from Table 4, both initialisation mechanisms, biased base assignment and the tabu mechanism, contribute to RNA-SSD's excellent performance. At the same time, even with only one of these mechanisms, RNA-SSD performs substantially better than RNAinverse. In particular, this is the case for the variant that does not use probabilistically biased base assignment. Thus, simple variants of RNA-SSD are preferable even in applications where probabilistic biases in base composition need to be avoided.

Another interesting difference between RNA-SSD and RNAinverse is the use of artificial capping structures for sealing the open ends that arise when splitting a given RNA structure. Our preliminary experiments showed that using no capping at all (and just connecting the two bases at the split point to seal the open end) often leads to undesired secondary structures when merging the partial sequences for two substructures. This was mainly due to the fact that the free bases that were concatenated after splitting formed pairs at merging. Thus, the insertion of an artificial branch, short and easy to design, intuitively made sense. For this purpose, we chose the structure described in Section 3, *i.e.* five paired, four unpaired, five paired bases). However, in our empirical performance we have seen some evidence suggesting that using this capping structure can sometimes provide more stabilisation to a multiloop than the original stem, and thus be detrimental to the performance of RNA-SSD, as in such a case what appears to be a stable substructure may destabilise after a merging step. Thus, we believe that by using better capping structures, especially ones that are chosen dynamically dependent on the stem they replace, the performance of RNA-SSD may be further improved.

Both RNA-SSD and RNAinverse are highly stochastic algorithms: When applied to the same structure multiple times, the time for finding a solution may vary substantially. (Note, however, that by using the same random seed, any run of RNA-SSD can be perfectly reproduced.) This has been taken into account in our performance evaluation by generally performing sufficiently many runs on each problem instance that reasonably stable statistics (such as mean run-times) were obtained. A more detailed analysis of the full empirical run-time distributions of both algorithms on particular problem instances indicates that the time required for reaching any given success probability is substantially shorter for RNA-SSD than for RNAinverse for both, artificial and biological RNA structures (see Figure 7). The high degree of variability in the time required for finding a solution is typical of many SLS algorithms for other combinatorial problems [14].

We now turn our attention from RNA-SSD's performance to the quality of the results it produces. One important aspect in this context is the stability of the designed structures. Note that RNA-SSD does not explicitly evaluate or optimise the thermodynamic stability of the desired secondary structure that is achieved by the designed sequence. Thus, *a priori* it is not clear whether the sequences designed by RNA-SSD will be reasonably stable under current models of RNA secondary structure prediction.

To investigate this issue, we selected three of the biological sequences underlying our test-set B and compared the stability of their MFE fold against that of the sequences we designed for the same (predicted) structures. (The structures were selected by choosing a small, medium, and long sequence from set B in an otherwised unbiased way.) In a first experiment, we used the partition function option of Vienna folder to predict for each of a number of sequences we designed for the three selected test structures, as well as for the respective biological sequence the following measures: (i) the probability, Prob[struc$|$seq], of the structure in the ensemble of all possible structures for that sequence, (ii) the minimum and median probability of correct base-pairings over the given sequence, and (iii) the maximum and median probability of incorrect base-pairings over the given sequence. We sorted the sequences for each structure according to the first of these measures, and report the stability results for the best and median sequences from the respective distribution of solutions along with those for the biological sequence.

As can be seen in Table 5, the sequences obtained by RNA-SSD are substantially more stable than the biological sequences according to these measures of stability. This is particularly striking when comparing the respective probabilities of the structures; the effect seems to increase with the size of the given structure. The differences between the best and median results obtained by RNA-SSD highlight the fact that by running RNA-SSD multiple times on the same structure, a set of substantially different solutions can be obtained. This property of the algorithm can be very useful in applications where more than one solution to a given design problem is needed, *e.g.*, in the context of screening a set of solutions for sequences with certain properties, such as stability of the respective structure.

				Prob[correct bps]		Prob[incorrect bps]		GC content
no	sequence	size	Prob[struc|seq]	minimum	median	maximum	median	paired / free
1	maximises Prob[struc|seq]	260	0.00372	0.38644	0.98775	0.00004	0.00002	0.67 / 0.44
1	25-th best seq Prob[struc|seq]	260	0.00044	0.17285	0.97163	0.00004	0.00002	0.70 / 0.44
1	biological seq.	260	0.00023	0.31180	0.86195	0.47472	0.00015	0.66 / 0.37
12	maximises Prob[struc|seq]	506	$2.07 \cdot 10^{-6}$	0.11606	0.96491	0.00009	0.00003	0.74 / 0.46
12	13-th best seq Prob[struc|seq]	506	$4.97 \cdot 10^{-8}$	0.06745	0.90128	0.00079	0.00006	0.65 / 0.48
12	biological seq.	506	$4.03 \cdot 10^{-8}$	0.03700	0.73800	0.69722	0.00010	0.62 / 0.42
17	maximises Prob[struc|seq]	856	$2.54 \cdot 10^{-10}$	0.11985	0.88559	0.01065	0.00011	0.68 / 0.44
17	median over 10 Prob[struc|seq]	856	$1.46 \cdot 10^{-12}$	0.00958	0.83251	0.10798	0.00011	0.74 / 0.43
17	biological seq.	856	$1.00 \cdot 10^{-16}$	0.06099	0.89210	0.77055	0.00011	0.63 / 0.41

Table 5
Stability measurements for sequences that fold to structures 1, 12, and 17 from test-set B (see Table 3). For a given sequence and structure, *Prob[struc|seq]* refers to the probability of the given structure for the given sequence in the ensemble of all possible pseudoknot-free structures for that sequence. For each structure, column *sequence* gives three sequences that fold to that structure, namely the sequence among the runs for Table 3 for which Prob[struc|seq] is maximised, the sequence among the runs for Table 3 whose Prob[struc|seq] value is closest to the median over all solutions found by RNA-SSD for that structure, and the biological sequence. The fifth and sixth columns give the minimum and the median base pair probabilities, taken over all of the (correct) base pairs in the respective structure. The seventh and eighth columns give the maximum and the median base pair probabilities, taken over all of the (incorrect) base pairs that are not contained in the target structure. The final column lists the fraction of bases G and C in the paired and unpaired regions of the respective RNA. The measurements were obtained using the partition function option of Vienna package.

It is interesting to note that, as a result of the probabilistic bias in RNA-SSD's sequence initialisation and recursive stochastic local procedure, the GC content of the designed sequences in both, paired and unpaired regions is similar to that of the original biological sequences (see Table 5). The fact that the GC content in paired regions of the designed sequences tends to be slightly higher than in the biological sequence provides a partial explanation for the observed differences in stability; however, given the magnitude of these differences, we strongly suspect that the main cause lies in other factors, particularly in the exact composition of stacked pairs of bases.

In a second experiment, we used Zuker's mfold algorithm [15] to evaluate the stability of the MFE structures of the same sequences. More specifically, for each sequence we determined the number of suboptimal structures found by mfold, the predicted energy of the MFE structure, and the energy gap between the MFE structure and the best suboptimal structure (using mfold's default parameters). As can

no	sequence	size	Prob[struc|seq]	no of	MFE	MFE-next gap
				structures	(kcal/mol)	(kcal/mol)
1	maximises Prob[struc|seq]	260	0.00372	3	-105.50	2.5
1	10-th best seq Prob[struc|seq]	260	0.00169	3	-99.9	2.1
1	25-th best seq Prob[struc|seq]	260	0.00044	4	-109.3	3
1	biological sequence	260	0.00023	7	-102.3	1.4
12	maximises Prob[struc|seq]	506	$2.07 \cdot 10^{-6}$	6	-219.6	0.7
12	5-th best seq Prob[struc|seq]	506	$3.78 \cdot 10^{-7}$	6	-182.4	3.6
12	12-th best seq Prob[struc|seq]	506	$5.05 \cdot 10^{-8}$	6	-199.7	0.7
12	biological sequence	506	$4.03 \cdot 10^{-8}$	23	-184.7	1.8
17	maximises Prob[struc|seq]	856	$2.54 \cdot 10^{-10}$	6	-350.6	2.3
17	2nd best seq Prob[struc|seq]	856	$2.84 \cdot 10^{-12}$	2	-367.9	2.1
17	5-th best seq Prob[struc|seq]	856	$1.46 \cdot 10^{-12}$	9	-387.1	4.5
17	biological sequence	856	$1.00 \cdot 10^{-16}$	13	-336.2	2.2

Table 6
Additional stability measurements for sequences that fold to structures 1, 12, and 17 from test-set B (see Table 3). For each structure, column *sequence* gives four sequences that fold to that structure. For structure 1, these are the sequence among the runs for Table 3 for which Prob[struc|seq] is maximised, the sequence among the runs for Table 3 with the 10th and 25th highest Prob[struc|seq] values, and the biological sequence. For structure 12, these are the sequence among the runs for Table 3 for which Prob[struc|seq] is maximised, the sequence among the runs for Table 3 with the 5th and 12th highest Prob[struc|seq] values, and the biological sequence. For structure 17, these are the sequence among the runs for Table 3 for which Prob[struc|seq] is maximised, the sequence among the runs for Table 3 with the 2nd and 5th highest Prob[struc|seq] values, and the biological sequence. The fourth column shows for each structure and sequence, the probability of the structure for the sequence in the ensemble of all possible pseudoknot-free structures for that sequence. The fifth column shows for each sequence the number of structures, including the MFE structure and all suboptimal structures, as predicted by mfold on that sequence. The sixth column shows the MFE in kcal/mol, and the seventh column shows the gap between the MFE and the lowest energy of a suboptimal structure. The measurements were obtained using Zuker's mfold algorithm.

be seen from the results of this experiment, shown in Table 6, RNA-SSD can design sequences that have significantly lower (predicted) free energy than the biological sequences, and significantly fewer suboptimal structures within the same free energy range. Furthermore, the MFE gaps between the two structures with lowest free energy tends to be similar between the biological sequences and those designed by RNA-SSD. Further analysis shows that for the designed sequences the differences between the optimal and the lowest energy suboptimal structures tend to be smaller than for the biological sequences (10-15% *vs* 16-33% of the positions are paired differently).

Overall, both computational experiments indicate that the sequences obtained from RNA-SSD are predicted to be significantly more stable than the biological sequences, even though RNA-SSD does not explicitly optimise its solutions for stability of the respective secondary structure. Of course, this result is not as surprising as it may seem at the first glance when considering that the function of biological RNAs typically does not only depend on the secondary structure of the molecule. Given such additional constraints, *e.g.*, on the bases occurring at certain positions or on GC content, the space of solutions to the respective more complex RNA design problems can be expected to be substantially smaller, and may no longer contain solutions of the stability that can be quite easily achieved when no additional constraints are considered.

Finally, we address the question to which extent RNA-SSD is able to design the true secondary structures of biological RNAs, as opposed to predicted structures of biological sequences, as used in our test-set B in this study. Since RNA-SSD, like RNAinverse, uses Zuker's secondary structure prediction algorithm, as implemented in the Vienna package's 'fold' function, for evaluating its solution candidates and for guiding its search process, it inherits some of the fundamental limitations of that algorithm. In particular, Zuker's algorithm is limited to pseudoknot-free structures, and it is also known to be somewhat imprecise for large RNAs [15]. Notice, however, that in RNA-SSD, Zuker's algorithm is used as a subroutine that can in principle be replaced by *any* secondary structure prediction algorithm (such as the algorithms by Rivas and Eddy [16] or Gultaev et al. [17]).

Clearly, the choice of this subroutine has a significant impact on the performance of RNA-SSD as well as on the quality of its results; however, the overall search process is not in any way geared towards or tuned for the particular secondary structure prediction algorithm used in this study. (Note that while the structure decomposition relies on the pseudoknot-freeness, it can still work for pseudoknots with minor modifications.) Our use of the Vienna fold function was mainly motivated by the fact that the underlying algorithm and energy model are close to the state-of-the-art in energy-based secondary structure prediction, and that the code is publicly available. Running some of the RNA sequences designed by RNA-SSD through mfold, we found that in most cases, the desired structure is predicted; only in some cases, small differences between the predictions of Vienna fold function and mfold are observed, which reflect the fact that mfold uses a slightly different energy model.

A priori, it is not clear whether RNA-SSD has a realistic chance to solve design problems for real RNA secondary structures; in particular, it may be expected that due to limitations of Zuker's algorithm, there is no sequence for such real structures that would be predicted to fold correctly by either mfold or the Vienna fold function. (Think, for example, of RNA structures that are substantially stabilised by

		size	RNA-SSD		RNAinverse	
no	description (source)	(bases)	succ rate	exp time	succ rate	exp time
1	Minimal catalytic domains of the hairpin ribozyme satelite RNA of the Tobacco ringspot virus ([18] Fig. 1a)	65	100/100	0.35	57/100	1.50
2	U3 snoRNA 5' domain from Chlamydomonas reinhardtii, in vivo probing ([19] Fig. 6B)	79	100/100	0.05	100/100	0.12
3	H.marismortui 5S rRNA ([20])	122	(100/100) (2)	(168.88)	(29/100) (2)	(44.51)
4	VS Ribozyme from Neurospora mitochondria ([21] Fig. 1A)	167	80/100	1.50	49/100	58.53
5	R180 ribozyme ([22] Fig. 2)	178	6/100	13429.36	(1/100) (5)	(16502.23)
*6	XS1 Ribozyme, Bacillus subtilis P RNA based ribozyme ([23] Fig. 2A)	314	13/100	903.29	1/100	94986.38
*7	Homo Sapiens RiboNuclease P RNA ([24], Fig. 4)	342	4/100	1427.57	1/100	97809.30
8	S20 mRNA from E. coli ([25] Fig. 2)	372	40/100	1222.75	(1/100) (8)	(98435.55)
9	Halobacterium cutirubrum RNAse P RNA ([26] Fig. 2F)	375	(1/100) (6)	(82439.65)	0/100 (–)	–
10	Group II intron ribozyme D135 from Saccharomyces cerevisiae mitochondria ([27] Fig. 5)	583	99/100	26.85	0/100 (–)	–

Table 7
Performance results for RNA-SSD and RNAinverse on structures from the biological literature (test-set C). Structures marked with an asterisk ($*$) were obtained from original, pseudoknotted structures by disregarding 8bp in each case in order to remove the pseudoknot. In all other cases, the original structures were pseudoknot-free. For each of RNA-SSD and RNAinverse, for structures where correct solutions were found, the *succ rate* column gives, for each structure, the number of runs performed in which a correct solution was found, divided by the total number of runs performed, and the *exp time* column gives the expected time for finding the correct solution, estimated by formula (1). For structures where only approximate solutions were found by RNAinverse, the *succ rate* column gives the fraction of runs in which the best approximate (rather than correct) solution was found as well as the distance to the desired structure, and the *exp time* column gives the expected time for finding this approximate solution. in 2 of 50 runs, Dashes (–) indicate cases where RNAinverse did not return any solution within the cutoff time of 1000 CPU seconds.

tertiary interactions or pseudoknots.) Therefore, we performed a final experiment to see how close RNA-SSD solutions can get to such real RNA structures. For this experiment, ten structures comprised of between 60 and 600 bases, consistent with experimental evidence and empirical data, were selected from the biological literature. Of these, only two contained pseudoknots, which we removed by disregarding eight base pairs in each case. For each of the 10 structures in our test-set C, we performed 100 independent runs with RNA-SSD and RNAinverse, respectively, using the same experimental protocol as described above, except that unsuccessful runs were terminated after a cutoff time of 1000 CPU seconds.

As can be seen from the results reported in Table 7, RNA-SSD performs well and substantially better than RNAinverse in all cases, particularly for the larger structures with more than 300 bases. It should be noted that none of the ten biological sequences is predicted to fold into the correct structure by the Vienna fold function used in RNA-SSD and RNAinverse. We hypothesise that the reason why RNA-SSD finds sequences that are predicted to fold correctly is because RNA-SSD is able to design sequences that fold much more stably into a desired structure than does the biological sequence for that structure. Zuker's algorithm is based on a number of simplifying assumptions (such as pseudoknot-freeness) and uses an energy model (including numerous, empirically determined parameters) that provides only an approximation of the real thermodynamic energy (for example, because it ignores the role of RNA tertiary structure in secondary structure formation and stabilisation). However, this model does capture the major energy contributions and is likely to provide a reasonable approximation of the true free energy of a given sequence and structure. For the biological sequences, this approximation clearly is not accurate enough to obtain the correct prediction. RNA-SSD, however, can find much more stable solutions, for which the inaccuracies of the energy model used for structure prediction are no longer as critical. As a consequence of this hypothesis, the limitations of current energy models for RNA structure are less problematic for structure design than for prediction; however, this issue clearly needs further investigation.

In a similar experiment with three larger RNA structures (1500–2000 bases) that were obtained from comparative sequence analysis and are believed to be very close to the true biological structures, we found that RNA-SSD, while unable to precisely design the target structure, constructed sequences whose predicted MFE structure is substantially closer to the target than that of the respective biological sequences. (RNAinverse failed to return any result within 200 hours of CPU time on our reference machines.) These results confirm the explanation given above and additionally suggest that the effect of the previously mentioned inaccurracies in the energy model underlying Zuker's algorithm are cumulative in the length of the given sequence and the complexity of the respective secondary structure. This latter assumption is consistent with the observation that for large biological RNAs with complex structures, Zuker's algorithm and related energy-based RNA secondary structure prediction methods perform rather poorly. However, as previously discussed, RNA-SSD (as well as RNAinverse) uses Zuker's algorithm merely as a subroutine, and we expect that by replacing this subroutine with an implementation of a substantially better structure prediction algorithm (if and when one becomes available) would lead to even better performance of RNA-SSD.

7 Conclusions and Future Work

We have introduced a new algorithm for the RNA secondary structure design problem, an interesting and complex constraint satisfaction problem from computational biology. Our algorithm is based on a stochastic local search technique that uses a carefully designed probabilistic initialisation procedure and a hierarchical structure decomposition technique to keep the high cost of evaluating candidate solutions manageable. The empirical performance of our new algorithm on a broad range of biological and artificially generated RNA structures is substantially better than the state-of-the-art method for this problem, the RNAinverse algorithm from the Vienna RNA Package, both in terms of speed as well as with respect to the structures that can be solved. We attribute this success to a powerful combination of modern constraint-solving methods (particularly, advanced SLS techniques) and problem-specific biological knowledge, as for example embodied in the heuristic initialisation method. The RNA-SSD software is publicly available under the name of RNA Designer at www.rnasoft.ca [1].

This work can be extended in various directions. Firstly, we are convinced that further substantial improvement of our algorithm can be achieved. By using a customised implementation of Zuker's RNA secondary structure prediction algorithm, it should be possible to significantly speed up the evaluation of candidate sequences. This modified evaluation procedure would not apply the full dynamic programming algorithm underlying Zuker's method for each evaluation, but start from a previous dynamic programming matrix and update only those elements that have been affected by one or more local sequence changes. Another area for further improvements is the SLS algorithm used for the smallest substructure; we expect that more advanced SLS methods for constraint satisfaction problems, such as tabu-search techniques (see, *e.g.*, [28]), will lead to improved performance of the overall algorithm. Also, we are currently studying recursive SLS procedures that incorporate a dynamic stochastic decomposition into even smaller substructures (which are conceptually related to the segments currently used during sequence initialisation); preliminary results suggest that this extension leads to substantial performance improvements, especially for large RNA structures with more than 1000 bases.

Also, further empirical analysis of our algorithm as well as RNAinverse could help to better understand the strengths and limitations of both approaches. We plan to study both algorithms' behaviour on a wider range of biologically plausible randomly generated sequences with controlled properties to determine, for example, which factors make instances of the RNA secondary structure design problem hard or easy to solve. This type of investigation is facilitated by our random RNA structure generator, whose underlying probabilistic model can be easily extended to yield structures based on important structural and statistical properties (such as helix length, number and location of bulges, *etc.*) of various classes of biological RNAs.

Finally, the functionality of our algorithm and its software implementation can be extended in several useful ways, and indeed this will be necessary in order to apply RNA-SSD to many biological applications. We welcome input from the biological and biomolecular computation communities on which additional functions would be particularly useful. We have already extended the algorithm to support additional sequence constraints that limit certain positions to contain certain bases and to bias the GC content of the designed sequence [1]. Sequence constraints are needed, for example, in the designs of nanostructures with “sticky ends” [29,4]. Other extensions that we are considering are design of pseudoknotted secondary structures, RNA strands that form desired joint secondary structure with one or more other RNA or DNA strands, or both. Such extensions could be useful in design of DNA and RNA enzymes, DNA and RNA nanostructures [29,4] or molecular motors [30]. Since adding such functionality to RNA-SSD will likely increase the time needed for secondary structure design, we conjecture that the algorithmic advances reported here will be very important in obtaining efficient algorithms for extensions to the basic secondary design problem.

References

[1] M. Andronescu, R. Aguirre-Hernández, A. Condon, H. H. Hoos, RNAsoft: a suite of RNA secondary structure prediction and design software tools, Nucl. Acids. Res. 31 (2003) 3416–3422.

[2] N. Usman, J. McSwiggen, Catalytic RNA (ribozymes) as drugs, Annual Reports in Medicinal Chemistry 30 (1995) 285–294.

[3] T. Cech, Ribozyme engineering, Current Opinions in Structural Biology 2 (1992) 605–609.

[4] E. Winfree, F. Liu, L. Wenzler, N. Seeman, Design and self-assembly of 2D DNA crystals, Nature 394 (1998) 539–544.

[5] R. Lyngsø, M. Zuker, C. Pedersen, Fast evaluation of internal loops in RNA secondary structure prediction, Bioinformatics 15 (1999) 440–445.

[6] I. Hofacker, W. Fontana, P. Stadler, L. Bonhoeffer, M. Tacker, P. Schuster, Fast folding and comparison of RNA secondary structures, Chemical Monthly 125 (1994) 167–188.

[7] D. H. Mathews, J. Sabina, M. Zuker, D. H. Turner, Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure, J. Mol. Biol. 288 (1999) 911–940.

[8] A. Zuker, B. Mathews, C. Turner, Algorithms and thermodynamics for RNA secondary structure prediction: A practical guide, in: J. Barszewski, B. Clark (Eds.), RNA Biochem & Biotech, Kluwer Academic Publishers, 1999.

[9] S. Minton, M. Johnston, A. Philips, P. Laird, Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems, Artificial Intelligence 52 (1992) 161–205.

[10] C. Heitsch, A. Condon, H. H. Hoos, From RNA secondary structure to coding theory: A combinatorial approach, in: Proceedings of the Eighth International Workshop on DNA Based Computers, LNCS 2568, Springer Verlag, 2003, pp. 215–228.

[11] N. Seeman, De novo design of sequences for nucleic acid structural engineering, Journal of Biomolecular Structure and Dynamics 8 (3) (1990) 573–581.

[12] H. Hoos, T. Stützle, Evaluating Las Vegas algorithms – pitfalls and remedies, in: Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufman, 1998, pp. 238–245.

[13] P. Schuster, W. Fontana, P. Stadler, I. Hofacker, From sequences to shapes and back: A case study in RNA secondary structures, in: Proceedings of the Royal Society London B 255, 1994, pp. 279–284.

[14] D. Clark, J. Frank, I. Gent, E. MacIntyre, N. Tomov, T. Walsh, Local search and the number of solutions, in: Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming, LNCS 1118, Springer Verlag, 1996, pp. 119–133.

[15] H. Hoos, Stochastic local search – methods, models, applications, Ph.D. thesis, Darmstadt University of Technology, Department of Computer Science (1998).

[16] E. Rivas, S. Eddy, A dynamic programming algorithm for RNA structure prediction including pseduoknots, J. Mol. Biol. 285 (1999) 2053–2068.

[17] A. Gultaev, F. van Batenberg, C. Pleij, The computer simulation of RNA folding pathways using a genetic algorithm, J. Mol. Biol. 250 (1995) 37–51.

[18] M. J. Fedor, Structure and function of the hairpin ribozyme, J. Mol. Biol. 297 (2) (2000) 269–291.

[19] M. Antal, A. Mougin, M. Kis, E. Boros, G. Steger, G. Jakab, F. Solymosy, C. Branlant, Molecular characterization at the RNA and gene levels of U3 snoRNA from a unicellular green alga, chlamydomonas reinhardtii, Nucl. Acids. Res. 28 (2000) 2959–2968.

[20] M. Szymanski, M. Barciszewska, V. Erdmann, J. Barciszewski, 5S ribosomal RNA database, Nucl. Acids. Res. 30 (2002) 176–178.

[21] D. Lafontaine, D. Norman, D. Lilley, Structure, folding, and activity of the vs ribozyme: importance of the 2-3-6 helical junction, The EMBO Journal 20 (6) (2001) 1415–1424.

[22] Z. C. L. Sun, R. Gottlieb, B. Zhang, A selected ribozyme catalyzing diverse dipeptide synthesis, Chemistry & Biology 9 (2002) 619–628.

[23] E. Mobley, T. Pan, Design and isolation of ribozyme-substrate pairs using RNase P-based ribozymes containing altered substrate binding sites, Nucl. Acids. Res. 27 (21) (1999) 4298–4304.

[24] C. Pitulle, M. Garcia-Paris, K. R. Zamudio, N. R. Pace, Comparative structure analysis of vertebrate ribonuclease P RNA, Nucl. Acids. Res. 26 (14) (1998) 3333–3339.

[25] G. Mackie, Secondary structure of the mRNA for ribosomal protein S20. Implications for cleavage by ribonuclease E, J. Biol. Chem. 267 (1992) 1054–1061.

[26] E. Haas, D. Armbruster, B. Vucson, C. Daniels, J. Brown, Comparative analysis of ribonuclease P RNA structure in Archaea, Nucl. Acids. Res. 24 (7) (1996) 1252–1259.

[27] J. Swisher, C. Duartel, L. Sul, A. Pyle, Visualizing the solvent-inaccessible core of a group ii intron ribozyme, The EMBO Journal 20 (8) (2001) 2051–2061.

[28] O. Steinmann, T. Stützle, A. Strohmaier, Tabu search vs. random walk, in: KI:Advances in Artificial Intelligence, LNCS 1303, Springer Verlag, 1997.

[29] C. Mao, T. H. LaBean, J. H. Reif, N. C. Seeman, Logical computation using algorithmic self-assembly of DNA triple-crossover molecules, Nature 407 (2000) 493–496.

[30] B. Yurke, A. Turberfield, F. S. A.P. Mills Jr, J. Newmann, A DNA-fuelled molecular machine made of DNA, Nature 406 (2000) 605–608.

Appendix:
A Generator for Artificial RNA Structures with Controlled Properties

In order to allow us to evaluate our RNA-Design algorithm on larger sets of RNA structures with controlled properties, we designed and implemented a simple random generator for pseudoknot-free RNA secondary structures. Parameter ranges set by the user control the number and size of each type of loop in the randomly generated structures. The parameter ranges are summarised in Table 8. Throughout, we use the term “2-branch loop” to refer to a loop that is either an interior loop with two branches but is not a bulge or stacked pair, or to an external loop with two branches, and the term “multiloop” to refer either to an interior loop or external loop with more than two branches.

To generate a structure L given these parameters, the generator chooses numbers I and M uniformly at random from the ranges $[I_1, I_2]$ and $[M_1, M_2]$, respectively. Then the structure is built up in rounds, from the outside in (where the outside corresponds to the free ends), with either a 2-branch loop or a multiloop and associated stems (that include bulges) inserted per round, until the structure has I 2-branch loops and M multiloops. The hairpins are added at the end, closing the stems. To describe a round of the generator in detail, the following notation is useful: If the structure L is represented as a string $L_1 L_2 \dots L_k$ over the alphabet { ‘(’, ‘)’, ‘.’ }, then the insertion of a loop or stem structure l at position j results in the structure $L_1 L_2 \dots L_j l L_{j+1} \dots L_k$. Initially, $L =$ ‘..’, that is, L is a structure with two unpaired bases.

In a round, the type of loop to be inserted is determined as follows. In the first round, in which the type of the external loop is determined, with probability 1/2, the loop

	Hairpins	Stems	2-Branch Loops	Multiloops	Bulges
Size	$[h_1, h_2]$	$[s_1, s_2]$	$[i_1, i_2]$	$[m_1, m_2]$	$[b_1, b_2]$
Number	–	–	$[I_1, I_2]$	$[M_1, M_2]$	$[B_1, B_2]$
Branches	–	–	–	$[mb_1, mb_2]$	–

Table 8
Parameters for structure generator, specified by the user. For each type of loop, the *Size* range is a pair of non-negative integers. For example, the stem size range $[s_1, s_2]$ determines that each stem generated in the structure has a number of base pairs in the range $[s_1, s_2]$, with the number chosen uniformly at random from that range. For loops with two branches and for multiloops, the *Number* range is a pair of non-negative integers, and the total number of 2-branch loops and multiloops in the structure is chosen uniformly at random from that range. For bulges, the number range $[B_1, B_2]$ is contained in $[0, 0.2]$ and the ratio of bulges to base pairs in the stem is chosen uniformly at random from this range. The number of stems and hairpins are determined by the other parameters, and so are not specified. The number of branches is specified for multiloops only.

has one branch, with probability $I/(2(I+M))$ it has two branches (in which case it is counted as a 2-branch loop), and with probability $M/(2(I+M))$ it has more than two branches (in which it is counted as a multiloop). In all rounds other than the first, the probability of a multiloop is $M_i/(I_i + M_i)$ and the probability of a 2-branch loop is $I_i/(I_i + M_i)$, where M_i is M less the number of rounds prior to round i in which multiloops were added, and I_i is I less the number of rounds prior to i in which 2-branch loops were added. If the type of loop is a multiloop, the number of branches is chosen uniformly at random from the range $[B_1, B_2]$.

Then, the size e (number of unpaired bases) of the loop is determined. If the loop has one branch (which can only occur in the first round), e is chosen uniformly at random from the range $[h_1, h_2]$; if it has two branches, e is chosen uniformly at random from the range $[i_1, i_2]$, and if it has more than two branches, e is chosen uniformly at random from the range $[m_1, m_2]$.

Next, positions of the loop branches, or base pairs, other than the closing, or exterior, base pair, are determined. (Note that the external loop has no closing base pair, but all other loops have exactly one closing base pair.) Let b be the number of branches whose positions need to be determined. Thus, b is equal to the number of branches if the loop is the external loop, and is equal to the number of branches less 1 if the loop is not the external loop. Then, a string (structure) with e ‘`.`’s and b ‘`()`’s is chosen uniformly at random from the set of all such strings. If l is the resulting structure, *e.g.* ‘`..()....()..`’ for $e = 8$ and $b = 3$, then the positions of the ‘`()`’ substrings indicate the positions of the branches among the e unpaired bases. In general, we say that position j of a string is *open* if the symbols at positions j and $j+1$ are (and) respectively.

Next, b stems are inserted into l, at the b open positions in the string l. For each stem, its length s (number of base pairs) is chosen uniformly at random from the range $[s_1, s_2]$. Also, a number B is chosen in the range $[B_1, B_2]$, and B times s

(rounded down to an integer) bulge loops are inserted in the stem. Since structures that contain isolated, single base pairs between stretches of unpaired bases tend to be highly unstable, the generator ensures that at least one stacked pair lies between any two bulges. The result is that the structure l contains both a loop and stems associated with all of the branches of the loop, other than the closing branch.

Finally, the structure l (representing a loop and associated stems, other than the closing stem), is inserted into the overall structure L. If $L =$ ‘ . . ’, then l is inserted at position 2 of L, so that L is updated to be ‘ . l . ’. Otherwise, an open position of L is chosen uniformly at random from the set of all such positions, and the structure l is inserted into L at position j. This completes the description of a round of the generator, which adds a loop and associated stems to L.

Once the structure has M multiloops and I internal loops, a hairpin loop is inserted at each remaining open position of L. That is, for each open position j of L, the size h (number of unpaired bases) of the hairpin loop is chosen randomly and uniformly from the range $[h_1, h_2]$ and the string consisting of h ‘ . ’s is inserted at position j of L.

This construction method yields structures that share many of the features of RNA secondary structures found in nature (*e.g.*, rRNA). Salient properties of the generated structures can be controlled by the parameters of the probabilistic process.