A range of complex probabilistic models for RNA secondary structure prediction that includes the nearest-neighbor model and more

Elena Rivas, Raymond Lang and Sean R. Eddy

RNA 2012 18: 193-212 originally published online December 22, 2011
Access the most recent version at doi:10.1261/rna.030049.111

A range of complex probabilistic models for RNA secondary structure prediction that includes the nearest-neighbor model and more

ELENA RIVAS,[1,3] RAYMOND LANG,[2] and SEAN R. EDDY[1]

[1]Janelia Farm Research Campus, Howard Hughes Medical Institute, Ashburn, Virginia 20147, USA
[2]Xavier University of Louisiana, New Orleans, Louisiana 70125, USA

ABSTRACT

The standard approach for single-sequence RNA secondary structure prediction uses a nearest-neighbor thermodynamic model with several thousand experimentally determined energy parameters. An attractive alternative is to use statistical approaches with parameters estimated from growing databases of structural RNAs. Good results have been reported for discriminative statistical methods using complex nearest-neighbor models, including CONTRAfold, Simfold, and ContextFold. Little work has been reported on generative probabilistic models (stochastic context-free grammars [SCFGs]) of comparable complexity, although probabilistic models are generally easier to train and to use. To explore a range of probabilistic models of increasing complexity, and to directly compare probabilistic, thermodynamic, and discriminative approaches, we created TORNADO, a computational tool that can parse a wide spectrum of RNA grammar architectures (including the standard nearest-neighbor model and more) using a generalized super-grammar that can be parameterized with probabilities, energies, or arbitrary scores. By using TORNADO, we find that probabilistic nearest-neighbor models perform comparably to (but not significantly better than) discriminative methods. We find that complex statistical models are prone to overfitting RNA structure and that evaluations should use structurally nonhomologous training and test data sets. Overfitting has affected at least one published method (ContextFold). The most important barrier to improving statistical approaches for RNA secondary structure prediction is the lack of diversity of well-curated single-sequence RNA secondary structures in current RNA databases.

Keywords: RNA secondary structure prediction; probabilistic models; language for RNA grammar parsing; stochastic context-free grammars; maximum likelihood training

INTRODUCTION

Single-sequence RNA secondary structure prediction is important both in its own right (Inoue and Cech 1985; Nahvi et al. 2002; Fujita et al. 2011) and as a component of methods for RNA alignment (Knudsen and Hein 1999; Mathews and Turner 2002; Dowell and Eddy 2006; Bernhart et al. 2008), homology search (Eddy and Durbin 1994; Sakakibara et al. 1994; Nawrocki 2009), and motif finding (Rivas and Eddy 2001; Washietl et al. 2005). Traditionally, RNA secondary structure prediction is performed by energy minimization using the nearest-neighbor model with free-energy parameters fitted to biophysics results from small RNA oligonucleotides. The nearest-neighbor model of RNA structure assumes that the stability of a base pair depends on its adjacent bases, which could be either base-paired (a stacking contribution) or unpaired (a mismatch contribution) (Xia et al. 1998). Thermodynamic implementations of the nearest-neighbor model include Mfold (more recently UNAFold) (Zuker and Stiegler 1981; Zuker 2003; Markham and Zuker 2008), ViennaRNA (Hofacker et al. 1994; Hofacker 2003), and RNAstructure (Mathews et al. 1998; Reuter and Mathews 2010). These models show ample room for improvement (Mathews et al. 1999; Doshi et al. 2004).

Besides thermodynamic parameters, a wealth of other information is available about RNA secondary structure, including a growing number of NMR and crystal structures (Quigley and Rich 1975; Pley et al. 1994; Cate et al. 1996; Ferré-D'Amaré et al. 1998; Ban et al. 2000; Batey et al. 2000, 2004; Wimberly et al. 2000; Kazantsev et al. 2005; Montange and Batey 2008; Yang et al. 2010), databases of RNA sequence families and structures inferred from comparative sequence analysis (Cannone et al. 2002; Gardner et al. 2011), and experimental methods for probing secondary

[3]**Corresponding author.**
E-mail rivase@janelia.hhmi.org.
Article published online ahead of print. Article and publication date are at http://www.rnajournal.org/cgi/doi/10.1261/rna.030049.111.

structure accessibility that can now be applied genome-wide (Kertesz et al. 2010; Underwood et al. 2010). These information sources are statistical, not thermodynamic. This motivates interest in methods that combine statistics and thermodynamic methods (Juan and Wilson 1999; Andronescu et al. 2007) and in methods that are purely statistical (Dowell and Eddy 2004; Do et al. 2006).

Existing statistical methods for RNA secondary structure prediction that train their parameters totally (or partially) from trusted RNA structures include CONTRAfold (Do et al. 2006, 2007), Simfold (Andronescu et al. 2010), and ContextFold (Zakov et al. 2011). These statistical methods, which implement different variants of the nearest-neighbor model, seem to outperform thermodynamic methods (Hamada et al. 2009).

We distinguish two kinds of statistical approach. A discriminative method has a model parameterized with arbitrary values and is usually trained by optimizing the sum of conditional probabilities of structures given sequences in the training set. (The conditional probabilities are obtained by normalizing the structure's Boltzmann factor over all possible structures for the sequence.) A generative method has a model parameterized with probability values. A standard way of implementing probabilistic (generative) models for RNA is by using stochastic context-free grammars (SCFGs) (Durbin et al. 1998). Generative models are trained by optimizing the joint probability of sequences and their structures, which is a much simpler method than discriminative training. Generative models could also be trained by discriminative training. The relative performance of generative versus discriminative training has been studied extensively. The verdict is mixed, and it seems to be dependent on the problem, the model, and the amount of training data (Johnson 2001; Ng and Jordan 2002; Liang and Jordan 2008).

To date, the best RNA folding methods are discriminative. However, limited work has been reported on using generative methods that implement some of the complexity of the nearest-neighbor model (Rivas and Eddy 2000). Applications of SCFGs to RNA structure prediction to date have used much simpler models (Eddy and Durbin 1994; Sakakibara et al. 1994; Knudsen and Hein 1999; Rivas and Eddy 2001). This literature may have led some to conclude that SCFGs cannot be applied to more complex models. For instance, Do et al. (2006) mention "the difficulty of building SCFGs on par with energy-based models." However, there is no technical reason why probabilistic models cannot implement complex features (Nebel and Scheid 2010; Weinberg and Nebel 2010).

The aim of this article is to explore the space of probabilistic models, including parameterizations as complex as the nearest-neighbor model, and to enable the exploration of even more complex models. In order to explore complex SCFGs quickly and consistently, we have created TORNADO, a program that can parse many different grammars under one unified set of algorithms for folding sequences and training parameters. This is an important tool to have because while thermodynamic models are limited by experimentally determined parameters (Xia et al. 1998; Liu et al. 2011), the parameterization of an SCFG (or any other statistical method for that matter) depends only on the existence of enough trusted structures.

Using TORNADO, we explore many different RNA probabilistic models that incorporate all the features of the nearest-neighbor model and many more. We present TORNADO grammar emulations of standard methods that use thermodynamic (ViennaRNA) or arbitrary parameters (CONTRAfold). These emulations use all the parameters of the original models and perform almost identically to the native versions. TORNADO enables the alternative parameterization of these same grammars by probabilistic training, which provides a direct comparison between thermodynamic (or non-energy-based scores) and probabilistic parameterizations. Our main findings are that complex SCFGs on par with thermodynamic models can be made but that currently existing data sets of RNA structures do not support the satisfactory training or benchmarking of these complex (generative or discriminative) models.

RESULTS

A generalized RNA folding "super-grammar" specified by a programming language

TORNADO provides a formal language to express context-free grammars specialized for single-sequence RNA secondary structure. TORNADO builds on previous efforts to develop languages that express entire classes of models. For instance, the Church language is for general purpose stochastic generative models (Goodman et al. 2008). Algebraic Dynamic Programming (ADP) is specific for dynamic programming techniques in bioinformatics (Giegerich et al. 2004; Giegerich and Steffen 2006; Steffen 2006). The TORNADO language is designed to provide compact grammar descriptions for a wide range of RNA structural features such as nearest-neighbor dependencies (e.g., stacking rules or mismatches), including higher than first-order dependencies, and parameterization of arbitrary loop length distributions.

SCFGs consist of nonterminals, terminals (the actual residue emissions), and production rules that recursively determine which strings of terminals the grammar permits (Hopcroft and Ullman 1979). In TORNADO, nonterminals are specified with capital letters, and terminals with lowercase single letters (one letter per emission even if the emission consists of more than one residue). For example, one SCFG that has been used extensively is Pfold (Knudsen and Hein 1999; Pedersen et al. 2006; called g6s by Dowell and Eddy [2004] with stacking added). TORNADO code to produce g6s is

```
# g6s [Pfold grammar with stacking]
S - - > L(i, k) S(k + 1, j) | L                      # Start nonterminal has two rules
L - - > a : i & j   F(i + 1, j - 1) | a : i          # helix starts | one single emission
F - - > a : i & j : i - 1, j + 1 F(i + 1, j - 1) | L S  # helix continues | helix ends
```

This is a verbatim copy of supplemental file "g6s.grm." This grammar has three nonterminals (S, L, F). Each nonterminal has two rules, for a total of six rules. Rules for the same nonterminal can be put together with a | (the "or" symbol), as depicted here, or in separate lines as desired. There are three emitting rules in this grammar, each emitting a different residue type: one single residue emission (a:i), one plain base-pair emission (a:i&j), and one stacked pair emission dependent on the two adjacent outside bases (a:i&j:i−1,j+1). TORNADO's notation assumes that "i" always refers to the 5′ left-most residue and that "j" refers to the 3′ right-most residue. Emitted residues are separated from context residues with a colon, and a base pair is characterized by a "&" to distinguish it from two unpaired bases (e.g., a mismatch emission a:i,j:i−1,j+1). In a non-SCFG, each rule may be associated with an arbitrary score that might depend on the terminals in the rule. For an SCFG, each rule has associated a "transition" probability so that the sum of the transition probabilities for a given nonterminal is one. For each rule, each terminal corresponds to an "emission" (of one or several residues) and has associated a probability distribution. In this example, the existence of transition and emission distributions is specified implicitly by the rules.

Figure 1 shows a more complex grammar coded in TORNADO language. Here is a compact description of those and other more complex features allowed by TORNADO.

Four possible iterators

In addition to the left-most 5′ (i) and right-most 3′ (j) iterators, TORNADO allows up to two intermediate iterators represented by "k" or "l," such that $i \le k \le l \le j$. The i,j (k,l) notation establishes a connection with the actual dynamic programming routines that TORNADO will implement for the grammar. These iterators are not necessary for the formal grammar itself, but they simplify the parser without adding much additional complexity. Some simple rules admit simple forms without explicit iterators (like $S \rightarrow L$ or $S \rightarrow LS$ in the g6s example above), but the use of explicit iterators allows us to describe an arbitrarily large number of complex rules. For instance, a 1 nucleotide (nt) left bulge (a) emitted with the closing base pair (b, $\hat{b}$) and depending on the previously emitted base pair (c, $\hat{c}$) that has the formal grammar notation $P^{c,\hat{c}} \rightarrow ab\,F\,\hat{b}$, in TORNADO adopts the form [$P^{c,\hat{c}} \rightarrow$ a:i,i+1&j:i−1,j+1 F(i+2,j−1)].

Production rules

Production rules can include an arbitrary number of residue emissions, loop emissions, and nonterminals provided that the rule requires no more than four iterators. Examples of possible maximal combinations allowed in TORNADO's rules are as follows: three nonterminals and an arbitrary number of emissions; two nonterminals, one monosegment loop, and an arbitrary number of emissions; and one nonterminal, one disegment loop, and an arbitrary number of emissions.

Arbitrary residue emissions

Emissions can include an arbitrary number of residues and can depend on an arbitrary number of previously emitted residues (contexts). This generalizes the emissions used in the nearest-neighbor model. Typical examples of nearest-neighbor emissions are as follows:

Stacked base pairs [$P^{c,\hat{c}} \rightarrow a\,F\,\hat{a}$]: in which a base pair (a, $\hat{a}$) depends on a contiguous base pair (c, $\hat{c}$) (for arbitrary nonterminals F and $P^{c,\hat{c}}$). In TORNADO language, a:i&j:i−1,j+1 F(i+1,j−1).

Hairpin mismatches [$P^{c,\hat{c}} \rightarrow a\,[m\ldots m]\,b$]: in which the final two bases of a hairpin loop (a,b) depend on the closing base pair (c, $\hat{c}$). In TORNADO language, a:i,j:i−1,j+1 m. . .m(i+1,j−1).

Tetraloops depending on closing base pair [$P^{c,\hat{c}} \rightarrow a_1a_2a_3a_4$]: Hairpin loops with exactly four bases depending on the closing base pair (c, $\hat{c}$). In TORNADO language, a:i,i+1,i+2,i+3:i−1,j+1.

Internal loop mismatches [$P^{c,\hat{c}} \rightarrow a\,[d\ldots]b\,F\,\hat{b}[\ldots d]e$]: where for an internal loop limited by the two base pairs (c, $\hat{c}$) and (b, $\hat{b}$), the closing bases (a, e) depend on the adjacent base pair (c, $\hat{c}$), and the base pair (b, $\hat{b}$) depends on the adjacent bases in the internal loop. In TORNADO language, a:i,j:i−1,j+1 d. . .(i+1,k). . .d(l,j−1) F(k+2,l−2) b:k+1&l−1:k,l.

Left and right dangles [$P^{c,\hat{c}} \rightarrow a\,F \mid F\,a$]: in which a single left (or right) base depends on the adjacent base pair. In TORNADO language, a:i:i−1,j+1 F(i+1,j) or b:j:i−1,j+1 F(i,j−1).

Base pairs depending on left and right dangles [$P^{c} \rightarrow a\,F\,\hat{a}$] [$P^{c,d} \rightarrow a\,F\,\hat{a}$]: in which a base pair (a, $\hat{a}$) depends on the contiguous unpaired bases (c), (d), or both. In TORNADO language, a:i&j:i−1 F(i+1,j−1) or a:i&j:j+1 F(i+1,j−1) or a:i&j:i−1,j+1 F(i+1,j−1).

Other first-order emissions tested with TORNADO and not included in the standard nearest-neighbor model are as follows:

dangles in bulges [$P^{c,\hat{c}} \rightarrow a\,[m\ldots m]b\,F\,\hat{b}$]: in which the end base (a) of a bulge depends on the adjacent base pair

```
# BASIC GRAMMAR [Includes loops and stacking but no dangles]

# PARAMETER DEFINITIONS
# def : param name : param value
def : p-FIT_LENGTH : 30
def : p-MAX_LENGTH : p-FIT_LENGTH

# TRANSITION DISTRIBUTIONS
# tdist : n : t-name
tdist : 5 : t-P
tie : 1 : 2 # tie left and right bulges

# EMISSION DISTRIBUTIONS
# edist : nemit : ncontext : nbasepairs : basepair type : e-name
edist : 1 : 0 : 0 :        e1 # one single residue emission distribution
edist : 2 : 0 : 1 : _WW_ : e1 # one WW basepair distribution (helix opening)
edist : 2 : 0 : 1 : _WW_ : e2 # one WW basepair distribution (helix opening and closing)
edist : 2 : 2 : 1 : _WW_ : e1 # 16 WW basepair stacked distributions (helix extend)
edist : 2 : 2 : 1 : _WW_ : e2 # 16 WW basepair stacked distributions (helix closing)

# LENGTH DISTRIBUTIONS
# ldist : min : fit : max : l-name
# ldist-di : minL : minR : min sum : fit : max : l-name
ldist :           3 : p-FIT_LENGTH : p-MAX_LENGTH : l1 # Hairpin Loops
ldist :           1 : p-FIT_LENGTH : p-MAX_LENGTH : l2 # Bulges
ldist-di : 1 : 1 : 2 : p-FIT_LENGTH : p-MAX_LENGTH : l3 # Internal Loops

# RULES

 S  ->  a : i e1 S(i+1,j) | F0 S | e          # Start: a left base, or a left Helix, or End

F0  ->  a : i & j e1 F5(i+1,j-1)              # Helix starts
F0  ->  a : i & j e2 P (i+1,j-1)              # Helix (of one basepair) ends

F5  ->  a : i & j : i-1,j+1 e1 F5(i+1,j-1)    # Helix continues
F5  ->  a : i & j : i-1,j+1 e2 P (i+1,j-1)    # Helix ends

 P  ->  t-P  m...m(i,j) l1                                # Hairpin Loop
 P  ->  t-P  m...m(i,k) l2 F0(k+1,j)                      # Left Bulges
 P  ->  t-P              F0(i,k-1) m...m(k,j) l2          # Right Bulges
 P  ->  t-P  d...(i,k) ...d(l,j) l3 F0(k+1,l-1)           # Internal Loops
 P  ->  t-P  M2                                           # Multiloop

M2  ->  M1 M                             # TWO or more Helices
 M  ->  M1 M | R                         # ONE or more Helices
M1  ->  F0 | a : i e1 M1(i+1,j)          # ONE Helix, possibly with single left bases
 R  ->  M1 | R(i,j-1) a : j e1           # last Helix, possibly with left/right bases
```

FIGURE 1. TORNADO code for an unambiguous grammar that incorporates loops and stacking. This is an example intended to show how the TORNADO super-grammar works to describe several of the elements of the nearest-neighbor model while still fitting in one page. This "basic_grammar" is at the core of both ViennaRNA or CONTRAfold before terminal mismatches and dangles are added. There are eight nonterminals and 19 rules. Eleven rules include emission terminals. Seven of these emission terminals use a total of 35 predefined residue emission distributions. Residue emission distributions with different attributes (e.g., number of residues, context, base pairs) can have the same name. The tag "_WW_" indicates base pairs between Watson-Crick edges in *cis* for all 4 × 4 residue combinations. There are four loop emitting terminals for hairpins, left and right bulges, and internal loops, which use three length distribution ("l1," "l2," and "l3," respectively). The "l3" length distribution defines two independent segments. The length distributions have a maximum number of emissions determined by "p-MAX_LENGTH," and a maximum number of residues controlled by "p-FIT_LENGTH" after which emissions are fitted to a simple model. In this example, there is a prespecified transition distribution "t-P," which allows us to tie transitions for the left and right bulge loops. The total number of free tied parameters for this grammar is 1022. This figure is a verbatim copy of supplemental file "basic_grammar.grm."

$(c, \hat{c})$, and the closing base pair $(b, \hat{b})$ depends on the adjacent bulge base. In TORNADO language, a:i:i−1,j+1 m...m(i+1,k) b:k+1&j:k F(k+2,j−1).

mismatches (or dangles) in multiloops where multiloop bases contiguous to base pairs depend on the closing base pairs. Details of multiloop dangles are given in the Materials and Methods.

coaxial stacking [$P \rightarrow a\ F\ \hat{a}\ b\ F\ \hat{b}$]: where two contiguous stems with closing base pairs $(a, \hat{a})$ and $(b, \hat{b})$, respectively, have their final base-pair emissions depending on each other. In TORNADO language, a:i&k b:j&k+1:i,k F(i+1,k−1) F(k+2,j−1) or a:i&k,j&k+1 F(i+1,k−1) F(k+2,j−1).

TORNADO can also be used to build second (or higher)-order Markov dependencies, rather than just first order. Examples are

dangles (or more than one single base) depending on several bases [$P^{c,d,e} \rightarrow a\ F \mid a\ b\ F$]: In TORNADO language, a:i:i−1,i−2,i−3 F(i+1,j) and a:i,i+1:i−1,i−2, i−3 F(i+2,j).

higher-order stacked pairs $[P^{b,\hat{b},c,\hat{c}} \rightarrow a\ F\ \hat{a}]$: In TORNADO language, a:i&j:i−1,i−2,j+1,j+2 F(i+1,j−1).

three single bases depending on two base pairs $[P^{e,\hat{e},f,\hat{f}} \rightarrow a\ b\ c\ F]$: In TORNADO language, a:i,i+1,i+2:i−1,i−2, j+1,j+2 F(i+3,j).

Length distributions for loop emission

Monosegment loops (for instance for hairpins, bulges, multiloops or external bases), and disegment loops (for internal loops) can be specified. Disegment loops might include two independent length distributions or a joint one parameterized by the total length of the loop.

Length distribution tails for loop emissions

A length distribution can include a table of specific independent values for lengths up to a value (p-FIT_LENGTH in Fig. 1), and a tail (dependent on a small number of parameters) for lengths larger than p-FIT_LENGTH. Length distribution tails can be specified in TORNADO in the form of affine (for scores) or geometric (for probabilities) extrapolations.

Length distributions for stems

Base pairs can be emitted as stems of arbitrary lengths governed by a length distribution. Stem length distribution can be combined with stacking emission of the actual base pairs. This feature is a natural addition to the standard nearest-neighbor model.

Tying of parameters

Transitions can be tied internally (so that two rules for the same nonterminal share the same value) or externally (so that two different nonterminals can have the exact same transitions). Emission distributions can also be tied so that, for instance, a single residue emission (a:i) could be a marginalization of a mismatch emission (a:i,j), or a mismatch (a:i,j:i−1,j+1) could be the product of two independent dangles (a:i:i−1,j+1) and (b:j:i−1,j+1). A larger list of tying operations for residue emissions has been implemented (see TORNADO's documentation).

Specific distributions

For the purpose of tying parameters, transition, emission, and length distributions can be prespecified as part of the grammar description previous to providing the actual grammar rules.

Specific values

Specific values can be assigned to the different distributions as part of the description of the grammar. These values could be free-energy changes obtained from thermodynamic data or arbitrary scores provided by other means. Setting values is helped by the possibility of defining constants that can be interpreted numerically anywhere in the grammar description (and can be defined by mathematical operations), much like the macro definition directive (#define) works in C programming.

Arbitrary 4 × 4 canonical base pairs and noncanonical base pairs

TORNADO allows distinguishing 18 types of base pairs, depending on the edge (Watson-Crick, Sugar, or Hoogsteen) and the conformation (*cis* or *trans*) of the two bases (Leontis and Westhof 2001). In this work, we only used the canonical base-pairing type (Watson-Crick/Watson-Crick in *cis*) which could involve any of the 4 × 4 possible residue combinations (or be restricted by design to only G-C, A-U, and G-U base pairs).

Comments

Comments can be specified at any time using "#" or "//."

More details are given in the Materials and Methods and in the TORNADO documentation provided as part of the Supplemental Material.

This shows that SCFGs of the standard nearest-neighbor model and beyond are possible and are efficiently represented in the TORNADO language.

Inference algorithms implemented in TORNADO

The training of generative methods is usually done by optimizing the likelihood of the joint distribution of sequences and structures in a training set. This optimization problem has a closed form solution (given the parse trees), in which the parameters are estimated as their frequency of occurrence in the parsing of the pairs (sequence/structure) through the model, so-called maximum likelihood (ML) training.

TORNADO's "grm-train" implements the ML training method given a collection of individual sequences annotated with their secondary structures. In the presence of grammar ambiguity (i.e., when more than one path through the model would produce the same structure), the ML training method is ill-defined (Giegerich 2000). TORNADO arbitrarily chooses at random one of the alternative paths.

Discriminative methods only calculate the conditional probabilities of structures given the sequences and do not perform ML training. The optimization of the conditional distribution of structures given the sequences (conditional ML training [CML]) does not have a closed form and requires costly optimization procedures. CONTRAfold and Simfold were trained using CML. Other discriminative methods improve training time efficiency by avoiding global optimization over the entire training set (Zakov et al. 2011). CML training methods have not been implemented in TORNADO.

Structure prediction has been traditionally performed by reporting the structure with the minimum free-energy change (or the highest probability) obtained using the minimum free-energy algorithm or the Cocke-Younger-

Kasami (CYK) algorithms, which are two flavors of the same dynamic programming algorithm for thermodynamic or statistical methods, respectively. An alternative method (pioneered by Miyazawa for alignment algorithms [Miyazawa 1995] and later applied to hidden Markov models [Holmes 1998]) uses the posterior probabilities of any two positions being base-paired (which are calculated with either the McCaskill [McCaskill 1990] or the Inside/Outside algorithms [Lari and Young 1990, 1991], two flavors of the same algorithm for thermodynamic or statistical methods, respectively). By using those posterior probabilities, one can calculate the structure with the maximal expected accuracy (MEA). Different MEA methods have been implemented for RNA folding, either by maximizing the posterior probabilities of base pairs (Knudsen and Hein 1999; Do et al. 2006) or by calculating centroid estimators (Ding et al. 2005; Hamada et al. 2009). MEA methods have consistently improved performance both for thermodynamic and statistical models. Currently, most RNA folding packages use a MEA estimator to predict RNA structures (Lu et al. 2009).

TORNADO's "grm-fold" implements the CYK algorithm, the MEA method with a sensitivity-specificity trade-off (C-MEA) (Do et al. 2006), the centroid algorithm (CEN) (Ding et al. 2005), and the generalized centroid algorithm (G-CEN) (Hamada et al. 2009). Because there is no operational difference between the folding algorithms used for thermodynamic, statistical, or probabilistic models, TORNADO's folding programs can be used with grammars using any of the three parameterizations.

Specifically for probabilistically trained grammars, TORNADO includes the following: "grm-psample" to sample suboptimal structures from the posterior distribution of structures given a sequence, and "grm-emit" to generate joint sequence/structure pairs from the grammar.

Because TORNADO is designed to be general and capable of accommodating a large number of different features, it also lacks any optimization (Backofen et al. 2009). TORNADO is not meant to be the code for a final product, but rather an exploratory tool to search for better models of RNA secondary structure. The time complexity of the training and folding algorithms in TORNADO is $O(K\,L^3)$ for a sequence of length L and a design-dependent constant K, as is usual for these dynamic programming algorithms. Most TORNADO programs include a Message Passing Interface (MPI) implementation to run on clusters.

Benchmarking tools

Training and test sets

In order to train SCFGs with large numbers of parameters, we need large training sets. To assure proper benchmarking, we also need test sets that are not too similar to the training sets.

We collected the training sets of three previous studies of discriminative methods that implement the nearest-neighbor model (Do et al. 2006; Andronescu et al. 2007, 2010), augmented by a collection of rRNA domains obtained from Lu et al. (2009) as a substitute for the rRNA training set used in a fourth study of lightweight SCFGs (Dowell and Eddy 2004), which utilized full-length SSU and LSU sequences, which are too long to be practical for complex models in TORNADO.

To ensure sequence dissimilarity, we constructed a training set TrainSetA, which eliminates "nearly identical" sequences within a set, and "similar" sequences among the four sets. (The operational definitions of the terms "nearly identical" and "similar" are given in the Materials and Methods.) The training set TrainSetA includes SSU and LSU domains, SRP RNAs, RNase P RNAs, tmRNAs, as well as other small (20- to 50-nt) secondary structures deduced from tertiary structures in the Protein Data Bank. TrainSetA contains 3166 sequences; 47.9% of the residues in TrainSetA are base-paired, of which <0.1% are not A-U, C-G, or G-U base pairs. For details, see Figure 2.

We constructed a test set, TestSetA, using RNA trusted structures compiled from most of the same studies that provided the training sets (Dowell and Eddy 2004; Andronescu et al. 2007, 2010) (the set from Do et al. [2006] is small and was fully incorporated in the training set). In addition, we obtained trusted RNA structures from one more study that tested thermodynamic models (Lu et al. 2009). These four benchmarks include sequences from eight RNA families: tRNA, SRP RNA, tmRNA, RNase P RNA, 5S rRNA, telomerase RNA, group I introns, and group II introns. The original provenance of the trusted sequences is given in the Materials and Methods. The test set TestSetA was constructed by removing nearly identical sequences within the RNA families and similar sequences between families, with a final step in which we remove similar sequences to TrainSetA. TestSetA contains 697 sequences; 51.7% of the residues in TestSetA are base paired, 2.3% of which are not A-U, C-G, or G-U base pairs.

While TrainSetA and TestSetA have been constructed to ensure sequence diversity, they both contain the same types of RNA structures. In order to explore more structural diversity, from the Rfam database (Gardner et al. 2011), we obtained seed alignments for 22 additional RNA families with known three-dimensional structures and no homology with either TrainSetA or TestSetA. The 22 Rfam families include the following: 5.8S rRNA, spliceosomal RNAs (U1, U4), seven riboswitches, two ribozymes, nine *cis*-regulatory RNAs (such as Internal Ribosome Entry Sites, leader and frameshift RNAs), and bacteriophage pRNA.

From those 22 RNA families, a test set TestSetB was constructed by selecting sequences from the seed alignments with no more than 70% identity among each other, and a training set TrainSetB by selecting the rest of the sequences in the seed alignments that have no similarity with either of the test sets. TestSetB contains 430 sequences; 43.6% of the residues in TestSetB are base paired, of which 8.3% are not A-U, C-G, or G-U base pairs. TrainSetB

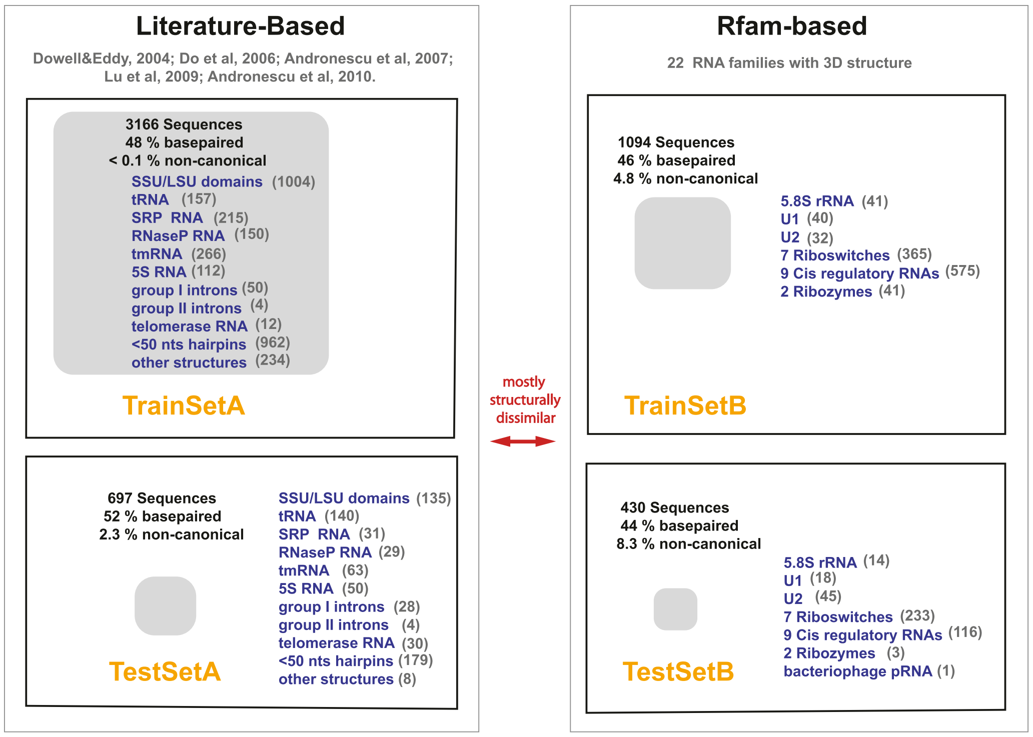

FIGURE 2. Description of the training and test sets. All four sets are composed of nonidentical sequences (i.e., no two sequences from one set have >95% similarity over at least 95% of either sequence). All four sets are composed of dissimilar sequences with respect to each other (i.e., no sequence from one set when compared by blast to another set has a hit with an e-value smaller than 0.0001 over at least 40% of the sequence). In addition, the literature-based "A" sets and the Rfam-based "B" sets are, by construction, structurally dissimilar.

contains 1094 sequences; 46.3% of the residues in TrainSetB are base paired, 4.3% of which are not A-U, C-G, or G-U base pairs.

We observe (Fig. 2) that the Rfam-derived sets have a lower fraction of base pairs and a higher number of noncanonical base pairs than do the sets derived from the literature. Regarding the smaller amount of base-pairing, the annotated Rfam structures are consensus structures for the multiple alignment and are prone to being underestimates of the base-pairing of any individual structure when that consensus is imposed on one individual sequence. Regarding the greater number of noncanonical base pairs, there are several reason for that. For one, alignment uncertainty in regions that align poorly could assign base pairs between misaligned residues. For another, most of the trusted structures reported in the literature had their noncanonical base pairs removed when the original benchmark sets used in TrainSetA were constructed, artificially decreasing their numbers. The Rfam-derived sets TrainSetB and TestSetB have the advantage of providing us with a larger diversity of secondary structures, although these sets possibly contain lower-quality structures, and have the bias that most of the consensus structures reported in the Rfam annotation of the families are predictions (despite some of the actual RNAs having solved three-dimensional structures).

A summary of the two training sets and two test sets is given in Figure 2: (1) The pair TrainSetA/TestSetA is a standard evaluation benchmark, typical of the literature, where the training and test set consist of sequences independent from each other but where the two sets share the same RNA structures. TrainSetA is essentially what CONTRAfold v2.02, Simfold, and ContextFold were trained on. (2) The pair TrainSetB/TestSetB is also composed of independent sequences that share the same structures, but for a different set of structures than those used in the previous case. (3) Using the combinations TrainSetA/TestSetB and TrainSetB/TestSetA allows us to test whether methods trained on one set of structures can predict the other set of independent structures.

Measures of folding accuracy

For a given (single) sequence, we measure folding accuracy by its sensitivity (SEN) and positive predictive value (PPV)

$$\mathrm{SEN} = \frac{\text{True - Pairs - Predicted}}{\text{True - Pairs}},$$
$$\mathrm{PPV} = \frac{\text{True - Pairs - Predicted}}{\text{Pairs - Predicted}}. \qquad (1)$$

Sometimes for simplicity, we use the F measure (or F1 measure, the harmonic mean of the sensitivity and PPV) (van Rijsbergen 1979) as a proxy for prediction accuracy,

$$F = \frac{1}{2}\frac{1}{\frac{1}{SEN} + \frac{1}{PPV}}. \tag{2}$$

For an entire test set of sequences, we use the total SEN and total PPVs,

$$\begin{aligned} \text{total - SEN} &= \frac{\text{total - True - Pairs - Predicted}}{\text{total - True - Pairs}}, \\ \text{total - PPV} &= \frac{\text{total - True - Pairs - Predicted}}{\text{total - Pairs - Predicted}}. \end{aligned} \tag{3}$$

Unless otherwise stated, our predictions are obtained with the C-MEA method (Do et al. 2006). By varying a tunable parameter, C-MEA produces ROC curves between SEN and PPVs. A particularly useful number to describes a given ROC curve is the "best *F*" measure, which corresponds to the maximal *F* among all the points depicted in the ROC curve.

Complex SCFGs implemented in TORNADO

Previous SCFGs used for RNA structure prediction were "lightweight" (Dowell and Eddy 2004). Can "heavyweight" SCFGs, of the complexity of the nearest-neighbor model, compete with state of the art thermodynamic or discriminative models? To address this question in a controlled fashion, we first used TORNADO to implement a grammar emulation of ViennaRNA (Hofacker 2003), named ViennaRNAG, and a grammar emulation of CONTRAfold (Do et al. 2006), named CONTRAfoldG. This allows us to do two kinds of comparisons: ViennaRNA to thermo-ViennaRNAG (ViennaRNAG with thermodynamic free-energy parameters) asks if our emulation is accurate. Then, comparing thermo-ViennaRNAG to probabilistic-ViennaRNAG (ViennaRNAG trained as a probabilistic model) asks how well probabilities perform relative to thermodynamic parameters, and analogously for discriminative methods using CONTRAfold and CONTRAfoldG.

TORNADO grammar emulation of the thermodynamic model ViennaRNA

The TORNADO grammar ViennaRNAG incorporates all the same features with essentially the same parameters as native ViennaRNA. Although both ViennaRNA and ViennaRNAG have the same architecture and score the same features, the number of free parameters in each necessarily differs. This is because thermodynamic models in some places assume zeros when a probabilistic model uses a probability distribution that needs to be normalized. Additionally, we were careful to write an unambiguous grammar (in order to avoid training ambiguities) (Giegerich 2000). This requires the use of a few additional nonterminals (especially for dangles and mismatches). ViennaRNA uses about 12,700 scores, while ViennaRNAG has 14,307 total free parameters after tying. All the 12,700 scores of ViennaRNA are included in ViennaRNAG. For details, see the Materials and Methods, and for a complete specification of the grammar, see supplemental files ViennaRNAG.grm and ViennaRNAGz_wcx.grm.

The control experiment of comparing native ViennaRNA with the TORNADO-emulation ViennaRNAG using the thermodynamic parameters is presented in Figure 3. A small difference is observed in favor of the native implementation for TestSetB. The difference is almost nonexistent for the important section (around the best *F*-value) of the ROC curves.

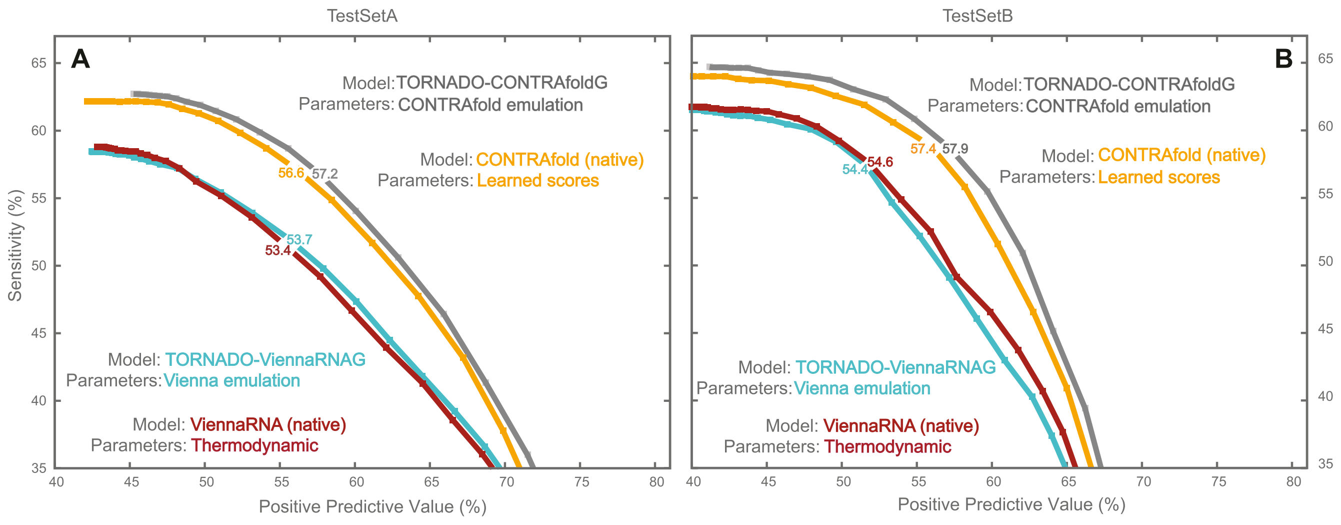

FIGURE 3. Comparison of the standard methods ViennaRNA and CONTRAfold to their TORNADO grammar emulations parameterized with the native values. Panel *A* shows results for TestSetA, and panel *B* shows results for TestSetB. Sensitivity and positive predictive value (PPV) are calculated for structures predicted using the maximal expected accuracy (C-MEA) method Do et al. (2006). The ROC curves have been obtained by varying a tunable positive parameter from 2^{-5} to 2^{10}. The number depicted on each curve corresponds to the maximal value of the *F* parameter, best *F* (defined in the Materials and Methods).

TORNADO grammar emulation of the discriminative model CONTRAfold

The state of the art for folding accuracy is now the CML method CONTRAfold (Hamada et al. 2009), but ML methods of similar complexity have not been tested to our knowledge. We created a TORNADO grammar emulation of CONTRAfold (Do et al. 2006). The TORNADO grammar CONTRAfoldG incorporates all the features of native CONTRAfold, except for two small details: the treatment of hairpin loops of zero and one residues, and a particular contribution for internal coaxials. These two CONTRAfold features involved complicated dependencies that would have required too many additional parameters to reproduce, and in particular for the case of zero and 1-nt hairpin loops, they did not seem worth reproducing. We show that performance is not degraded by those small differences.

CONTRAfoldG has 1276 total free tied parameters after tying, compared to the approximately 300 independent parameters in native CONTRAfold. CONTRAfoldG includes all the parameters of native CONTRAfold. It also needs to incorporate a few more to deal with the particular way in which CONTRAfold handles bulges of length one. More details can be found in supplemental files CONTRAfoldG.grm and CONTRAfoldGu_wcx.grm (the latter removes an ambiguity present in the native CONTRAfold implementation and formally imposes Watson-Crick complementarity).

The control result of comparing native CONTRAfold with the TORNADO-emulation CONTRAfoldG using the original parameters is presented in Figure 3. The CONTRAfoldG emulation seems to reproduce the original results with a systematic but small advantage. Incidentally, both our test sets confirm previous results regarding the better performance of CONTRAfold relative to ViennaRNA (Hamada et al. 2009).

In Figure 3, we also observe that for either ViennaRNA or CONTRAfold, the performance on both test sets is comparable in terms of "best *F*" achieved, although for the Rfam-derived set (TestSetB) that best *F* is achieved at higher sensitivity relative to PPV than in the literature-based method (TestSetA), which is consistent with the expectation that TestSetB has some number of unannotated base pairs.

A spectrum of SCFGs of increasing complexity

The standard nearest-neighbor model of RNA secondary structure implemented in RNA folding programs includes stacking of base pairs, dangles and mismatches for stacked pairs or for terminal pairs in hairpin loops and internal loops, specialized 1 × 2 and 2 × 2 internal loops, 1-nt bulges, length distributions in loops, and base pairs of the G-C, A-U, or G-U type (Xia et al. 1998). Expanded versions of the nearest-neighbor model are starting to explore thermodynamic parameters for free bases in three-branch multiloops (Liu et al. 2011). SCFGs can model all those features and more, and TORNADO gives us the opportunity to test the relative importance of the different elements of the nearest-neighbor model, plus elements beyond those in the current nearest-neighbor model.

In order to assess the contribution of the different features in the nearest-neighbor model, we created a spectrum of TORNADO grammars such that we either delete terms from a more complex grammar (ViennaRNAG or CONTRAfoldG) or add terms to a simpler grammar (g6) or modify an intermediate grammar (such as the basic_grammar in Fig. 1) in either direction.

We have also added new features beyond the current nearest-neighbor model such as specific distributions for bulges of length 1 and 2, dangles in bulges, dangles and mismatches in multiloops with an arbitrary number of branches, coaxial stacking, and length distributions for multiloops and externally emitted bases. All the features that did not degrade performance relative to ViennaRNAG were combined together in grammar "ViennaRNAGz_bulge2_ld_mdangle." We have alternative versions of all grammars allowing either all 4 × 4 possible base pairs or just the G-C, A-U, or G-U types. For details, see Table 1.

Training and performance of complex SCFGs

Benchmark of probabilistically trained ViennaRNAG and CONTRAfoldG shows the need for structurally diverse training sets

For the complex SCFGs that have nonprobabilistic analogs, how do their nonprobabilistic parameterizations compare to a probabilistically trained estimation of the same parameters? To answer that question, we trained ViennaRNAG and CONTRAfoldG using the literature-derived training set TrainSetA, which has close connections to the training sets used with the other trained methods we would like to compare with.

In Figure 4A, we show a standard training/test paradigm. We observe that the probabilistic grammars outperform their thermodynamic and discriminative counterparts: probabilistic-ViennaRNAG improves from 53.7% to 60.2% relative to thermodynamic-ViennaRNAG, and probabilistic-CONTRAfoldG improves slightly from 57.2% to 57.9% relative to CML-CONTRAfoldG.

However, this gain is an artifact due to overfitting, as shown in Figure 4B, where the training set (TrainSetA) and the test set (TestSetB) are structurally nonhomologous. Figure 4B shows (in contrast to Fig. 4A) that probabilistic-ViennaRNAG relative to thermodynamic-ViennaRNAG improves only from 54.1% to 55.0%, while probabilistic-CONTRAfoldG performance decreases from 57.9% to 54.1% relative to CML-CONTRAfoldG.

Overfitting is not limited to probabilistic models. In Figure 4, we also see that among the currently existing statistical methods, ContextFold suffers from the largest effect of differential performance between TestSetA and

TABLE 1. Probabilistic models exploring different structural features

Grammar	Benchmark (set best-F %)	Total free tied parameters 4 × 4 bps	6 bps	Remarks
g6	48.7	21	11	Pfold grammar (Knudsen and Hein 1999)
g6s	49.4	261	41	Pfold + stacking (Dowell and Eddy 2004)
g6_stem	49.7	294	74	Pfold + stacking + helix length dist
basic_grammar_nostack	56.5	572	532	loop length dist
basic_grammar	56.8	1022	582	loop length dist + stacking
basic_grammar_dangle	57.2	1143	643	basic_grammar + dangles
basic_grammar_coaxial	56.5	1279	629	basic_grammar + coaxial stacking
ViennaRNAGz_S	58.1	1862	892	ViennaRNAGz_SimpleInt without tetraloops
CONTRAfoldGS	58.4	2101	811	CONTRAfoldG with simpler 1-nt bulges
basic_grammar_hpfull	58.8	5342	2202	basic_grammar + hairpin tetraloops + hairpin closing mismatches
CONTRAfoldG	58.5	5448	1278	CONTRAfold emulation
ViennaRNAGz_SimpleInt	59.5	6105	2495	ViennaRNAG minus 2 × 2,2 × 1 internal loops
ViennaRNAGz_nostack	59.1	90,497	14,257	ViennaRNAG minus stacking
ViennaRNAG	59.9	90,947	14,307	ViennaRNA emulation
ViennaRNAGz_stem	59.5	90,980	14,340	ViennaRNAG + stem length dist
ViennaRNAGz_ld	60.4	91,012	14,374	ViennaRNAG + all emissions by length dist
ViennaRNAGz_mangle	59.9	91,187	14,397	ViennaRNAG + multiloop mismatches
ViennaRNAGz_coaxial	59.2	91,200	14,350	ViennaRNAG + coaxial stacking
ViennaRNAGz_bulge2	59.9	91,670	14,400	ViennaRNAG + explicit 1,2 bulges
ViennaRNAGz_bulge	59.9	91,766	14,436	ViennaRNAG + explicit 1,2 bulges + bulge dangles
ViennaRNAGz_bulge2_ld_mdangle	60.2	91,977	14,557	ViennaRNAG + explicit 1,2 bulges + all length dist + multiloop mismatches

We have constructed derivative models (with more or less complexity) starting from four key grammars highlighted in bold. All models have been implemented within TORNADO. The third and fourth columns provide the total number of tied parameters before and after imposing base pairs to be canonical. Models are ranked by increasing number of parameters. Parameters include transitions, emissions, and length distributions (dist). Each model is specified by a description file given by the grammar name followed by ".grm" (or "_wcx.grm" in the case of restricting to just A-U, G-C, and G-U base pairs) provided in the Supplemental Material, and written in TORNADO parsing language. A more detailed description of the grammars can be obtained by using the program grm-parse on the grammar description files. The second column provides performance for TestSetA + TestSetB. All grammars were trained on TrainSetA + 2 × TrainSetB.

TestSetB (best *F* for TestSetA is 64.4%, and best *F* for TestSetB is 49.0%). Incidentally, Figure 4 shows that CONTRAfold (as represented by its emulation), even if it is a trained model, is not as affected by overfitting (CONTRAfold's best *F* for TestSetA is 57.2%, and best *F* for TestSetB is 57.9%).

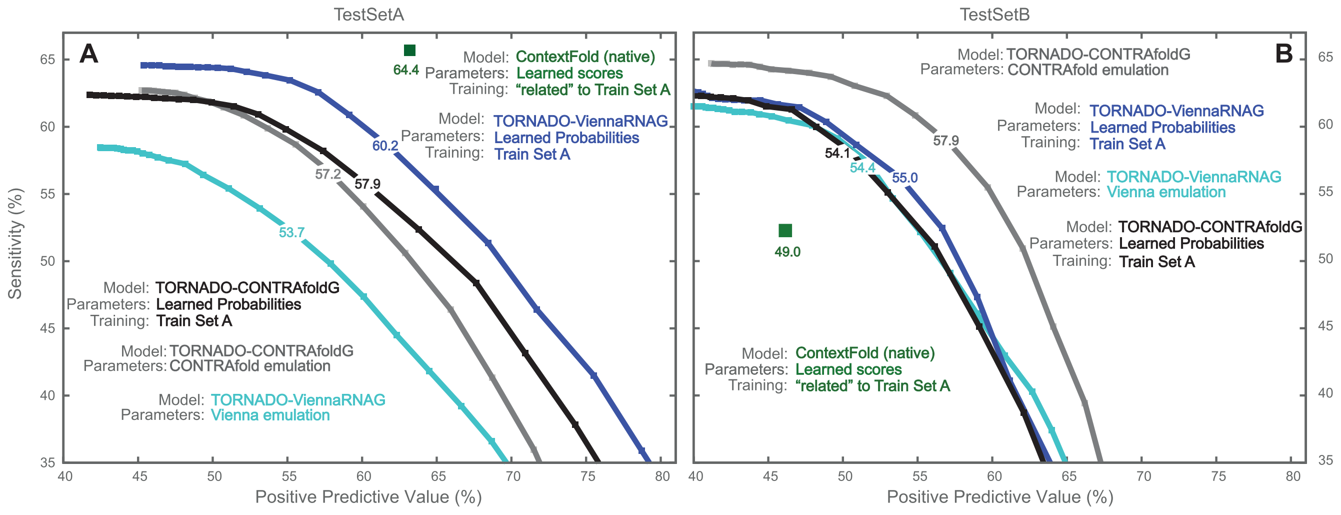

FIGURE 4. Performance of probabilistic ViennaRNAG and CONTRAfoldG. Effect of overfitting in models with large number of parameters. Panel *A* shows results for TestSetA, and panel *B* shows results for TestSetB. We compare the performance of ViennaRNAG and CONTRAfoldG using either the emulation scores or the probabilistic parameters. (The emulation-scores data are the same as in Fig. 3.) The probabilistic versions were trained using TrainSetA. We compare performances for the test sets TestSetA and TestSetB. We also include results for the method ContextFold, which trains a discriminative method with many parameters (∼50,000) in one of the training sets used to construct TrainSetA (more details in Materials and Methods).

We used a Bayesian bootstrapping technique (Rubin 1981) to estimate the standard statistical error in these results. The bootstrapped standard deviation (after 10 resamplings) for SEN and PPV are of the order of 1% for TestSetA and ~2% for TestSetB.

We made sure that ViennaRNA and CONTRAfold are the appropriate comparisons by comparing their performance to other existing methods, including both thermodynamic (UNAFold and RNAstructure) as well as statistical methods (Simfold). These methods show concordance between both test sets, but performance is poor overall. For UNAFold, best *F* is 51.0% for TestSetA and 53.3% for TestSetB; for RNAstructure, best *F* is 53.5% for TestSetA and 53.8% for TestSetB; and for Simfold, best *F* is 54.0% for TestSetA and 55.3% for TestSetB.

We made sure that C-MEA is a reasonable approach to produce predicted structures by comparing to structures predicted by the G-CEN MEA method. G-CEN shows very small variations with respect to C-MEA, oftentimes in the positive direction, but G-CEN does not manifest significant differences from method to method. For instance, the best *F*-values reported in Figure 4 change for probabilistic-CONTRAfoldG from 57.9% to 58.2% for TestSetA and from 57.9% to 58.0% for TestSetB. For probabilistic-ViennaRNAG, best *F* changed from 60.2% to 60.3% for TestSetA and from 54.1% to 54.2% for TestSetB. The G-CEN ROC curves are very similar to the C-MEA ROC curves (data not shown).

TrainSetA uses nine structurally distinct RNA families and compiles all the benchmarks used in the literature for training RNA secondary structure prediction methods. To test the hypothesis that training complex statistical models requires larger structural diversity than just nine different RNA structure families, we constructed another training set, TrainSetB, which includes RNA structures from 22 Rfam RNA families nonhomologous to those used in TrainSetA (see Materials and Methods). TrainSetB contains 1094 sequences, and it has no similar sequences with TrainSetA or any of the test sets TestSetA and TestSetB.

In Table 2, we show that for the larger grammars, both training sets overfit the test set that contains structurally similar RNA sequences. The small "g6" grammar is not affected by overfitting as seen by the similar performance of both test sets under all the different training combinations. However, the overtraining effect already appears for "basic_grammar" where the performance of training set TrainSetA is much better for TestSetA (structurally homologous to TrainSetA) than for TestSetB (structurally dissimilar to TrainSetA) and vice versa. We suspect that the distinct specification of hairpin loops, bulges, internal loops, and multiloops that "basic_grammar" introduces relative to "g6" is responsible for this effect. We have observed (by training grammars on particular families) that the relative occurrence of hairpin loops versus bulges or internal loops or multiloops in different RNA families is highly variable. We also observed (by making chimeric parameterizations taking specific counts from different training sets) that all types of distributions (transitions, emissions and loop lengths) have a contribution to this overfitting effect (data not shown).

We have shown that the literature-based training set TrainSetA does not exhibit the amount of structural diversity necessary to be a good training set for RNA secondary structure prediction when using statistical methods with large numbers of parameters, and that this affects both probabilistic and discriminative methods. In Table 2, we also show that combining the two training sets (as in TrainSetA + 2 × TrainSetB) can bring the performance in both sets to comparable values and still above their overtrained values. We do not know how well these RNA families reflect general properties of RNA secondary structure and how well TrainSetA + 2 × TrainSetB would perform with a hypothetical new RNA structural class. In addition, it would be much more satisfactory to have a well-balanced training set with different structures and sequences instead of performing an ad hoc weighting of different groups of sequences as we do here.

More complex and realistic features require larger training sets than are currently available

Even if we supposed that TrainSetA + 2 × TrainSetB was an unbiased sample of RNA structures, it still might be

TABLE 2. Effect of the structural diversity in the training set

	TrainSetA (set best *F* %)		TrainSetB (set best *F* %)		TrainSetA + TrainSetB (set best *F* %)		TrainSetA + 2 × TrainSetB (set best *F* %)	
Grammar	TestSetA	TestSetB	TestSetA	TestSetB	TestSetA	TestSetB	TestSetA	TestSetB
g6	47.8	46.2	48.5	49.3	48.7	47.0	49.1	47.5
basic_grammar	56.7	53.6	47.5	54.6	57.0	56.5	56.9	56.5
CONTRAfoldG	57.9	54.1	44.4	56.1	58.4	57.4	58.3	58.6
ViennaRNAG	60.2	54.4	42.8	56.0	60.4	57.7	60.2	59.4

We present results using four different training sets combinations for four probabilistic grammars with increasing degree of complexity. The "A" training and test sets are structurally distinct from the "B" sets, but within a type ("A" or "B"); sequences belong to the same structural RNA class. All four sets are dissimilar at the sequence level.

insufficient for the accurate parameterization of a complex grammar. A complex grammar includes parameters for features that occur rarely, such as stacking of noncanonical base pairs. We need to observe enough counts of these features to estimate a probability parameter. In probabilistic models, we can estimate the uncertainty in our parameters estimates resulting from a limited number of counts.

For a given grammar probability distribution $\{t_i\}$ for either residue emissions, loops, or transitions (such that $\sum_i t_i = 1$), the ML estimator of the parameters is $\hat{t}_i = C_i/C$, where C_i is the observed counts associated to t_i, and C stands for the total observed counts for that distribution. The observed counts $\{C_i\}$ are a multinomial random variable with standard deviation $\sigma(C_i) = \sqrt{Ct_i(1-t_i)}$. Thus, the estimated ML probabilities constitute another multinomial random variable with standard deviation $\sigma(\hat{t}_i = C_i/C) = \frac{\sigma(C_i)}{C} = \sqrt{\frac{t_i(1-t_i)}{C}}$. Thus the uncertainty in the probabilistic parameters due to the size of the training set is proportional to $\sqrt{1/C}$.

In Table 3, we observe that residue emission distributions are the most affected by parameter sampling uncertainty. The total number of residue counts is approximately the same for all grammars, and of the order of 630,000. However, the number of emissions distributions increases substantially for larger grammars, so that, for instance, for ViennaRNAG we have some distributions with a relative uncertainty of 0.56. Sampling variability affects the 1 × 2 and 2 × 2 internal loop emission distributions the most. Larger data sets are required for the parameterization of these internal loop emission distributions, as well as for the parameterization of other multiple emissions with high-order contexts. Similarly, training of noncanonical base pairs is affected by undersampling, with many distributions lacking any observations at all (data not shown). For transition distributions, the variability remains more uniform as the complexity of the grammar increases. The total number of transition counts is of the order of 450,000 (for grammars that use loop length distributions). The number of transition distributions (despite unambiguous dangles and mismatches) does not suffer as dramatic an increase as that of residue emissions, and uncertainty remains relatively stable. The situation for loop length distributions is similar to that of transition distributions (data not shown).

More complex and realistic features have incremental effect on folding accuracy

Despite the limitations of our training sets, we can start to assess the relative importance of the different elements of RNA secondary structure, using the range of SCFGs introduced before. There is some variability for the different combinations of training and test sets, but overall we observe that a few distinct features dominate the difference from the worse to the best performing models and that other features (usually very costly in number of parameters) have small incremental effect. We present results using training set TrainSetA + 2 × TrainSetB on the joint test set TestSetA + TestSetB (Table 1; Fig. 5), and we will describe where the results for this combination are not consistently observed for other training/testing combinations.

The most important improvement (of ∼7%–8% depending on the test set, as measured by best F) occurs by expanding simple lightweight SCFGs in order to distinguish hairpin loops, bulges, and internal loops characterized by specific transition, length, and emission distributions. That is the major difference between grammars "g6" and "basic_grammar_nostack" in Table 1. This increases the number of parameters by about 25-fold (from 21–572 in this particular implementation that fits length distributions up to 30 nt) but still keeps the uncertainty in the determination of parameters close to that of the simple SCFGs (see Table 3).

Next in importance come three independent structural elements: mismatches for hairpin loops and internal loops, specific hairpin tetraloops, and stacking of base pairs. All

TABLE 3. Effect of training set size on SCFGs with large number of parameters

	Residue emission distributions			Transition distributions		
		variability ($1/\sqrt{C}$)			variability ($1/\sqrt{C}$)	
Grammar	No. dist	Range	Mean ± SD	No. dist	Range	Mean ± SD
g6	2	[0.0016, 0.0021]	0.0018 ± 0.0004	3	[0.0015, 0.0021]	0.0017 ± 0.0004
basic_grammar_nostack (6 bp)	8	[0.0025, 0.0119]	0.0049 ± 0.0032	8	[0.0025, 0.0121]	0.0056 ± 0.0033
basic_grammar (6 bp)	18	[0.0025, 0.0221]	0.0091 ± 0.0055	8	[0.0025, 0.0121]	0.0056 ± 0.0033
basic_grammar_hpfull (6 bp)	30	[0.0025, 0.0851]	0.0196 ± 0.0182	8	[0.0025, 0.0121]	0.0056 ± 0.0033
ViennaRNAGzS (6 bp)	53	[0.0026, 0.0589]	0.0198 ± 0.0136	12	[0.0025, 0.0203]	0.0089 ± 0.0058
ViennaRNAGz_SimpleInt (6 bp)	59	[0.0026, 0.0851]	0.0227 ± 0.0163	12	[0.0025, 0.0203]	0.0089 ± 0.0058
ViennaRNAGz (6 bp)	163	[0.0024, 0.5774]	0.1189 ± 0.1108	12	[0.0025, 0.0203]	0.0091 ± 0.0057
ViennaRNAGz_bulge2_ld_mdangle (6 bp)	202	[0.0035, 0.5774]	0.1179 ± 0.1031	13	[0.0025, 0.0203]	0.0109 ± 0.0051

For a selection of SCFGs, we report statistics regarding the variability on the estimation of parameters due to sample size. All grammars were trained on TestSetA + 2 × TestSetB.

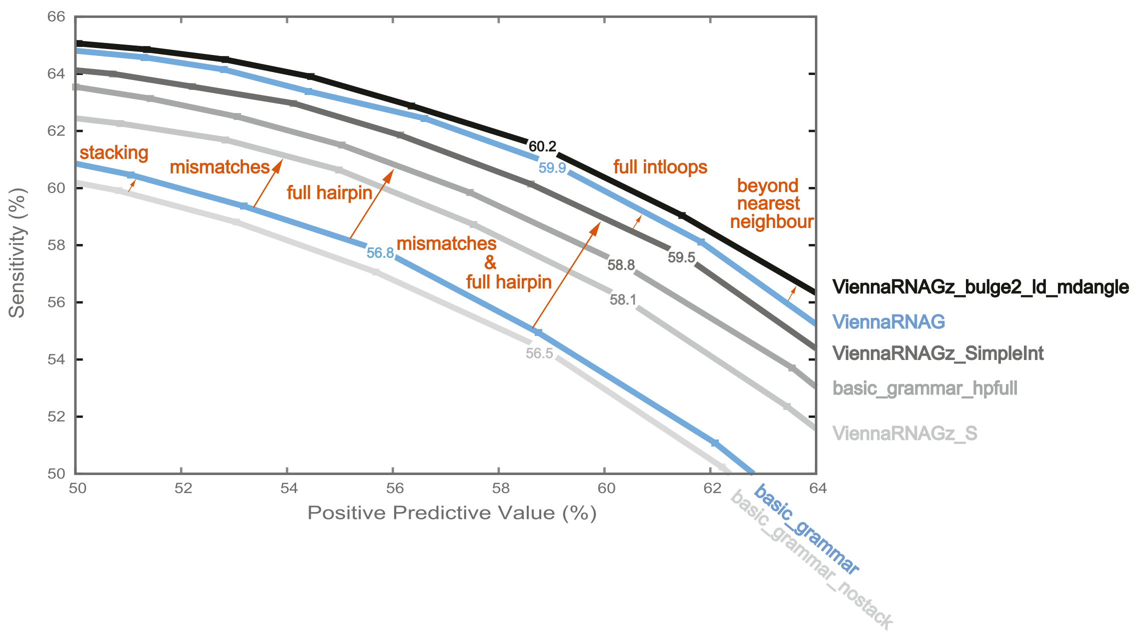

FIGURE 5. A gradient of probabilistic grammars. We have selected seven grammars from a larger range of SCFGs to illustrate the impact on performance of some of the salient features of RNA secondary structure. The simplest grammar presented is "basic_grammar_nostack," which specifies loops with length distributions up to internal loops but does not include stacking or mismatches. The remaining six grammars add features cumulatively on the previous one. "basic_grammar" adds stacking relative to "basic_grammar_nostack." "ViennaRNAGz_S" adds mismatches for hairpin, 1-nt bulges, and internal loops. "basic_grammar_hpfull" adds specific tetraloop emissions relative to "basic_grammar." "ViennaRNAGz_SimpleInt" combines both mismatches and specific tetraloop emissions. "ViennaRNAG" adds specific distributions for 1 × 1, 1 × 2, and 2 × 2 internal loops. Finally, "ViennaRNAGz_bulge2_ld_dangle" incorporates several other features not used so far in any nearest-neighbor model. All SCFGs were trained using TrainSetA + 2 × TrainSetB. We present results for the joint set TestSetA+TestSetB. More details about the grammars are provided in Table 1. More details about the training of these grammars are provided in Table 3.

three improve performance slightly, but their relative contribution is somewhat dependent on the particular training and test set, although stacking is usually the weakest signal. In general, the joint description of these three elements does not produce an additive increase in performance. In Figure 5, we show that for the particular training set/test set combination used in this section, the largest impact is achieved by adding terminal mismatches and specific tetraloops in the description of hairpin loops with a best *F* increase of 2.0% (comparison of "basic_grammar" with "basic_grammar_hpfull"). Next comes the effect of adding mismatches with a best *F* increase of 1.3% (comparison of "basic_grammar" with "ViennaRNAGz_S"). Finally the least important contribution is due to base-pair stacking with a best *F* increase of <0.5%. Similar stacking effects are also observed by comparing grammars "g6" with "g6s," "basic_grammar_nostack" with "basic_grammar," and "ViennaRNAGz_nostack" with "ViennaRNAG" in Table 1.

We find almost no improvement when introducing a length distribution to model the actual nongeometric distribution of helix lengths (helices between 4 and 7 nt are the most abundant). This effect is seen by comparing grammars "g6" with "g6_stem," and "ViennaRNAG" with "ViennaRNAGz_stem" in Table 1. Coaxial stacking of helices also does not seem to provide any overall improvement as seen by comparing grammars "basic_grammar" with "basic_grammar_coaxial" and "ViennaRNAG" with "ViennaRNAGz_coaxial" in Table 1.

The incorporation of other features that are thought to be important in RNA secondary structure tends to improve performance only in small increments. For instance, the 1 × 2- and 2 × 2-specific distributions for internal loops (with contextual dependence on both closing base pairs) increase the number of parameters from 6000 ("ViennaRNAGz_SimpleInt," or the very closely related "basic_grammar_hpfull") to ~92,000, but only improve performance by ~0.5%. The sample variances for some of those residue emissions are very large, leaving the possibility that a larger training set could be able to produce better results for these distributions.

DISCUSSION

In this article, we show that SCFGs can describe generative unambiguous models for RNA structure as complex as those of thermodynamic models. We have developed a large family of related SCFGs that cover the spectrum from the simpler grammars to complex grammars detailing the nearest-neighbor model and beyond (Table 1). In particular, we present two SCFG emulations of current state of the art models, one thermodynamic (ViennaRNA) and the other statistical (CONTRAfold) (Fig. 3). The generation of all the different SCFGs as well as their training and testing has been aided by the introduction of TORNADO, a specific language to parse RNA grammars into a "super-grammar" that can fit all features of RNA structure used so far and more.

Previously, statistical models of RNA folding implementing the nearest-neighbor model have all been chosen to be discriminative, not probabilistic. One reason for this appears to be a misconception that SCFGs of the nearest-neighbor model cannot be built (Do et al. 2006); another reason is the belief that discriminative models have more freedom by not imposing normalization constraints (Lafferty et al. 2001). This work shows that is not the case. SCFGs and CML models of similar complexity have similar performance. The choice between a probabilistic and a discriminative model is not so obvious. We believe that probabilistic models will show advantages in integrating other sources of information; in addition, they are easier to train.

From a practical standpoint, for some applications, we need to combine an RNA folding model with a model of other type of information (comparative analysis, experimental accessibility), and we want to know the trade-offs between performance and number of parameters, given the available training data. TORNADO allows us to perform such exploration of models. We show that for training sets comparable to the ones used here, it seems appropriate to use any of the grammars with 1000–2000 parameters (with mismatches, tetraloops and stacking in some combination) but to leave aside more complex features for which the gain in performance is small.

We have shown that for complex models (SCFGs or otherwise), performance is highly sensitive to the structural diversity present in the training sample, not just to the total size and sequence diversity of the sample. In order to construct robustly trained models and to be able to benchmark different methods, we need to have a larger number of structurally diverse RNA families (perhaps of the order to 50 to 100 structurally distinct families) each containing a large number of diverse single sequences (perhaps of the order of 50 to 100) with well annotated structures.

Perhaps surprisingly, we are far from having this amount of RNA structural data for training statistical methods. Although databases like Rfam collect on the order of a thousand different RNA families, many of these families are structurally similar (e.g., Rfam v10.0 has 477 miRNA families), and the number of dissimilar sequences per family is low (except for a few families such as tRNA and rRNA). Perhaps more importantly, the quality of the annotation of the *individual* structures is also poor, because Rfam is a database of multiple sequence alignments and consensus structures, not individual structures. In addition, the consensus structure itself is frequently just a prediction, often by one of the methods we would be benchmarking.

Ideally, we would collect individual RNA structures from three-dimensional data, but the number of known RNA tertiary structures remains relatively small. Databases such as compaRNA (Puton et al. 2011) that harvest RNA structures from the Protein Data Bank include (after removing rRNAs) only about 250 unique sequences, half of which are <33 nt. Three-dimensional data would also allow us to extract accurate information regarding non-canonical pairs, information that is unreliable in current RNA databases such as Rfam because of inconsistent annotation. At the moment, existing annotation of non-canonical pairs is quite sparse and is not well integrated into a standard sequence-structure annotation (Nagaswamy et al. 2000; Stombaugh et al. 2009). Standardized methods for archiving those individual sequences and structures are also needed (Gardner and Bateman 2009; Bateman et al. 2011).

Therefore, a main conclusion of this work is that statistical methods for RNA secondary structure prediction are promising but held back by the current state of well-annotated RNA structure databases. We strongly support the efforts to create databases of individual RNA structures such as RNA Central (Bateman et al. 2011).

MATERIALS AND METHODS

TORNADO: a specific parser for a large spectrum of RNA grammars

The TORNADO parser includes a lexical interpreter (file grm_parsegrammar.lex) that reads the input file, and a compiler (file grm_parsegrammar.y) that implements a "meta" context-free grammar (the language parser for RNA grammars) and translates the input file for a specific RNA grammar into a generic C structure that can be used by any of the TORNADO inference programs.

Figure 1 shows an example of a grammar described in TORNADO language. Here we provide some description on how to write an RNA grammar with TORNADO. A more complete description is given in the TORNADO documentation. In addition to the actual grammar rules (which come last), one can specify (in the following order) arbitrary parameters, transition distributions, emission distributions, length distributions, and production rules:

1. *Arbitrary parameters* that might be useful later on in the definition of the grammar. Parameter names have to start with "p-." The description is "def: ⟨param_name⟩: ⟨param_value⟩." Parameter values can have dependencies on previously defined parameters. A large number of expressions can be used such as addition, subtraction, multiplication, division, max, min, log, exp, and sqrt, among others.
2. *Transition distributions* can be prespecified for tying purposes; otherwise they get defined internally for each nonterminal. Types of tying allowed for transitions are equating different elements of a given distribution (see Fig. 1), assigning the same distribution to different nonterminals that have the same number of rules, or specifying a particular parameterization of those distributions. Transition distribution names have to start with "t-."
3. *Emission distributions* are specified by providing the number of emissions, contexts, base pairs and the nature of the base pairs, and the emission name separated by a semicolons. The number

of emissions and contexts is in principle unconstrained. Emission names are of the form "e⟨n⟩," where ⟨n⟩ is a natural number. Emissions with different properties (i.e., different number of base pairs or emissions or contexts) can use the same name.

4. *Length distributions* need to specify a minimum length, a maximum length, optionally a "fit" length at which point one assumes an extrapolated tail, and a name for the distribution. Possible distribution tails allowed are "affine," which is used in thermodynamic models, and "linear," which in log space corresponds to assuming a geometric distribution tail. Length distribution names are of the form "l⟨n⟩," where ⟨n⟩ is a natural number.

   Two types of length distribution are allowed: "monosegment," which is used for instance for hairpin loops, and bulges, and "disegment" (ldist-di), which is used for internal loops or stems. Each length distribution is associated with a single residue emission distribution that gets trained but cannot be tied to external emission distributions. For full disegment length distributions, one also needs to specify the minimum number of residues for the left and right segments (see Fig. 1).
5. *Production rules* start with a single nonterminal to the left (as required formally by SCFGs), followed by an arrow "→" followed by an arbitrary number of terminals and nonterminals grouped into rules. A rule is a group of terminals and nonterminals executed together. The different rules associated to a nonterminal can be given all together connected by |s or in separate lines or a combination of the two. The rules for a given nonterminal do not need to be consecutive, and they can appear in between the rules for other nonterminals.

Rules are composed of nonterminals and terminals. Nonterminals are represented by capital letters or by capital letters followed by a natural number. There are four types of terminals: residue terminals, which produce a finite number of residues according to an emission distribution; monosegment and disegment terminals, which produce a variable number of residues according to a length distribution; and the "empty string" terminal. Residue terminals are represented by any lowercase letter with the exception of "e," which is reserved for the "empty string" terminal, and "i," "j," "k," and "l," which are reserved for iterators. Each monosegment terminal "m…m(i,j)" uses a monosegment length distribution. Disegment terminals "d…(i,k) d…(l,j)" can specify a disegment length distribution or a monosegment length distribution, in which case TORNADO assumes that the argument of the distribution is the sum of the two segments. The special stem disegment terminal "d…(i,k)d′…(l,j)" is reserved to the emission of whole stems for which $k - i = j - l$. Stem disegments can be tied to external base pair or stacked base pair emission distributions.

Managing parameters numbers in TORNADO: tied emission distributions

Thermodynamic models often use the same "scores" in situations where a normalized probability model has to use different parameters. For instance, the score of a mismatch depending on a base pair or of a base pair depending on two unpaired bases is obtained from the same free-energy function. In a probabilistic model, those "scores" correspond to two different distributions (one emits two unpaired residues, the other a base pair) that need to be tied if we do not want to have an explosion of parameters. There are ways of tying those distributions by using standard probabilistic relationships of independence, marginalization, and conditioning, as if the scores were considered log probabilities. Some of the tying relationships implemented in TORNADO follows:

- Joint.
  $P(i,j \mid i-1, j+1) = P(i \mid i-1, j+1) * P(j \mid i-1, j+1)$. An example from ViennaRNA is the terminal mismatch in a multiloop, which is the sum of the two dangles dependent on the closing base pair. This is equivalent to an assumption of probabilistic independence.
- Rotation.
  $P(i\&j \mid i-1, j+1) = P(j+1, i-1 \mid j\&i) * P(j\&i)/P(i-1, j+1)$. An example from ViennaRNA is the parameter for an internal loop terminal base pair dependent on the left and right internal dangles, which is set to be identical to the mismatch score of the two "rotated" dangles depending on the "rotated" base pair. The probabilistic model adds two more distributions, the base pair probability $P(j\&i)$ and the mismatch probability $P(i{-}1, j{+}1)$, for which their "equivalent" scores in the thermodynamic model are set to zero by design.

These emission dependencies allow us to establish a link between a thermodynamic (or score-based) implementation of the nearest-neighbor model and a probabilistic one while avoiding an explosion of probabilistic parameters. TORNADO allows other emission-distribution dependencies such as marginalization, conditional dependencies, and others.

Tying emission distributions helps keep the number of grammar parameters under control. In TORNADO it is also possible to restrict a base pair emission to be strictly a A-U, G-C, or G-U pair. For instance, a (A-U;G-C;G-U)-constrained version of the basic grammar in Figure 1 can be found in the supplemental file "basic_grammar_wcx.grm." The number of total free tied parameters decreases from 1022 to 582 in the constrained "basic_grammar." (The basic properties of a grammar can be obtained by running "grm-parse basic_grammar_wcx.grm.")

Managing parameter numbers in TORNADO: unambiguous dangles

One very distinctive property of the nearest-neighbor model is having scores for specific residues dangling off other adjacent bases. Introducing dangles in a generative model unambiguously requires adding some extra nonterminals, but this can be done quite systematically. Starting from the simple grammar in Figure 1, adding dangles at the end of helices requires splitting the role of the S (start) nonterminal into two new nonterminals: S^{+} (for which a left dangle has been emitted previously; thus, one is free to add more residues to the left) and S^{-} (for which no bases can ever be emitted to the left). In addition, the F0 nonterminal (signaling the start of a helix) has to be replaced by four nonterminals: $F0^{++}$ (at which point left and right dangles have been generated; thus, the helix-starting base pair can be conditioned on the two previously emitted left and right bases), $F0^{-+}$ (at which point right dangle(s) have been emitted but no left dangle is allowed; thus, the helix-starting base pair can be conditioned only on the previously emitted right base), $F0^{+-}$ (left dangles have been generated but no right dangles are allowed; thus, the helix-starting base pair can only be conditioned on a left dangle), and $F0^{--}$ (in which case no dangles have been emitted so far; thus, the helix-starting base pair cannot be conditioned on anything). The new rules (in TORNADO pseudo-code) are as follows:

```
$S$          $\rightarrow$ a $S^{+}$ | $S^{-}$ | e                                                  # replaces S $\rightarrow$ $a$ S | F0 S | e
$S^{+}$      $\rightarrow$ a $S^{+}$ | e
$S^{+}$      $\rightarrow$ $F0^{++}$ a $S^{+}$ | $F0^{+-}$ $S^{-}$ | $F0^{+-}$
$S^{-}$      $\rightarrow$ $F0^{-+}$ a $S^{+}$ | $F0^{--}$ $S^{-}$ | $F0^{--}$
$F0^{++}$    $\rightarrow$ a : i & j : i − 1, j + 1 F5 (i + 1, j − 1) | a : i & j : i − 1, j + 1 P (i + 1, j − 1) # bp dependent on L + R-dangle
$F0^{+-}$    $\rightarrow$ a : i & j : i − 1       F5 (i + 1, j − 1) | a : i & j : i − 1       P (i + 1, j − 1) # bp dependent on L-dangle
$F0^{-+}$    $\rightarrow$ a : i & j : j + 1       F5 (i + 1, j − 1) | a : i & j : j + 1       P (i + 1, j − 1) # bp dependent on R-dangle
$F0^{--}$    $\rightarrow$ a : i & j               F5 (i + 1, j − 1) | a : i & j               P (i + 1, j − 1) # bp without dangles
```

Internal mismatches follow a similar design in generative models. New nonterminals $M2^{\alpha\beta}$, $M^{\alpha\beta}$, $M1^{\alpha\beta}$, and $R^{\alpha\beta}$, (with α,β = {+,−}) need to be introduced for bookkeeping of whether dangles have been already emitted (+) or not (−). The rules for internal mismatches in TORNADO pseudo-code are,

```
$P$                 $\rightarrow$ a $M2^{++}$ b | a $M2^{+-}$ | $M2^{-+}$ b | $M2^{--}$   # replaces P $\rightarrow$ M2
$M2^{\alpha\beta}$  $\rightarrow$ $M1^{\alpha+}$a $M^{+\beta}$ | $M1^{\alpha-}M^{-\beta}$             # replace M2 $\rightarrow$ M1 M
$M^{\alpha\beta}$   $\rightarrow$ $M1^{\alpha+}$a $M^{+\beta}$ | $M1^{\alpha-}M^{-\beta}$ | $R^{\alpha\beta}$       # replace M $\rightarrow$ M1 M | R
$M1^{+\beta}$       $\rightarrow$ a $M1^{+\beta}$ | $F0^{+\beta}$                          # replace M1 $\rightarrow$ a M1 | F0
$M^{-\beta}$        $\rightarrow$ $F0^{-\beta}$
$R^{\alpha+}$       $\rightarrow$ $R^{\alpha+}$ a | $M1^{\alpha+}$                           # replace R $\rightarrow$ R a | M1
$R^{\alpha-}$       $\rightarrow$ $M1^{\alpha-}$.
```

See supplemental file "basic_grammar_dangle.grm" for the actual TORNADO code for adding dangles to the simple grammar in Figure 1. Assuming that there are distinct left and right dangle distributions and that mismatches are independent and the same whether they appear externally or in loops (the actual nearest-neighbor model is more complicated than that), the number of total free tied parameters for the simple grammar increases from 1022 to 1143.

One can also add prespecified values to all of the grammar transition, emission, and length parameters, for instance, to implement an existing model with specific parameter values. As an example, a full description of the TORNADO-emulations of the ViennaRNA and CONTRAfold models can be found in the supplemental files ViennaRNAG.grm and CONTRAfoldG.grm.

Given an input file (like Fig. 1), TORNADO interprets and validates the grammar and automatically defines dynamic programming recursions for the different inference programs in correct hierarchy of nonterminals. This description language is sufficient to investigate a wide range of RNA secondary structure models inspired and extending on the nearest-neighbor model.

Training and testing data sets

We aimed to build a comprehensive benchmark from previously published studies. The sequence collections used to generate the training set TrainSetA come from the following sources:

- From Do et al. (2006), file "S-151Rfam" obtained from "http://www.cs.ubc.ca/labs/beta/Projects/RNA-Params/." "S-151Rfam" contains 151 sequences, each from one of the different families tagged as published by Rfam. The total number of residues is 20,581 of which 9848 are base paired (48%). Sequence length varies from 23–568 nt, with an average of 136 nt. An early version of CONTRAfold (v1.0) was trained on S-151Rfam.
- From Andronescu et al. (2007), file "S-Processed-TRA" obtained from "http://www.rnasoft.ca/CG/." "S-Processed-TRA" contains 3439 sequences, 612,414 residues of which 339,420 are base paired (55%). Sequence length varies from 10–695 nt, with an average of 178 nt. "S-Processed-TRA" includes a mixture of SRP RNAs, RNaseP RNAs, tmRNAs, and rRNAs as well as some other secondary structures inferred from tertiary structures in the Protein Data Bank. Models trained on "S-Processed-TRA" include CONTRAfold v2.02, and Simfold-CG 1.1.
- From Andronescu et al. (2010), file "S-Full-Train" obtained from "http://www.cs.ubc.ca/labs/beta/Projects/RNA-Params/." "S-Full-Train" contains 2586 sequences, 691,343 residues of which 371,428 are base paired (61%). Sequence length varies from 10–699 nt, with an average length of 267 nt. The type of RNA molecules in this test set is similar to those of "S-Processed-TRA." Models trained on "S-Full-Train" include Simfold-BL*, Simfold-BL-FR*, and ContextFold.
- From Lu et al. (2009), 88 small-subunit and 27 large-subunit rRNA domains, archived for this work in the Supplemental Material under the name "rRNAdom." "rRNAdom" contains a total of 45,700 residues, of which 24,382 are base paired (53%). Sequence length varies from 72–734 nt, with an average length of 397 nt.

Given these sources, the operational definitions we used to generate TrainSetA (and other sets in this work) are the following: the sequences of a file are said to be "nonidentical" if no two sequences in the file have a BLASTN hit of >95% identity over at least 95% of length of one of the sequences. The sequences of a target file are "dissimilar" to those of a reference file if no target sequence has a BLASTN hit against the reference file with an e-value smaller than 0.0001 over at least 40% of the length of the target sequence. These are relatively relaxed conditions that would still allow to survive for instance a full-length RNA together with a small hairpin (usually extracted from the Protein Data Bank) from that same molecule. Other more restricted definitions of similarity were tested, but they reduced the total number of sequences too much.

The training set TrainSetA was created by merging all nonidentical sequences from all four sources above and then removing similar sequences between the sets. The training set TrainSetA contains 3166 sequences, about half the size of the starting set. TrainSetA contains a total of 630,279 residues, of which 333,466 are base paired (47.9%). Sequence length varies from 10–734 nt, with an average length of 199 nt. TrainxSetA contains <0.1% noncanonical base pairs. The training set TrainSetA can be found in the Supplemental Material under the name "TrainSetA.sto." (Files with ".sto" suffix follow the Stockholm format.) Details regarding the actual occurrences of different RNA types in TrainSetA are given in Figure 2.

The sequences used to generate TestSetA come from the following sources:

- From Steinberg et al. (1993), 1415 tRNAs obtained from "ftp.embl.heidelberg.de" and archived in the Supplemental Material under the name "trna1415_annote.sto." From this data set, we randomly selected 200 nonidentical sequences, so that the test set is not dominated by tRNAs (supplemental file "trna1415_annote_Unique_random200.sto").
- From Dowell and Eddy (2004), 81 SRP RNAs (the Signal Recognition Particle database) (Gorodkin et al. 2001), 97 tmRNAs (the transfer messenger RNA database) (Williams 1999), and 225 RNaseP RNAs (the Ribonuclease P RNA database) (Brown 1999) archived for this work under the names "rnabench_srp.stk," "rnabench_tmRNA.stk," and "rnabench_RNaseP.stk," respectively.
- From Lu et al. (2009), 309 5S rRNAs, 37 telomerase RNA, 16 group I introns, and three group II introns archived for this work under the names "5s.sto," "telomerase.sto," "grpI.sto," and "grp2w.sto," respectively.
- From Andronescu et al. (2007), file "S-Processed-TES" with 974 sequences, obtained from "http://www.rnasoft.ca/CG/." The type of RNA molecules in this test set is similar to those of "S-Processed-TRA." "S-Processed-TES" has been used in previous benchmarks in conjunction with "S-Processed-TRA" as a training set (Andronescu et al. 2007).
- From Andronescu et al. (2010), file "S-Full-Test" with 659 sequences obtained from "http://www.rnasoft.ca/CG/." The type of RNA molecules in this test set is similar to those of "S-Processed-TRA." "S-Full-Test" has been used for benchmarking in conjunction with "S-Full-Train" as a training set (Andronescu et al. 2010; Zakov et al. 2011).

TestSetA was created by merging all nonidentical sequences from all sources above and then removing similar sequences between the sets. In addition, we removed all the sequences similar to the training set TrainSetA. TestSetA contains 697 sequences, 135,939 residues of which 70,214 are base paired (51.7%). Sequence length varies from 10–768 nt, with an average length of 195 nt. TestSetA contains 2.3% noncanonical base pairs. TestSetA can be found in the Supplemental Material under the name "TestSetA.sto." Details regarding the actual sequences present in TestSetA are given in Figure 2.

The literature-based TrainSetA/TestSetA benchmark describes mainly nine distinct RNA secondary structures (with small variations such as the short bacterial versus the large eukaryotic SRP RNAs), and a few small hairpins. (The "S-151Rfam" set does have more structural variation, but because it includes only one sequence per RNA family, the actual effect of that diversity in training is almost negligible.) We used Rfam v10.0 section of "42 families with three dimensional structure" ("http://rfam.sanger.ac.uk/family/browse/with_structure#A") to construct a set that is structurally independent from TrainSetA, a set containing known and diverse structures and also with a relatively large sample of individual sequences. From the 31 families left after removing the ones already included in TrainSetA/TestSetA, we selected sequences from their seed alignments with no more than 70% identity among each other. Finally, 22 RNA families (430 sequences total) survived as being dissimilar to TrainSetA and became the test set TestSetB. TestSetB is composed of 14 5.8S rRNAs, 18 U1 spliceosomal RNAs, 45 U4 spliceosomal RNAs, 233 riboswitches (from seven different families: FMM, glmS, lysine, Purine, PreQ1, SAM and TPP), 116 *cis* regulatory elements (from nine different families), three ribozymes, and one bacteriophage pRNA. TestSetB contains 430 sequences, a total of 52,097 residues of which 22,728 are base paired (43.6%). Sequences in TestSetB vary in length from 27–244 nt, with an average of 121 nt. In TestSetB, 8.3% of the base pairs are noncanonical base pairs.

We also created a training set TrainSetB using the 22 Rfam families above, by selecting the sequences in the seed alignments dissimilar with either TestSetB, TrainSetA, or TestSetA. TrainSetB contains 1094 sequences, a total of 112,398 residues of which 52,065 are base paired (46.3%). Sequence length varies from 27–237 nt, with an average length of 103 nt. TrainSetB contains 4.3% noncanonical base pairs.

All four sets are dissimilar at the sequence level. In addition, the pair "TrainSetB/TestSetB" contains a set of 22 diverse structures mostly different from those included in the pair "TrainSetA/TestSetA."

Measures of folding accuracy

TORNADO implements two slightly different methods to calculate the predicted secondary structure from the posterior probabilities of base pairs. The C-MEA method (Do et al. 2006) maximizes the sum of odd ratios $\gamma \frac{P_{i,j}}{P_i P_j}$, and the generalized centroid estimator (G-CEN) (Hamada et al. 2009) maximizes the sum of $P_{i,j}$s larger than $\frac{1}{1+\gamma}$, where $P_{i,j}$ are the posterior probabilities of positions i and j being base paired, P_i is the posterior probability of position i being single stranded, and γ is a positive tunable real number. By varying the parameter γ (from 2^{-5} to 2^{10}), we produce ROC curves for SEN and PPV. For the experiments in this work, we do not observe significant differences between the two methods.

For a given test set, we use the total-SEN and total-PPV values described in Equation 3. Other investigators (Mathews et al. 1999;

Do et al. 2006; Andronescu et al. 2010; Zakov et al. 2011) use instead the average values over the test set,

$$\text{mean-SEN} = \frac{1}{N}\sum_{i=1}^{N}\text{SEN}(i), \quad \text{mean-PPV} = \frac{1}{N}\sum_{i=1}^{N}\text{PPV}(i). \tag{4}$$

Generally, total SEN and PPV are smaller than the average ones, so that one cannot compare results across studies. The difference is particularly large in our test sets, which contain sequences of quite variable lengths. As an example, the best *F*-values reported in Figure 4A if one uses averages instead of total SEN and PPV increase from 53.7% to 63.7% for ViennaRNAG using native parameters; from 60.2% to 70.8% for probabilistic-ViennaRNAG; from 57.2% to 66.4% for CONTRAfoldG with native scores; from 57.9% to 69.0% for probabilistic-CONTRAfoldG, and from 64.4% to 72.6% for ContextFold.

The relationship between "total" and "mean" accuracy measures can be seen more clearly by rewriting total-SEN and total-PPV (Equation 3) as

$$\begin{aligned}\text{total-SEN} &= \sum_{i=1}^{N}\frac{\text{True-Pairs-Predicted}(i)}{\text{True-Pairs}(i)}\,\frac{\text{True-Pairs}(i)}{\text{total-True-Pairs}}\\ &= \sum_{i=1}^{N}\text{SEN}(i)\,\frac{\text{True-Pairs}(i)}{\text{total-True-Pairs}},\end{aligned} \tag{5}$$

$$\begin{aligned}\text{total-PPV} &= \sum_{i=1}^{N}\frac{\text{True-Pairs-Predicted}(i)}{\text{Pairs-Predicted}(i)}\,\frac{\text{Pairs-Predicted}(i)}{\text{total-Pairs-Predicted}}\\ &= \sum_{i=1}^{N}\text{PPV}(i)\,\frac{\text{Pairs-Predicted}(i)}{\text{total-Pairs-Predicted}}.\end{aligned} \tag{6}$$

The total sensitivity weights the sensitivity of each sequence SEN (i) by the fraction of true pairs that it contains, and it weights the PPV of each sequence PPV (i) by the fraction of total predicted pairs it contains. On the other hand, in the average measures all sequences count as equal, although shorter sequences are easier to predict than longer ones.

Despite the lower absolute values, we consider the "total" method the more appropriate of the two because it avoids giving too much weight to small, easier to predict sequences. Unless otherwise stated, we use total-SEN and total-PPV throughout the article.

Databases and programs

We used the databases Rfam v10.0 from "http://rfam.sanger.ac.uk" and compaRNA (May 11, 2011) from "http://iimcb.genesilico.pl/comparna/." We used several external programs, most of them alternative RNA secondary structure prediction tools. From package ViennaRNA v1.8.4, we used program "RNAfold" with options "-p -MEA" (Hofacker 2003). From package CONTRAfold v2.02, we used program "contrafold predict" with option "- -gamma <x>" (Do et al. 2006). From package RNAstructure v5.2, we used program "partition" with default settings followed by "MaxExpect" with options "-s 1 -g <x>" (Reuter and Mathews 2010). From package UNAFold -3.8, we used program "hybrid-ss-min" with option "-s DAT" (Markham and Zuker 2008). From package MultiRNAFold-2.0, we used program "simfold -p <paramfile>." The parameter files were downloaded from http://www.cs.ubc.ca/labs/beta/Projects/RNA-Params/ on January 4, 2011, and correspond to "parameters_CG1.1.txt," which defines Simfold-CG 1.1; "parameters_BLstar.txt," which defines Simfold-BL*; and "parameters_BLFRstar.txt," which defines Simfold-BL-FR* (Andronescu et al. 2010). From package ContextFold v1.00, we used program "Predict" with default parameters that use their best performing model "StHighCoHigh" (Zakov et al. 2011). For sequence comparison, we used blastall 2.2.22.

Availability

The flex/Bison and ANSI C source code for TORNADO v0.1 is freely available under the GNU General Public License (GPL) from http://selab.janelia.org/. TORNADO's code and documentation (version 0.1), the Perl scripts used to generate the experiments, and the data sets needed to reproduced the results in this work have been collected in a tarball available as part of the Supplemental Material.

SUPPLEMENTAL MATERIAL

Supplemental material is available for this article.

ACKNOWLEDGMENTS

We thank Goran Ceric for smoothly running the computing core at Janelia Farm and the Benasque Center for Science (Spain), where part of this work was developed, for its hospitality. We thank Fred P. Davis, Eric N. Nawrocki, and S.E. Roian Egnor for comments on the manuscript. Work is supported by the Howard Hughes Medical Institute. E.R. was supported at early stages of this work by NIH grant R01 GM070538.

Author contributions: S.R.E. conceived the idea and wrote a prototype of TORNADO's inference code. R.L. wrote a prototype of the TORNADO language. E.R. wrote the code, designed, and carried out the experiments. The manuscript was cowritten by E.R. and S.R.E.

Received August 23, 2011; accepted November 1, 2011.

REFERENCES

Andronescu M, Condon A, Hoos HH, Mathews DH, Murphy KP. 2007. Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics* **23:** i19–i28.

Andronescu M, Condon A, Hoos HH, Mathews DH, Murphy KP. 2010. Computational approaches for RNA energy parameter estimation. *RNA* **16:** 2304–2318.

Backofen R, Tsur D, Zakov S, Ziv-Ukelson M. 2009. Sparse RNA folding: time and space efficient algorithms. In *Proceedings of the 20th Symposium on Combinatorial Pattern Matching*, pp. 249–262. Springer-Verlag, Berlin, Heidelberg.

Ban N, Nissen P, Hansen J, Moore PB, Steitz TA. 2000. The complete atomic structure of the large ribosomal subunit at 2.4 Å resolution. *Science* **289:** 905–920.

Bateman A, Agrawal S, Birney E, Bruford EA, Bujnicki JM, Cochrane G, Cole JR, Dinger ME, Enright AJ, Gardner PP, et al. 2011. RNA central: a vision for an international database of RNA sequences. *RNA* **17:** 1941–1946.

Batey RT, Rambo RP, Lucast L, Rha B, Doudna JA. 2000. Crystal structure of the ribonucleoprotein core of the signal recognition particle. *Science* **287:** 1232–1239.

Batey RT, Gilbert SD, Montange R. 2004. Structure of a natural guanine-responsive riboswitch complexed with the metabolite hypoxanthine. *Nature* **441:** 1172–1175.

Bernhart SH, Hofacker IL, Will S, Gruber AR, Stadler PF. 2008. RNAalifold: improved consensus structure prediction for RNA alignments. *BMC Bioinformatics* **9:** 474. doi: 10.1186/1471-2105-9-474.

Brown JW. 1999. The ribonuclease P database. *Nucleic Acids Res* **27:** 314. doi: 10.1093/nar/27.1.314.

Cannone JJ, Subramanian S, Schnare MN, Collett JR, D'Souza LM, Du Y, Feng B, Lin N, Madabusi LV, Müller KM, et al. 2002. The Comparative RNA Web (CRW) Site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. *BMC Bioinformatics* **3:** 2. doi: 10.1186/1471-2105-3-2.

Cate JH, Gooding AR, Podell E, Zhou K, Golden BL, Kundrot CE, Cech TR, Doudna JA. 1996. Crystal structure of a group I ribozyme domain: principles of RNA packing. *Science* **273:** 1678–1685.

Ding Y, Chan CY, Lawrence CE. 2005. RNA secondary structure prediction by centroids in a Boltzmann weighted ensemble. *RNA* **11:** 1157–1166.

Do CB, Woods DA, Batzoglou S. 2006. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics* **22:** e90–e98.

Do CB, Foo CS, Ng AY. 2007. Efficient multiple hyperparameter learning for log-linear models. In *Advances in neural information processing systems*, Vol. 20, pp. 377–384. MIT Press, Cambridge, MA.

Doshi KJ, Cannone JJ, Cobaugh CW, Gutell RR. 2004. Evaluation of the suitability of free-energy minimization using nearest-neighbor energy parameters for RNA secondary structure prediction. *BMC Bioinformatics* **5:** 105. doi: 10.1186/1471-2105-5-105.

Dowell RD, Eddy SR. 2004. Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. *BMC Bioinformatics* **5:** 71. doi: 10.1186/1471-2105-5-71.

Dowell RD, Eddy SR. 2006. Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics* **7:** 400. doi: 10.1186/1471-2105-7-400.

Durbin R, Eddy SR, Krogh A, Mitchison GJ. 1998. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, Cambridge, UK.

Eddy SR, Durbin R. 1994. RNA sequence analysis using covariance models. *Nucleic Acids Res* **22:** 2079–2088.

Ferré-D'Amaré AR, Zhou K, Doudna JA. 1998. Crystal structure of a hepatitis delta virus ribozyme. *Nature* **395:** 567–574.

Fujita Y, Furuta H, Ikawa Y. 2011. Evolutionary optimization of a modular ligase ribozyme: a small catalytic unit and a hairpin motif masking an element that could form an inactive structure. *NAR* **38:** 3328–3339.

Gardner PP, Bateman A. 2009. A home for RNA families in RNA biology. *RNA Biol* **6:** 2–4.

Gardner PP, Daub J, Tate JG, Moore BL, Osuch IH, Griffiths-Jones S, Finn RD, Nawrocki EP, Kolbe DL, Eddy SR, et al. 2011. Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Res* **39:** D141–D145.

Giegerich R. 2000. Explaining and controlling ambiguity in dynamic programming. In *Proceedings of the 11th Annual Symposium on Combinatorial Pattern Matching* (ed. R Giancarlo and D Sankoff), pp. 46–59. Springer-Verlag, Berlin.

Giegerich R, Steffen P. 2006. Challenges in the compilation of domain specific language for dynamic programming. In *Proceedings of the 2006 ACM symposium on applied computing*. Association for Computing Machine (ACM), New York.

Giegerich R, Meyer C, Steffen P. 2004. A discipline of dynamic programming over sequence data. *Sci Comput Program* **51:** 215–263.

Goodman ND, Mansighka VK, Roy D, Bonawitz K, Tenenbaum JB. 2008. Church, a language for generative models. In *Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, VA.

Gorodkin J, Knudsen B, Zwieb C, Samuelsson T. 2001. SRPDB (signal recognition particle database). *Nucleic Acids Res* **29:** 169–170.

Hamada M, Kiryu H, Sato K, Mituyama T, Asai K. 2009. Prediction of RNA secondary structure using generalized centroid estimators. *Bioinformatics* **25:** 465–473.

Hofacker IL. 2003. Vienna RNA secondary structure server. *NAR* **31:** 3429–3431.

Hofacker IL, Fontana W, Stadler PF, Bonhoeffer LS, Tacker M, Schuster P. 1994. Fast folding and comparison of RNA secondary structures (the Vienna RNA package). *Monatsh Chem* **125:** 167–188.

Holmes I. 1998. "Studies in probabilistic sequence alignment and evolution." PhD thesis, University of Cambridge, Cambridge, UK.

Hopcroft JE, Ullman JD. 1979. *Introduction to automata theory, languages, and computation*. Addison-Wesley, Reading, MA.

Inoue T, Cech TR. 1985. Secondary structure of the circular form of the tetrahymena rRNA intervening sequence: a technique for RNA structure analysis using chemical probes and reverse transcriptase. *Proc Natl Acad Sci* **82:** 648–652.

Johnson M. 2001. Joint and conditional estimation of tagging and parsing models. In *Proceedings of the Association for Computational Linguistics (ACL)*. Morgan Kaufmann Publishers, Toulouse, France.

Juan V, Wilson C. 1999. RNA secondary structure prediction based on free energy and phylogenetic analysis. *J Mol Biol* **289:** 935–947.

Kazantsev AV, Krivenko AA, Harrington DJ, Holbrook SR, Adams PD, Pace NR. 2005. Crystal structure of a bacterial ribonuclease P RNA. *Proc Natl Acad Sci* **102:** 13392–13397.

Kertesz M, Wan Y, Mazor E, Rinn JL, Nutter RC, Chang HY, Segal E. 2010. Genome-wide measurement of RNA secondary structure in yeast. *Nature* **467:** 103–107.

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

Lafferty J, McCallum A, Pereira F. 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *18th Proceedings of the International Conference in Machine Learning*, pp. 282–289. Morgan Kaufmann Publishers, Williamstown, MA.

Lari K, Young SJ. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Comput Speech Lang* **4:** 35–56.

Lari K, Young SJ. 1991. Applications of stochastic context-free grammars using the inside-outside algorithm. *Comput Speech Lang* **5:** 237–257.

Leontis NB, Westhof E. 2001. Geometric nomenclature and classification of RNA base pairs. *RNA* **7:** 499–512.

Liang P, Jordan MI. 2008. An analysis of generative, discriminative, and pseudolikelihood estimators. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*. Omnipress, Helsinki, Finland.

Liu B, Diamond JM, Mathews DH, Turner DH. 2011. Fluorescence competition and optical melting measurements of RNA three-way multibranch loops provide a revised model for thermodynamic parameter. *Biochemistry* **50:** 640–653.

Lu ZJ, Gloor JW, Mathews DH. 2009. Improved RNA secondary structure prediction by maximizing expected pair accuracy. *RNA* **15:** 1805–1813.

Markham NR, Zuker M. 2008. UNAFold: software for nucleic acid folding and hybriziation. In *Bioinformatics, volume II. Structure, function and applications* (ed. JM Keith), pp. 3–31. Humana Press, Totowa, NJ.

Mathews DH, Turner DH. 2002. Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. *J Mol Biol* **317:** 191–203.

Mathews DH, Andre TC, Kim J, Turner DH, Zuker M. 1998. An updated recursive algorithm for RNA secondary structure prediction with improved thermodynamic parameters. In *Molecular modeling of nucleic acids* (ed. NB Leontis and J Santalucia Jr.), pp. 246–257. American Chemical Society, Washington, DC.

Mathews DH, Sabina J, Zuker M, Turner DH. 1999. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J Mol Biol* **288:** 911–940.

McCaskill JS. 1990. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers* **29:** 1105–1119.

Miyazawa S. 1995. A reliable sequence alignment method based on probabilities of residue correspondences. *Protein Eng* **8:** 999–1009.

Montange R, Batey RT. 2008. Structure of the S-adenosylmethionine riboswitch regulatory mRNA element. *Nature* **441:** 1172–1175.

Nagaswamy U, Voss N, Zhang Z, Fox GE. 2000. Database of non-canonical base pairs found in known RNA structures. *Nucleic Acids Res* **28:** 375–376.

Nahvi A, Sudarsan N, Ebert MS, Zou X, Brown KL, Breaker RR. 2002. Genetic control by a metabolite binding mRNA. *Chem Biol* **9:** 1043–1049.

Nawrocki EP. 2009. "Structural RNA homology search and alignment using covariance models." PhD thesis, Washington University School of Medicine, St. Louis, MO.

Nebel ME, Scheid A. 2010. Analysis of the free energy in a stochastic RNA secondary structure model. In *IEEE/ACM Transactions on Computational Biology and Bioinformatics. IEEE computer Society Digital Library. IEEE Computer Society*. http://doi.ieeecomputersociety.org/10.1109/TCBB.2010.126.

Ng AY, Jordan MI. 2002. On discriminative vs. generative classifiers: a comparison of logistic regression and naive bayes. In *Advances in neural information processing systems (NIPS)* (ed. T Dietterich et al.), Vol. 14, pp. 841–848. MIT Press, Cambridge, MA.

Pedersen JS, Bejerano G, Siepel A, Rosenbloom K, Lindblad-Toh K, Lander ES, Kent J, Miller W, Haussler D. 2006. Identification and classification of conserved RNA secondary structures in the human genome. *PLoS Comput Biol* **2:** e33. doi: 10.1371/journal.pcbi.0020033.

Pley HW, Flaherty KM, McKay DB. 1994. Three-dimensional structure of a hammerhead ribozyme. *Nature* **372:** 68–74.

Puton T, Rother K, Kozlowski L, Tkalinska E, Bujnicki JM. 2011. CompaRNA, a server for continuous benchmarking of automated methods for RNA structure prediction. http://iimcb.genesilico.pl/comparna/.

Quigley GJ, Rich A. 1975. Structural domains of transfer RNA molecules. *Science* **194:** 796–806.

Reuter JS, Mathews DH. 2010. RNAstructure: software for RNA secondary structure prediction and analysis. *BMC Bioinformatics* **11:** 129. doi: 10.1186/1471-2105-11-129.

Rivas E, Eddy SR. 2000. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics* **6:** 583–605.

Rivas E, Eddy SR. 2001. Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics* **2:** 8. doi: 10.1186/1471-2105-2-8.

Rubin DB. 1981. The Bayesian bootstrap. *Ann Stat* **9:** 130–134.

Sakakibara Y, Brown M, Underwood RC, Mian IS, Haussler D. 1994. Stochastic context-free grammars for modeling RNA. In *Proceedings of the Twenty-Seventh Annual Hawaii International Conference on System Sciences: Biotechnology Computing*, pp. 284–293. Los Alamitos, CA, IEEE Computer Society Press, Washington, DC.

Steffen P. 2006. "Compiling a domain specific language for dynamic programming." PhD thesis, Bielefeld University, Germany.

Steinberg S, Misch A, Sprinzl M. 1993. Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Res* **21:** 3011–3015.

Stombaugh J, Zirbel CL, Westhof E, Leontis NB. 2009. Frequency and isostericity of RNA base pairs. *Nucleic Acids Res* **37:** 2294–2312.

Underwood JG, Uzilov AV, Katzman S, Onodera CS, Mainzer JE, Mathews DH, Lowe TM, Salama SR, Haussler D. 2010. FragSeq: transcriptome-wide RNA structure probing using high-throughput sequencing. *Nat Methods* **7:** 995–1001.

van Rijsbergen CJ. 1979. *Information retrieval*. London Butterworths, London.

Washietl S, Hofacker IL, Stadler PF. 2005. Fast and reliable prediction of noncoding RNAs. *Proc Natl Acad Sci* **102:** 2454–2459.

Weinberg F, Nebel ME. 2010. Extending stochastic context-free grammars for an application in bioinformatics. *Lect Notes Comput Sci* **6031:** 585–595.

Williams KP. 1999. The tmRNA website. *NAR* **27:** 165–166.

Wimberly BT, Brodersen DE, Jr Clemons WM, Morgan-Warren RJ, Carter AP, Vonrhein C, Hartsch T, Ramakrishnan V. 2000. Structure of the 30S ribosomal subunit. *Nature* **407:** 306–307.

Xia T, Jr SantaLucia J, Burkard ME, Kierzek R, Schroeder SJ, Jiao X, Cox C, Turner DH. 1998. Thermodynamic parameters for an expanded nearest-neighbor model for formation of RNA duplexes with Watson-Crick base pairs. *Biochemestry* **37:** 14719–14735.

Yang S, Parisien M, Major F, Roux B. 2010. RNA structure determination using SAXS data. *J Phys Chem B* **114:** 10039–10048.

Zakov S, Goldberg Y, Elhadad M, Ziv-Ukelson M. 2011. Rich parameterization improves RNA structure prediction. In *RECOMB 2011, LNBI 6577* (ed. V Bafna and SC Sahinalp), pp. 546–562. Springer-Verlag, New York.

Zuker M. 2003. Mfold web server for nucleic acid folding of hyprodization prediction. *NAR* **31:** 3406–3415.

Zuker M, Stiegler P. 1981. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res* **9:** 133–148.