BIOINFORMATICS

Vol. 24 ISMB 2008, pages i68–i76
doi:10.1093/bioinformatics/btn177

A max-margin model for efficient simultaneous alignment and folding of RNA sequences

Chuong B. Do*, Chuan-Sheng Foo and Serafim Batzoglou

Computer Science Department, Stanford University, Stanford, CA 94305, USA

ABSTRACT

Motivation: The need for accurate and efficient tools for computational RNA structure analysis has become increasingly apparent over the last several years: RNA folding algorithms underlie numerous applications in bioinformatics, ranging from microarray probe selection to *de novo* non-coding RNA gene prediction.

In this work, we present RAF (RNA Alignment and Folding), an efficient algorithm for simultaneous alignment and consensus folding of unaligned RNA sequences. Algorithmically, RAF exploits sparsity in the set of likely pairing and alignment candidates for each nucleotide (as identified by the CONTRAfold or CONTRAlign programs) to achieve an effectively quadratic running time for simultaneous pairwise alignment and folding. RAF's fast sparse dynamic programming, in turn, serves as the inference engine within a discriminative machine learning algorithm for parameter estimation.

Results: In cross-validated benchmark tests, RAF achieves accuracies equaling or surpassing the current best approaches for RNA multiple sequence secondary structure prediction. However, RAF requires nearly an order of magnitude less time than other simultaneous folding and alignment methods, thus making it especially appropriate for high-throughput studies.

Availability: Source code for RAF is available at: http://contra.stanford.edu/contrafold/

Contact: chuongdo@cs.stanford.edu

1 INTRODUCTION

The secondary structure adopted by an RNA molecule *in vivo* is a vital consideration in many bioinformatics analyses. In PCR primer design, stable secondary structures can obstruct proper binding of the primer to DNA (Dieffenbach *et al.*, 1993); in RNA folding pathway studies, secondary structure forms the basic scaffold on which more complicated 3D structures organize (Brion and Westhof, 1997); and in computational non-coding RNA gene prediction, RNA secondary structural stability provides the characteristic signal for distinguishing real RNA sequence from non-functional transcripts (Eddy, 2002).

To date, the most powerful non-experimental methods for determining RNA secondary structure rely primarily on position-specific patterns of nucleotide covariation in multiple homologous RNA sequences. Specifically, enrichment for complementarity in pairs of columns from an RNA multiple alignment, especially when primary sequence is not conserved, provides strong evidence for potential base-pairings in the RNA's *in vivo* structure. A primary limitation of covariation analysis, however, is the difficulty of obtaining reliable sequence alignments for divergent RNA families. This shortcoming is especially relevant in the detection of ncRNA genes, as secondary structural constraints often exist even when primary sequence conservation is lacking (Torarinsson *et al.*, 2006).

In this article, we describe RNA alignment and folding (RAF), a new algorithm for predicting RNA secondary structure from a collection of unaligned homologous RNA sequences. Algorithmically, RAF belongs to a category of RNA secondary structure prediction methods which *simultaneously* align and fold RNA sequences. By optimizing a pair of unaligned RNA sequences for both sequence homology and structural conservation concurrently, simultaneous alignment and folding approaches sidestep the usual problem of needing accurate sequence alignments before the folding is done. By exploiting sparsity in the set of likely base pairings and aligned nucleotides, RAF achieves $O(L^2)$ running time for sequences of length L, improving significantly upon the $O(L^4)$ running times of typical simultaneous folding and alignment approaches.

The main contribution of RAF, however, is its application of discriminative machine learning techniques for parameter estimation to the problem of simultaneous alignment and folding. Unlike previous methods, RAF's scoring model does not rely on *ad hoc* combinations of thermodynamic free energies for structural features (Mathews *et al.*, 1999) with arbitrary alignment match and gap penalties (Hofacker *et al.*, 2002), nor does RAF attempt the ambitious task of simultaneously modeling the evolutionary history of both sequences and structure (Knudsen and Hein, 2003). Instead, RAF defines a fixed set of basis features describing aspects of the alignment, RNA secondary structure, or both. RAF then poses the task of learning weights for these features as a convex optimization problem, giving rise to efficient algorithms with guaranteed convergence to optimality.

The concept of using discriminative methods for parameter estimation rather than relying solely on parameters compiled from experimental measurements originated with the CONTRAfold (Do *et al.*, 2006b) program, and later also became the basis of the CG (Andronescu *et al.*, 2007) method. In a manner analogous to these two previous methods for single sequence secondary structure prediction, RAF demonstrates that automatic learning of parameters can also confer benefits to multiple sequence structure prediction accuracy.

2 METHODS

The RAF algorithm consists of four components: (1) a simple yet flexible objective function for pairwise alignment and folding of unaligned RNA sequences; (2) a fast Sankoff-style inference engine for maximizing this objective function via sparse dynamic programming; (3) a simple progressive strategy for extending the pairwise algorithm to handle multiple unaligned sequence inputs; and (4) a max-margin framework for automatically learning model parameters from training data. We describe each of these in turn.

*To whom correspondence should be addressed.

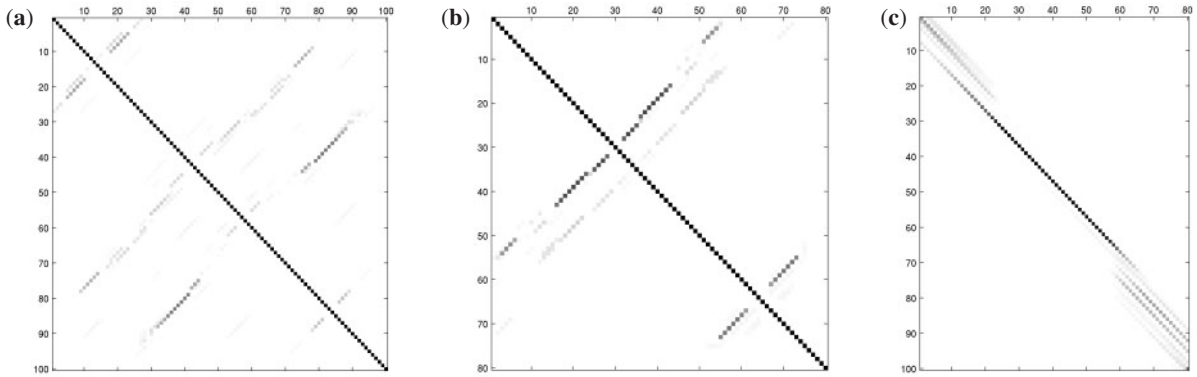

Fig. 1. Sparsity patterns in posterior probability matrices. Panels (**a**) and (**b**) illustrate the pairwise pairing posterior probabilities for two different sequences (such as generated by a single-sequence probabilistic or partition function–based RNA folding program). Panel (**c**) shows the alignment match probabilities for these sequences (such as generated by a probabilistic HMM). In each panel, the darkness of each square represents the posterior confidence in the corresponding base pairing or alignment match. While the single sequence folder or the pairwise sequence aligner may not be able to identify the single correct folding or alignment, respectively, the set of likely candidate base pairings and matched positions, nonetheless, is extremely sparse.

2.1 The RAF scoring model

We begin our description of the algorithm by describing a scoring scheme for alignments and consensus foldings of two sequences. Let a and b be a pair of unaligned input RNA sequences. We refer to a candidate alignment and consensus secondary structure of a and b collectively as a *parse*. Formally, a parse y for a pair of sequences a and b is a set whose elements consist of base pairings (a_i, a_j) belonging to sequence a, base pairings (b_k, b_l) belonging to sequence b, and aligned positions (a_i, b_k) between a and b.

For a given parse y from the space of all valid[1] parses $\mathcal{Y}$, RAF uses a simple scoring scheme which takes into account aligned positions and conserved base pairings. Specifically, RAF defines the score, $\textsc{Score}(y;\mathbf{w})$, of such a parse y to be

$$\sum_{(a_i,b_k)\in y} \psi_{\mathbf{w}}^{\text{aligned}}(i,k) + \sum_{\langle(a_i,a_j),(b_k,b_l)\rangle\in\mathcal{B}(y)} \psi_{\mathbf{w}}^{\text{paired}}(i,j;k,l),$$

where $\psi_{\mathbf{w}}^{\text{aligned}}(i,k)$ and $\psi_{\mathbf{w}}^{\text{paired}}(i,j;k,l)$ are scoring terms for aligned positions and conserved base pairs, respectively, and where $\mathcal{B}(y)$ is the set of all conserved base pairings. In turn, RAF models each scoring term as a linear combination of arbitrary basis features (Appendix A.1):

$$\psi_{\mathbf{w}}^{\text{aligned}}(i,k) = \sum_{p=1}^{n_{\text{aligned}}} w_p \cdot \phi_p^{\text{aligned}}(i,k)$$

$$\psi_{\mathbf{w}}^{\text{paired}}(i,j;k,l) = \sum_{q=1}^{n_{\text{paired}}} w_{q+n_{\text{aligned}}} \cdot \phi_q^{\text{paired}}(i,j;k,l),$$

where $\mathbf{w}\in\mathbb{R}^{n_{\text{aligned}}+n_{\text{paired}}}=\mathbb{R}^n$ is a vector of scoring parameters.

2.2 Fast pairwise alignment and folding

Given the scoring scheme described in the previous section, the problem of simultaneous alignment and folding reduces to the optimization problem,

$$y^* = \arg\max_{y\in\mathcal{Y}} \textsc{Score}(y;\mathbf{w}). \tag{1}$$

In principle, the solution to (1) follows immediately from the original dynamic programming algorithm for simultaneous alignment and folding presented by Sankoff (1985). Sankoff's algorithm, however, has an $O(L^{3K})$ time complexity and $O(L^{2K})$ space complexity for K sequences of length L, rendering it impractical for all but the smallest multiple folding problems. Therefore, most programs for RNA simultaneous alignment and folding use heuristics to reduce time and memory requirements while minimally compromising alignment and structure-prediction quality. Some heuristics used in previous programs have included incorporating structural information into a single alignment scoring matrix (Dalli *et al.*, 2006), disallowing multi-branch loops (Gorodkin *et al.*, 1997), and precomputing potential conserved helices prior to alignment (Tabei *et al.*, 2006; Touzet and Perriquet, 2004).

The most popular heuristics, however, involve reduction of the portion of the dynamic programming matrices (which we call the *DP region*) that must be computed. For example, some methods restrict the DP region to a strip of fixed width about the diagonal (Hofacker *et al.*, 2004; Mathews and Turner, 2002) or about an initial alignment path (Kiryu *et al.*, 2007). Other methods rely on external single-sequence folding and probabilistic alignment programs to generate base pairing probability matrices (Torarinsson *et al.*, 2007; Will *et al.*, 2007) or alignment match posterior probability matrices (Kiryu *et al.*, 2007), and then exploit the sparsity of these matrices in order to reduce the amount of computation required.

The RAF algorithm adopts the last of these strategies. Namely, RAF uses a single-sequence RNA secondary structure prediction program (CONTRAfold; Do *et al.*, 2006b) and a pairwise RNA sequence alignment program (CONTRAlign; Do *et al.*, 2006a),[2] respectively, to construct a *constraint set* $\mathcal{C}$ of allowed base pairs and aligned positions in a and b. Given a constraint set $\mathcal{C}$, RAF then replaces (1) with the reduced inference problem,

$$y^* = \arg\max_{y\in\mathcal{Y}_\mathcal{C}} \textsc{Score}(y;\mathbf{w}), \tag{2}$$

where $\mathcal{Y}_\mathcal{C} = \{y\in\mathcal{Y} : y\subseteq\mathcal{C}\}$ is the space of valid parses, restricted to those which contain only base pairings and alignment matches from the constraint set $\mathcal{C}$ (Fig. 1).

To obtain the set of allowed base pairings, RAF uses the implementation of McCaskill's algorithm (McCaskill, 1990) from CONTRAfold in order to compute the posterior probability of each possible base pairing in sequence a,

[1] We say that a parse y of inputs a and b is *valid* provided that (1) each nucleotide of a and b base pairs with at most one other nucleotide in the same sequence; (2) each nucleotide aligns with at most one nucleotide in the opposite sequence; (3) neither sequence contains pseudo-knotted base pairings; (4) the alignment of the two sequences does not contain rearrangements or repeats; and (5) all base pairings are conserved.

[2] The original CONTRAlign program was designed for protein sequences. We adapted this for RNAs by removing all protein-specific features (e.g. hydrophobicity), modifying the underlying alphabet (A, C, G and U) and simply retraining on the appropriate training set.

Table 1. Comparison of computational complexity of RNA simultaneous folding and alignment algorithms

Algorithm	Time complexity	Space complexity
Sankoff	$O(L^6)$	$O(L^4)$
FOLDALIGN	$O(L^4)$	$O(L^4)$
LocARNA	$O(c^2L^4)$	$O(c^2L^2)$
Murlet	$O(d^2L^2+d^3L^3/\kappa^6)$	$O(d^2L^2)$
RAF	$O(\min(c,d)\cdot cd^2L^2)$	$O(\min(c,d)\cdot cdL)$

Here, L denotes the sequence length, c is the number of candidate base pairs per position, d is the number of candidate alignment matches per position and κ is the minimum allowed distance between adjacent helices.

and similarly for sequence b. All base pairs with posterior probability at least ϵ_{paired} are then retained. Similarly, to determine the set of allowed aligned positions, RAF retains those matches whose posterior probability, according to a version of the CONTRAlign program adapted for RNAs, is at least $\epsilon_{\text{aligned}}$. If these cutoffs $\epsilon_{\text{aligned}}$ and ϵ_{paired} are chosen to be too low, then the reduction of the dynamic programming space achieved for $\mathcal{Y}_{\mathcal{C}}$ will not be significant. Conversely, a higher cutoff could also degrade performance by excluding portions of the DP matrix which actually correspond to the true parse of the input sequences. A similar approach for pruning the space of candidate alignments and folds via *fold and alignment envelopes* was implemented in the Stemloc (Holmes, 2005) program. A number of other programs exploit either base-pairing sparsity (Torarinsson *et al*., 2007; Will *et al*., 2007) or alignment sparsity (Dowell and Eddy, 2006; Harmanci *et al*., 2007; Kiryu *et al*., 2007) separately.

Assuming $O(c)$ and $O(d)$ bounds on the number of candidate base pairing and alignment partners, respectively, per position of both sequences, we show that the time complexity of the RAF algorithm scales *quadratically* in the length of the sequences, while the space complexity scales *linearly* (Appendix B.1). A comparison table of asymptotic time and space complexity of a number of modern RNA simultaneous folding and alignment approaches is shown in Table 1. In practice, we find that RAF's scaling reflects the theoretical bounds, achieving running times often an order of magnitude faster than current simultaneous alignment and folding methods.[3]

2.3 Extension to multiple alignment

Using the RAF pairwise alignment subroutine, we can also address the problem of aligning two alignments. Let S and T be two sets of sequences that we wish to align; furthermore, we denote their corresponding alignments as A and B.

To align a pair of alignments, we first define new basis features $\{\phi_p^{\text{aligned}}(i,k)\}_{p=1}^{n_{\text{aligned}}}$ and $\{\phi_q^{\text{paired}}(i,j;k,l)\}_{q=1}^{n_{\text{paired}}}$ to simply be the average over all pairs of sequences $s\in S$ and $t\in T$ of the basis features for aligning s and t, remapped to the coordinates of the alignments A and B. Second, we define the new constraint set $\mathcal{C}$ for aligning the two alignments to be the union over all pairs of sequences $s\in S$ and $t\in T$ of the constraint sets for each pair, again remapped to the alignment coordinates. Finally, using these new features and our new constraint set, we simply call the existing RAF subroutine for fast-pairwise alignment and folding.

[3]We note that the method described here bears some relation to the 'candidate list' algorithm of Wexler *et al*. (2007), which maintains sparse lists of potential bifurcation points for single sequence folding. By showing that the number of relevant bifurcation points has a negligible dependence on sequence length, the authors provide an effectively quadratic time algorithm for single-sequence folding. Here, our algorithm also relies on sparsity of bifurcation point candidates when dealing with pairwise alignment and folding, but unlike in the previous algorithm, the candidates are provided explicitly via the constraint set $\mathcal{C}$.

Using this new subroutine for aligning alignments, we can then perform multiple alignment in RAF using a standard progressive strategy (Feng and Doolittle, 1987). Specifically, we cluster the sequences with a UPGMA (Sneath and Sokal, 1962) tree-building procedure, using the expected accuracy similarity measure (Do *et al*., 2005). Finally, we perform progressive alignment by aligning subgroups of sequences according to the tree.

2.4 A max-margin framework

Given a set of training examples, $S=\{(a^{(i)},b^{(i)},y^{(i)})\}_{i=1}^m$, the parameter estimation problem is the task of identifying a vector of weights $\mathbf{w}=(w_1,w_2,\ldots,w_n)\in\mathbb{R}^n$ for which the RAF inference algorithm, as described in the previous section, will yield accurate alignments and consensus structures. In this section, we present a max-margin framework for parameter estimation in RAF.

2.4.1 Formulation In the max-margin framework, our goal is to obtain a parameter vector $\mathbf{w}$ for which running the RAF inference algorithm will generate accurate alignments and consensus structures. Clearly, this goal is met if for each training example $(a^{(i)},b^{(i)},y^{(i)})$ from our training set S,[4]

$$\textsc{Score}(y^{(i)};\mathbf{w})>\textsc{Score}(y';\mathbf{w}),\qquad \forall y'\in\mathcal{Y}_{\mathcal{C}}^{(i)}\setminus\{y^{(i)}\}. \tag{3}$$

In such a case, we would be guaranteed that the maximum of (2) is attained for $y^*=y^{(i)}$ (provided the true parse $y^{(i)}$ belongs to $\mathcal{Y}_{\mathcal{C}}^{(i)}$), and hence our inference procedure would necessarily return the correct alignment and consensus folding. This intuition is captured in the following convex optimization problem:

$$\begin{aligned}\underset{\mathbf{w}\in\mathbb{R}^n,\boldsymbol{\xi}\in\mathbb{R}^m}{\text{minimize}}\ & \tfrac{1}{2}C||\mathbf{w}||^2+\tfrac{1}{m}\textstyle\sum_{i=1}^m\xi_i\\ \text{subject to}\ & \textsc{Score}(y^{(i)};\mathbf{w})-\textsc{Score}(y';\mathbf{w})\\ &\quad\geq\Delta(y^{(i)},y')-\xi_i,\\ &\qquad\qquad i=1,\ldots,m,\\ &\qquad\qquad y'\in\mathcal{Y}_{\mathcal{C}}^{(i)}\cup\{y^{(i)}\}.\end{aligned} \tag{4}$$

Here, C is a regularization constant, and $\Delta(y^{(i)},y')$ is a non-negative distance measure between pair of parses, conventionally referred to as the *loss function*, which takes value 0 if and only if its two arguments are equal (Section 2.4.2).

The inequality constraints play the role of (3)—they try to ensure that the training output $y^{(i)}$ scores higher than any alternative incorrect parse y' by some positive amount $\Delta(y^{(i)},y')$. In cases where this condition is not achieved, the objective function incurs a penalty of ξ_i. Finally, the regularization term $(\tfrac{1}{2})C||\mathbf{w}||^2$ is a penalty used to prevent overfitting.[5]

2.4.2 The loss function The loss function $\Delta(y^{(i)},y')$ in (4) plays two significant roles. Technically, the loss function establishes an appropriate scale for the parameters of the problem and prevents the trivial solution, $\mathbf{w}=\mathbf{0}$. Intuitively, however, the loss function also helps to make the max-margin optimization robust. By choosing a loss function that takes large positive values for incorrect candidate outputs y' that differ from the true output $y^{(i)}$ in a very critical way, but that takes small positive values for incorrect candidate outputs y' whose errors are more forgivable, the loss function allows the user to implement a notion of 'cost' for different types of mistakes in the max-margin model.

For RAF, we defined the loss function by restricting our attention to four types of parsing errors: (1) *false positive base-pairings* ($(a_i,a_j)\in y'\setminus y^{(i)}$, or

[4]Note that our notation hides the dependencies of the Score function on each of the input sequences $a^{(i)}$ and $b^{(i)}$, and similarly for the unconstrained and constrained space of parses, $\mathcal{Y}^{(i)}$ and $\mathcal{Y}_{\mathcal{C}}^{(i)}$.

[5] By default, we used $C=1$. We found that when running the online Pegasos optimization algorithm (Section 2.4.3) for a fixed number of iterations, the resulting generalization performance for RAF is relatively insensitive to the value of C used, provided that C is not too large.

similarly in sequence b), (2) *false negative base-pairings* ($(a_i, a_j) \in y^{(i)} \setminus y'$, or similarly in sequence b), (3) *false positive aligned matches* ($(a_i, b_k) \in y' \setminus y^{(i)}$) and (4) *false negative aligned matches* ($(a_i, b_k) \in y^{(i)} \setminus y'$). Then, we set

$$\begin{aligned}\Delta(y^{(i)}, y') = {} & \gamma^{\text{FP paired}} \cdot \text{No. of false positive base pairings} \\ & + \gamma^{\text{FN paired}} \cdot \#\text{'s. of false negative base pairings} \\ & + \gamma^{\text{FP aligned}} \cdot \#\text{'s. of false positive aligned matches} \\ & + \gamma^{\text{FN aligned}} \cdot \#\text{'s. of false negative aligned matches.}\end{aligned}$$

The numbers $\gamma^{\text{FN paired}}$, $\gamma^{\text{FP paired}}$, $\gamma^{\text{FN aligned}}$ and $\gamma^{\text{FP aligned}}$ are hyperparameters, chosen by the user prior to training the RAF algorithm, which allow the user to express her preference for models with either high sensitivity or high specificity for base-pairing positions and aligned nucleotides.[6]

2.4.3 Optimization algorithm At first glance, the constrained optimization problem stated in (4) appears to be a standard convex quadratic program and hence solvable using off-the-shelf packages for convex programming. In reality, for each training example, the optimization problem has an exponential number of inequalities, one corresponding to each possible candidate parse y' of the input sequences! Despite our use of constraints sets to reduce the set of allowed candidate outputs, in most cases, this space is still too large to enumerate.

One approach to deal with this problem is an iterative algorithm known as *constraint generation* (or *column generation*), as used in the program CG (Andronescu *et al.*, 2007). In this approach, the parameter vector $\mathbf{w}_t$ at each time t is the solution to a reduced version of (4) in which only a small subset of the constraints are retained. Next, one checks if $\mathbf{w}_t$ violates any of the constraints of the original full optimization problem by more than an prescribed tolerance of ϵ. If so, the worst violated constraint is added to the current set of constraints to form a new reduced optimization problem, whose solution, in turn, gives the next iterate $\mathbf{w}_{t+1}$. If not, the optimization algorithm terminates. Each of the optimization problems in the sequence requires a quadratic programming solver.

Here, we take a simpler approach based on the recent SVM training algorithm of (Shalev-Shwartz and Singer, 2007) and Shalev-Shwartz *et al.* (2007). Omitting details, we begin by converting (4) into an equivalent unconstrained problem: namely, minimize (with respect to $\mathbf{w} \in \mathbb{R}^n$),

$$f(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \max_{y' \in \mathcal{Y}_C^{(i)} \cup \{y^{(i)}\}} \Big\{ \text{SCORE}(y'; \mathbf{w}) + \Delta(y^{(i)}, y') - \text{SCORE}(y^{(i)}; \mathbf{w}) \Big\} + \frac{1}{2} C ||\mathbf{w}||^2. \tag{5}$$

Next, we use strong duality from optimization theory in order to derive an upper bound B on the norm of the optimal solution of our unconstrained problem (Appendix C.1). Finally, we actually run the optimization procedure by applying the simple update rule,

$$\mathbf{w}_{t+1} \leftarrow \mathbf{\Pi}_B \left[\mathbf{w}_t - \frac{1}{Ct} \cdot \mathbf{g}_t \right], \tag{6}$$

starting from $\mathbf{w}_1 = \mathbf{0}$. Here, $\mathbf{g}_t \in \partial f(\mathbf{w}_t)$ is any subgradient of the objective function $f(\mathbf{w})$ evaluated at $\mathbf{w} = \mathbf{w}_t$, and the operator $\mathbf{\Pi}_B[\cdot]$ projects a vector onto an origin-centered ball of radius B (i.e. $\mathbf{\Pi}_B[\mathbf{v}] = (B/||\mathbf{v}||)\mathbf{v}$ if $||\mathbf{v}|| > B$ and $\mathbf{\Pi}_B[\mathbf{v}] = \mathbf{v}$ otherwise). Intuitively, the algorithm works much like a standard gradient descent procedure adapted for non-differentiable objective functions, but with the added twist that the projection operation ensures that the weight vector iterates stay with a region of the parameter space where the optimum is known to exist.

Given an existing routine for computing subgradients of the unconstrained objective, this algorithm can be implemented in a few lines of code with no complicated numerical optimization software. As shown by Singer and Shalev-Shwartz, the algorithm is also quite efficient, requiring only $\tilde{O}(m/C\epsilon)$ iterations to achieve ϵ accuracy on a training set of m examples. An online variant of the algorithm, in which the subgradients $\mathbf{g}_t$ in each step are computed based only on a randomly sampled subset of the training data (e.g. a single example), achieves an $\tilde{O}(1/C\epsilon)$ expected running time, independent of m, the size of the training set.

2.4.4 Subgradient computation Finally, we show how to compute a subgradient $\mathbf{g}_t \in \partial f(\mathbf{w}_t)$. In order to simplify notation, define an n-dimensional vector $\mathbf{\Phi}(y)$ whose pth component is

$$\Phi_p(y) = \begin{cases} \displaystyle\sum_{(a_i, b_k) \in y} \phi_p^{\text{aligned}}(i,k) & \text{if } 1 \le p \le n_{\text{aligned}} \\ \displaystyle\sum_{\langle (a_i, a_j), (b_k, b_l) \rangle \in \mathcal{B}(y)} \phi_{p - n_{\text{aligned}}}^{\text{paired}}(i,j;k,l) & \\ & \text{if } n_{\text{aligned}} + 1 \le p \le n, \end{cases}$$

from which it follows that $\text{SCORE}(y; \mathbf{w}) = \mathbf{w}^T \mathbf{\Phi}(y)$. We can apply the usual rules for computing subgradients (see, e.g. Bertsekas *et al.*, 2003) to obtain

$$\mathbf{g}_t = C\mathbf{w}_t + \frac{1}{m} \sum_{i=1}^{m} \left(\mathbf{\Phi}(y_*^{(i)}) - \mathbf{\Phi}(y^{(i)}) \right), \tag{7}$$

where $y_*^{(i)}$ is simply any y' which attains the maximum in the ith term of the summation in (5), for $\mathbf{w} = \mathbf{w}_t$. Each 'loss-augmented' maximization, in turn, is easily performed by modifying the original RAF inference procedure to incorporate an appropriately defined additional scoring matrix, $\phi_0(i,j;k,l)$, with fixed weight $w_0 = 1$.

3 RESULTS

To evaluate the performance of RAF on real data, we collected training and testing data from a variety of sources. In particular, for training, we obtained Rfam 8.1 (Griffiths-Jones *et al.*, 2005), a database of alignments and covariance models for RNA families along with annotated secondary structures where available. For testing, we obtained BRAliBASE II (Gardner *et al.*, 2005), a benchmark set for RNA alignment programs. We also obtained a testing set of RNA families used by the authors of the recent program, MASTR (Lindgreen *et al.*, 2007).

An important concern in the validation of RNA alignment programs is the confounding factor that unless cross-validation is properly performed, the performance that one sees on any given validation set is not likely to be a reliable judge of the program's performance on future data. Even in cases where the training and evaluation tests are disjoint but still contain sequences from the same RNA family, evaluation can still give misleading results, because the weights learned for loop lengths and composition will be biased toward specific properties of that RNA family.

To be absolutely sure of no contamination between training and testing data, we preprocessed our Rfam training set of alignments and consensus structures (October 2007 version, 607 families) by excluding all families for which either of the two testing databases contained an example from that family. We then also removed all families for which only automatically predicted consensus structures were known, leaving a total of 154 families. Finally, we generated a training set $\mathcal{T}_1$ of up to 10 randomly sampled pairwise alignments with consensus structures from each remaining family (1361 pairwise alignments in total), a training set $\mathcal{T}_2$, of up to 10 randomly sampled sequences with structures from each family (1179 sequences in total), and a training set $\mathcal{T}_3$, containing one

[6] By default, we used $\gamma^{\text{FN paired}} = 10$, and $\gamma^{\text{FP paired}} = \gamma^{\text{FN aligned}} = \gamma^{\text{FP aligned}} = 1$ in order to emphasize prediction of correct base pairings.

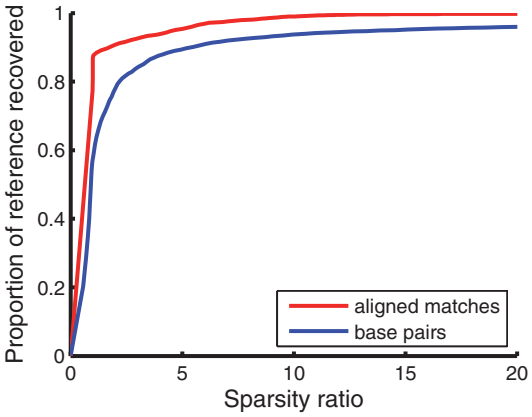

Fig. 2. Trade-off between sparsity factor and proportion of reference base-pairings or aligned matches covered when varying the cutoffs ϵ_{paired} and $\epsilon_{\text{aligned}}$. This graph was made using training set $\mathcal{T}_3$.

randomly sampled five-way multiple alignment from each family (118 multiple alignments in total).

RAF uses two external programs, CONTRAlign (Do *et al.*, 2006a) and CONTRAfold (Do *et al.*, 2006b), to compute alignment match and base-pairing posterior probabilities, respectively. To ensure proper cross-validation, CONTRAlign was retrained from scratch using $\mathcal{T}_1$, and CONTRAfold was retrained using $\mathcal{T}_2$. Finally, the RAF algorithm itself was trained using all pairwise projections of each multiple alignment of $\mathcal{T}_3$. Our strict cross-validation procedure significantly reduces both the size and coverage of the training sets used for CONTRAlign and CONTRAfold, and thus places RAF at a significant disadvantage in the comparisons shown here. Nonetheless, as shown in the following sections, RAF performs well, indicating its ability to generalize for sequences not present in the training set.

3.1 Alignment and base-pairing constraints

To observe the effects of different cutoffs $\epsilon_{\text{aligned}}$ and ϵ_{paired}, we computed the proportions of reference base pairings and reference aligned matches recovered for varying cutoff constraints. In addition, we also computed the sparsity ratio (i.e. the maximum number of pairing partners or matching partners for any nucleotide, averaged over the entire training set) for each cutoff. A plot of these two values for training set $\mathcal{T}_3$ is shown in Figure 2. As seen in the figure, nearly complete coverage of base pairings and alignment matches can be retained when each sparsity factor is roughly 10.[7]

3.2 Evaluation metrics

To evaluate the quality of the resulting alignments, we used five different scoring measures:

(1) the standard sum-of-pairs (SP) score (Thompson *et al.*, 1999), which computes the proportion of matches in a reference alignment which are present in the predicted alignment,

[7]In practice, we found that using cutoffs of $\epsilon_{\text{aligned}} \sim 0.01$ and $\epsilon_{\text{paired}} \sim 0.002$ gave a good trade-off between speed and accuracy of our algorithm when using CONTRAlign and CONTRAfold; these cutoffs correspond roughly to average sparsity factors of ~10 each, respectively.

Table 2. Performance comparison on BRAliBASE II datasets. The best number in each column is marked in bold

Dataset	Program	Time (s)	SP	Sens	PPV	MCC
5S rRNA	Murlet	687	0.94	**0.70**	**0.70**	**0.70**
	LocARNA	812	0.93	0.55	0.60	0.57
	RNA Sampler	2361	0.90	0.55	0.64	0.59
	RAF	**87**	**0.95**	0.66	0.66	0.66
group II intron	Murlet	962	**0.78**	0.75	**0.76**	**0.75**
	LocARNA	250	0.74	0.79	0.65	0.72
	RNA Sampler	1626	0.72	0.77	0.65	0.71
	RAF	**48**	0.78	**0.83**	0.65	0.73
SRP	Murlet	20548	**0.88**	**0.75**	**0.78**	**0.76**
	LocARNA	22467	0.85	0.66	0.70	0.68
	RAF	**1290**	0.87	0.72	0.71	0.70
tRNA	Murlet	525	0.93	0.86	0.90	0.88
	LocARNA	246	**0.95**	0.86	0.90	0.88
	RNA Sampler	763	0.92	**0.93**	**0.91**	**0.92**
	RAF	**52**	0.94	0.81	0.85	0.83
U5	Murlet	1772	**0.84**	0.69	0.75	0.72
	LocARNA	549	0.80	0.56	0.61	0.58
	RNA Sampler	4084	0.77	0.75	0.70	0.72
	RAF	**99**	0.82	**0.83**	**0.79**	**0.81**

(2) sensitivity (Sens), the proportion of base pairings in a reference parse which are recovered in the predicted parse,

(3) specificity or positive-predictive value (PPV), the proportion of base pairings in a predicted parse which are also present in the reference parse, and

(4) the Matthews correlation coefficient (MCC) (Matthews, 1975), which we approximate as $\sqrt{\text{Sens} \cdot \text{PPV}}$, following Gorodkin *et al.* (2001).

3.3 Comparison of accuracy

In our first accuracy assessment, we evaluated RAF as well as a number of other current RNA secondary structure prediction programs using the BRAliBASE II dataset. In particular, the first dataset from BRAliBASE II contains collections of 100 five-sequence subalignments, sampled from five specific Rfam families (5S rRNA, group II intron, SRP, tRNA and U5). For each of these alignments, we ran a number of current multiple-sequence RNA secondary structure prediction programs, including Murlet v0.1.1 (Kiryu *et al.*, 2007), LocARNA v1.2.2a (Will *et al.*, 2007), and RNA Sampler v1.3 (Xu *et al.*, 2007). Wherever any of these programs required access to external pairing-posterior probabilities, we used ViennaRNA v1.7 (Hofacker *et al.*, 1994). The results of the comparison are shown in Table 2.

As seen from the table, on the BRAliBASE II benchmark, RAF attains comparable accuracy to the other methods, achieving either the best or second-best overall accuracy according to MCC on four out of the five datasets. The running time of the method, however, is dramatically faster than the other algorithms, often taking an order of magnitude less time than many of the other programs.

We also obtained the dataset used in the benchmarking of the MASTR RNA secondary structure prediction program. For a number of different programs, pre-generated predictions for each input file

Table 3. Performance comparison on MASTR benchmarking sets. The best number in each column is marked in bold.

Program	SP	Sens	PPV	MCC
CLUSTAL W + Alifold	0.81	0.57	0.73	0.65
FoldalignM	0.78	0.38	**0.81**	0.55
LocARNA	0.75	0.41	0.77	0.56
MASTR	0.84	0.64	0.73	0.68
Murlet	**0.89**	0.62	0.78	0.70
RNAforester	0.53	0.55	0.55	0.55
RNA Sampler	0.82	0.65	0.70	0.67
RAF	0.88	**0.68**	0.77	**0.72**

are available for download on the MASTR website. In addition to scoring these pre-generated predictions, we also generated and scored predictions using Murlet and RAF. The results are shown in Table 3. In this benchmark set, RAF obtains the highest overall MCC.

We emphasize, however, that benchmarking results such as these should be taken with a grain of salt; both the BRAliBASE II and MASTR benchmarking sets are extremely restricted in their coverage of the space of RNA families, choosing to focus on a few individual RNA families only. As a result, methods carefully tuned to the benchmarks may perform less well on diverse RNA families not found in either of these benchmarks. By using cross-validation, we improve the chances that RAF's validation results really do indicate reliable out-of-sample performance.

We also note that the performance of RAF on particular RNA families is often closely related to the accuracy of the underlying alignment and single-sequence models used to derive folding and alignment constraints. Because the tools involved in the RAF pipeline all rely on automatic parameter learning, RAF allows the possibility of learning custom parameter sets well-suited for predictions on particular RNA families.

4 DISCUSSION

We presented RAF, a new tool for simultaneous folding and alignment of RNA sequences which exploits sparsity in base pairing and alignment probability matrices and max-margin training in order to achieve faster running times and higher accuracy than previous tools.

Besides its speed, one principal advantage of the RAF methodology is its use of a flexible scoring function for combining an arbitrary set of functions into a coherent objective function for alignment scoring. The ability to introduce new basis scoring functions into the RAF scoring model means that there remains a rich space of possible features to explore.

In addition, the use of the max-margin framework to identify relevant linear combinations of scoring functions has other promising potential applications. For example, Wallace *et al.* (2006) recently introduced M-Coffee, a meta-algorithm for protein sequence alignment, which combines the results of several different protein sequence alignment programs using the T-Coffee framework. The difficulty of identifying appropriate weights for the various programs used in the M-Coffee scoring scheme (i.e. some heuristically derived tree-based weights the authors tried did not give a significant improvement in accuracy over flat weights), led the authors to rely on a uniform weight model, treating programs known to be more accurate on equal footing with less accurate aligners. The max-margin framework developed in this paper obviates the need for heuristically-derived weights altogether.

ACKNOWLEDGEMENTS

C.B.D. was supported by an NSF Graduate Research Fellowship. C.S.F. was supported by an A*STAR National Science Scholarship. This material is based in part upon work supported by the NSF under grant number EF-0312459.

Conflict of Interest: none declared.

REFERENCES

Andronescu,M. *et al.* (2007) Efficient parameter estimation for RNA secondary structure prediction. *Bioinformatics*, **23**, 19–28.

Bertsekas,D.P. *et al.* (2003) *Convex analysis and optimization*. Athena Scientific.

Brion,p. and Westhof,E. (1997) Hierarchy and dynamics of RNA folding. *Annu. Rev. Biophys. Biomol. Struct.*, **26**, 113–137.

Dalli,D. *et al.* (2006) STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time. *Bioinformatics*, **22**, 1593–1599.

Dieffenbach,C.W. *et al.* (1993) General concepts for PCR primer design. *PCR Methods Appl.*, **3**, 30–37.

Do,C.B. *et al.* (2005) PROBCONS: probabilistic consistency-based multiple sequence alignment. *Genome Res.*, **15**, 330–340.

Do,C.B. *et al.* (2006a) CONTRAlign: discriminative training for protein sequence alignment. In *RECOMB*, pp. 160–174.

Do,C.B. *et al.* (2006b) CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, **22**, e90–e98.

Dowell,R.D. and Eddy,S.R. *et al.* (2006) Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. *BMC Bioinformatics*, **7**, 400.

Eddy,S.R. (2002) Computational genomics of noncoding RNA genes. *Cell*, **109**, 137–140.

Feng,D.F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *J. Mol. Evol.*, **25**, 351–360.

Gardner,P.P. *et al.* (2005) A benchmark of multiple sequence alignment programs upon structural RNAs. *Nucleic Acids Res.*, **33**, 2433–2439.

Gorodkin,J. *et al.* (1997) Finding the most significant common sequence and structure motifs in a set of RNA sequences. *Nucleic Acids Res.*, **25**, 3724–3732.

Gorodkin,J. *et al.* (2001) Discovering common stem-loop motifs in unaligned RNA sequences. *Nucleic Acids Res.*, **29**, 2135–2144.

Griffiths-Jones,S. *et al.* (2005) Rfam: annotating non-coding RNAs in complete genomes. *Nucleic Acids Res.*, **33**, D121–D124.

Harmanci,A.O. *et al.* (2007) Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. *BMC Bioinformatics*, **8**.

Hofacker,I.L. *et al.* (1994) Fast folding and comparison of RNA secondary structures (The Vienna RNA Package). *Monatsh. Chem.*, **125**, 167–188.

Hofacker,I.L. *et al.* (2002) Secondary structure prediction for aligned RNA sequences. *J. Mol. Biol.*, **319**, 1059–1066.

Hofacker,I.L. *et al.* (2004) Alignment of RNA base pairing probability matrices. *Bioinformatics*, **20**, 2222–2227.

Holmes,I. (2005) Accelerated probabilistic inference of RNA structure evolution. *BMC Bioinformatics*, **6**.

Kiryu,H. *et al.* (2007) Murlet: a practical multiple alignment tool for structural RNA sequences. *Bioinformatics*, **23**, 1588–1598.

Knudsen,B. and Hein,J. *et al.* (2003) Pfold: RNA secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res.*, **31**, 3423–3428.

Lindgreen,S. *et al.* (2007) MASTR: multiple alignment and structure prediction of non-coding RNAs using simulated annealing. *Bioinformatics*, **23**, 3304–3311.

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

Mathews,D.H. *et al.* (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *J. Mol. Biol.*, **288**, 911–940.

Matthews,B.W. (1975) Comparison of predicted and observed secondary structure of T4 phage lysozyme. *Biochim. Biophys. Acta.*, **405**, 442–451.

McCaskill,J.S. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119.
Sankoff,D. (1985) Simultaneous solution of the RNA folding, alignment and protosequence problems. *SIAM J. Appl. Math.*, **45**, 810–825.
Shalev-Shwartz,S. and Singer,Y. (2007) Logarithmic regret algorithms for strongly convex repeated games, 2007.
Shalev-Shwartz,S. *et al.* (2007) Pegasos: Primal estimated sub-gradient solver for svm. In *ICML*, pp. 807–814.
Sneath,P.H. and Sokal,R.R. (1962) Numerical taxonomy. *Nature*, **193**, 855–860.
Tabei,Y. *et al.* (2006) SCARNA: fast and accurate structural alignment of RNA sequences by matching fixed-length stem fragments. *Bioinformatics*, **22**, 1723–1729.
Taskar,B. *et al.* (2003) Max-margin markov networks. In *NIPS 16*.
Thompson,J.D. *et al.* (1999) A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Res.*, **27**, 2682–2690.
Torarinsson,E. *et al.* (2006) Thousands of corresponding human and mouse genomic regions unalignable in primary sequence contain common RNA structure. *Genome Res.*, **16**, 885–889.
Torarinsson,E. *et al.* (2007) Multiple structural alignment and clustering of RNA sequences. *Bioinformatics*, **23**, 926–932.
Touzet,H. and Perriquet,O. (2004) CARNAC: folding families of related RNAs. *Nucleic Acids Res.*, 32 (Web Server), W142–W145.
Wallace,I.M. *et al.* (2006) M-Coffee: combining multiple sequence alignment methods with T-Coffee. *Nucleic Acids Res.*, **34**, 1692–1699.
Wexler,Y. *et al.* (2007) A study of accessible motifs and RNA folding complexity. *J. Comput. Biol.*, **14**, 856–872.
Will,S. *et al.* (2007) Inferring noncoding RNA families and classes by means of genome-scale structure-based clustering. *PLoS Comput. Biol.*, **3**.
Xu,X. *et al.* (2007) RNA Sampler: a new sampling based algorithm for common RNA secondary structure prediction and structural alignment. *Bioinformatics*, **23**, 1883–1891.

APPENDIX

A.1 RAF FEATURES

The features used by the RAF program, as evaluated in this article, consist of alignment features, $\{\phi_p^{\text{aligned}}(i,k)\}$ and pairing features, $\{\phi_q^{\text{paired}}(i,j;k,l)\}$. Specifically, the alignment features, $\boldsymbol{\phi}^{\text{aligned}}(i,k) \in \mathbb{R}^4$ for a candidate alignment match (a_i, b_k) are

$$\begin{bmatrix} P(a_i \text{ aligns with } b_k) \\ P(a_i \text{ aligns with } b_k)^2 \\ \log P(a_i \text{ aligns with } b_k) \\ -\log P(a_i \text{ unaligned}) - \log P(b_k \text{ unaligned}) \end{bmatrix}. \tag{A1}$$

The pairing features, $\boldsymbol{\phi}^{\text{paired}}(i,j;k,l) \in \mathbb{R}^4$ for a conserved base pairing $\langle (a_i,a_j),(b_k,b_l)\rangle$ are given by $\boldsymbol{\phi}^{\text{paired}}(i,j;k,l) = \boldsymbol{\phi}^{\text{paired}}(a_i,a_j) + \boldsymbol{\phi}^{\text{paired}}(b_k,b_l)$. In turn, $\boldsymbol{\phi}^{\text{paired}}(a_i,a_j) \in \mathbb{R}^4$ is given by

$$\begin{bmatrix} P(a_i \text{ pairs with } a_j) \\ P(a_i \text{ pairs with } a_j)^2 \\ \log P(a_i \text{ pairs with } a_j) \\ -\log P(a_i \text{ unpaired}) - \log P(a_j \text{ unpaired}) \end{bmatrix}, \tag{A2}$$

and similarly for $\boldsymbol{\phi}^{\text{paired}}(b_k,b_l)$. Thus, the model contains a total of eight features whose weights must be learned. Here, the posterior probabilities for aligned positions and base-pairing positions are computed using the CONTRAlign (Do *et al.*, 2006a) and CONTRAfold (Do *et al.*, 2006b) programs, respectively.

B.1 THE RAF INFERENCE ENGINE

In the section, we describe the RAF inference engine for fast approximate simultaneous alignment and consensus folding for pairs of sequences. In particular, we first present some exact recurrences for alignment and folding, and then use restrictions on the set of allowed base pairings and aligned positions to achieve an improvement in computational complexity.

B.1.1 Recurrences

First, we describe a straightforward $O(L^6)$ dynamic programming recurrence for computing the optimal simultaneous alignment and consensus fold for a pair of sequences a and b.

To compute the optimal parse of a and b, we construct 2 four-dimensional matrices, S and D. Here, $S_{i,j;k,l}$ denotes the optimal score for aligning and folding $a_{i+1}a_{i+2}\ldots a_j$ with $b_{k+1}b_{k+2}\ldots b_l$. Furthermore, $D_{i,j;k,l}$ denotes the optimal score for aligning and folding these same substrings, subject to the additional constraint that the outermost positions (a_{i+1},a_j) and (b_{k+1},b_l) form conserved base pairs.

For $0 \le i \le j \le |a|$ and $0 \le k \le l \le |b|$, we have

$$S_{i,j;k,l} = \max \begin{cases} 0 & \text{if } i=j \text{ and } k=l \\ S_{i,j-1;k,l} & \text{if } j>i \\ S_{i,j;k,l-1} & \text{if } l>k \\ S_{i,j-1;k,l-1} + \psi_{\mathbf{w}}^{\text{aligned}}(j,l) & \text{if } j>i \text{ and } l>k \\ \max\limits_{\substack{j':i\le j'\le j-2 \\ l':k\le l'\le l-2}} \left(S_{i,j';k,l'} + D_{j',j;l',l}\right), & \end{cases} \tag{B1}$$

and for $0 \le i < i+2 \le j \le |a|$ and $0 \le k < k+2 \le l \le |b|$,

$$D_{i,j;k,l} = S_{i+1,j-1;k+1,l-1} + \psi_{\mathbf{w}}^{\text{paired}}(i+1,j;k+1,l) + \psi_{\mathbf{w}}^{\text{aligned}}(i+1,k+1) + \psi_{\mathbf{w}}^{\text{aligned}}(j,l). \tag{B2}$$

Here, recurrence (B1) takes the form of a standard Needleman-Wunsch procedure for aligning the substring $a_{i+1}a_{i+2}\ldots a_j$ with $b_{k+1}b_{k+2}\ldots b_l$, with an extra case to handle bifurcations in the base-pairing structure of the RNAs. At the end of the recurrence, $S_{0,|a|;0,|b|}$ gives the score of the optimal alignment and consensus fold of the input sequences a and b. By using traceback pointers in the standard way, the optimal parse can be recovered easily once the recurrence has been evaluated.

In the next section, we explore how these recurrences may be sped up considerably if a constraint set $\mathcal{C}$ of allowed base pairings and aligned positions is known ahead of time. For complexity analysis, we assume $O(c)$ and $O(d)$ bounds on the number of candidate base pairing and alignment partners per sequence position, respectively.

B.1.2 Exploiting base-pairing sparsity

LocARNA (Will *et al.*, 2007) was the first program for simultaneous alignment and folding of RNA to take advantage of base pairing sparsity in a manner that significantly improved in both running time and memory usage. In this section, we recount the innovations of LocARNA as they are applied in RAF. In the next section, we extend these ideas to also account for alignment sparsity.

First, observe that since all parses in $\mathcal{Y}_{\mathcal{C}}$ contain only conserved base pairings, the evaluation of (B2) may be restricted to only

those $D_{i,j;k,l}$ cells for which both $(a_{i+1},a_j)\in\mathcal{C}$ and $(b_{k+1},b_l)\in\mathcal{C}$. Similarly, the inner loop for considering bifurcations in (B1) may also be restricted to only those j' and l' for which both $(a_{j'+1},a_j)\in\mathcal{C}$ and $(b_{l'+1},b_l)\in\mathcal{C}$. Since the bottleneck in the dynamic programming complexity is the number of executions of the innermost loop in (B1), it follows that restricting the considered bifurcations in the manner described above yields an $O(c^2L^4)$ running time; in particular, for each i and k, computing all values of $S_{i,\bullet;k,\bullet}$ takes $O(c^2L^2)$ time as each entry of the D matrix is touched at most once. This optimization was originally implemented as part of the LocARNA (Will *et al.*, 2007) and FoldAlignM (Torarinsson *et al.*, 2007) algorithms.

Second, consider the task of computing all entries in the D matrix. From (B2), we see that the values $D_{i,\bullet;k,\bullet}$ depend only on $S_{i+1,\bullet;k+1,\bullet}$. Similarly, from (B1), the values $S_{i+1,\bullet;k+1,\bullet}$ depend only on $D_{j',j;l',l}$ for $j'\geq i+1$ and $l'\geq k+1$. Thus, ordering computations in the following way allows the recurrences to be evaluated in a single pass:

For $i \leftarrow |a|-2$ downto 0
  For $k \leftarrow |b|-2$ downto 0
    Compute $S_{i+1,\bullet;k+1,\bullet}$
    Compute $D_{i,\bullet;k,\bullet}$

Furthermore, since $S_{i+1,\bullet;k+1,\bullet}$ is only needed while computing $D_{i,\bullet;k,\bullet}$ (but not for any later values of i and k), we need only to retain one $S_{i+1,\bullet;k+1,\bullet}$ matrix in memory at any given time while computing the D matrix. This observation was originally incorporated in the LocARNA program of Will *et al.* (2007).

Finally, observe that once the D matrix has been computed, the score $S_{0,|a|;0,|b|}$ of the optimal parse is easily obtainable in $O(c^2L^2)$ time by recomputing $S_{0,\bullet;0,\bullet}$. Likewise, computing the full traceback requires at most $O(c^2L^3)$ time, negligible relative to the cost of computing the D matrix itself. Thus, we obtain an overall $O(c^2L^4)$ time complexity with $O(c^2L^2)$ space complexity (for storing the D matrix).

B.1.3 Exploiting alignment sparsity

To exploit sparsity in the set of allowed aligned positions in $\mathcal{C}$, we again use the strategy of limiting the DP region. We accomplish this by first considering the simpler problem of computing the reduced DP region $\mathcal{A}$ (known as the *alignment envelope*) for pairwise sequence alignment without folding scores. Using $\mathcal{A}$, we then define a reduced DP region for our original alignment and folding task.

For the first step, consider the following restatement of recurrence (B1) using the notation $\tilde{S}_{j,l}=S_{0,j;0,l}$, where we have omitted the case involving bifurcations/base pairing:

$$\tilde{S}_{j,l}=\max\begin{cases}0 & \text{if } j=0 \text{ and } l=0\\ \tilde{S}_{j-1,l} & \text{if } j>0\\ \tilde{S}_{j,l-1} & \text{if } l>0\\ \tilde{S}_{j-1,l-1}+\psi_{\mathbf{w}}^{\text{aligned}}(j,l) & \text{if } j>0 \text{ and } l>0.\end{cases}$$

As before, $\tilde{S}_{j,l}$ represents the optimal score of aligning $a_1a_2\ldots a_j$ to $b_1b_2\ldots b_l$. Here, our goal is to find $\mathcal{A}$, the minimal set of cells containing no holes,[8] such that for every parse $y\in\mathcal{Y}_{\mathcal{C}}$, there exists some DP path through $\mathcal{A}$ corresponding to an alignment with the same set of aligned positions. Under the assumption that $\mathcal{A}$ contains no holes, we can represent $\mathcal{A}$ by keeping track of its boundaries: for each $j\in\{0,1,\ldots,|a|\}$, let $\langle\mathcal{A}.\textsc{First}[j],\mathcal{A}.\textsc{Last}[j]\rangle$ denote the first and last positions $l\in\{0,1,\ldots,|b|\}$ such that $\tilde{S}_{j,l}\in\mathcal{A}$.

We compute these boundaries in linear time using the following procedure. First, we adjust the boundaries to include $\tilde{S}_{j-1,l-1}\in\mathcal{A}$ and $\tilde{S}_{j,l}\in\mathcal{A}$ for each candidate aligning pair $(a_j,b_l)\in\mathcal{C}$. In addition, we also include the corners $\tilde{S}_{0,0}$ and $\tilde{S}_{|a|,|b|}$ in $\mathcal{A}$. Finally, we force the boundaries of $\mathcal{A}$ to satisfy the monotonicity conditions

$$\mathcal{A}.\textsc{First}[0]\leq\mathcal{A}.\textsc{First}[1]\leq\ldots\leq\mathcal{A}.\textsc{First}[|a|]$$
$$\mathcal{A}.\textsc{Last}[0]\leq\mathcal{A}.\textsc{Last}[1]\leq\ldots\leq\mathcal{A}.\textsc{Last}[|a|]$$

in such a way that guarantees all DP cells $\tilde{S}_{j,l}\in\mathcal{A}$ are accessible via some DP path from $\tilde{S}_{0,0}$ to $\tilde{S}_{|a|,|b|}$.

For the second step, we define the reduced DP region for our original simultaneous alignment and folding recurrences as the set $\mathcal{R}$ of all positions $S_{i,j;k,l}$ such that $\tilde{S}_{i,k}\in\mathcal{A}$ and $\tilde{S}_{j,l}\in\mathcal{A}$. To use this reduced DP region $\mathcal{R}$, then, we simply force $S_{i,j;k,l}=-\infty$ for all $S_{i,j;k,l}\notin\mathcal{R}$. Under this restriction, we can reduce the amount of computation performed in the recurrence (B1) by iterating only over cells $S_{i,j;k,l}\in\mathcal{R}$, and similarly, restricting the evaluation of the D matrix in (B2) to only those cells $D_{i,j;k,l}$ for which $S_{i+1,j-1;k+1,l-1}\in\mathcal{R}$. To ensure that each allowed parse belongs to $\mathcal{Y}_{\mathcal{C}}$, we could penalize any base pairing or aligned position not in $\mathcal{C}$ by $-\infty$. In practice, we instead augment $\mathcal{C}$ to include all aligned matches allowed by $\mathcal{R}$, since this can be done at no increase in computational complexity.

To analyze the new computational complexity of the algorithm, we begin by bounding the size of D matrix in two different ways. First, for each of the $O(cL)$ base pairs $(a_i,a_j)\in\mathcal{C}$, there are $O(d)$ aligning partners for a_i and $O(d)$ aligning partners for a_j, giving a total size of $O(cd^2L)$. Alternatively, for each of the $O(dL)$ aligning pairs $(a_i,b_k)\in\mathcal{C}$, there are $O(c)$ base-pairing partners for a_i and $O(c)$ base-pairing partners for b_k, giving a total size of $O(c^2dL)$. Thus, the size of the D matrix is $O(\min(c,d)\cdot cdL)$.

As in Section B.1.2, the space complexity of the algorithm is dominated by cost of storing the D matrix, and hence, is $O(\min(c,d)\cdot cdL)$. Similarly, the time complexity can be estimated as the number of evaluations of the innermost loop in the bifurcation case of (B1). Since the innermost loop touches each entry of the D matrix at most once for each i and k, and since there are $O(dL)$ choices of $(a_i,b_k)\in\mathcal{A}$, it follows that the time complexity of the algorithm is $O(\min(c,d)\cdot cd^2L^2)$.[9]

C.1 NORM BOUND

In this section, we derive a bound on the maximum norm of the optimal parameter vector $\mathbf{w}^*$ for (4). From standard arguments (see,

[8] That is, $\tilde{S}_{j,l}\in\mathcal{A}$ whenever $\{\tilde{S}_{j_1,l},\tilde{S}_{j_2,l}\}\subseteq\mathcal{A}$ for some $j_1<j<j_2$, or $\{\tilde{S}_{j,l_1},\tilde{S}_{j,l_2}\}\subseteq\mathcal{A}$ for some $l_1<l<l_2$.

[9] Note that in these bounds, we assume an $O(c)$ bound on the number of base-pairing partners per position, and an $O(d)$ bound on the number of aligning partners per position. A weaker condition would be to assume an $O(cL)$ bound on the total number of candidate base-pairing partners for sequences a and b and similarly, an $O(dL)$ bound on the total number of candidate aligned positions; under these conditions, we obtain a worst-case space complexity of $O(\min(c,d)^2L^2)$ and a worst case time complexity of $O(\min(c,d)^2dL^3)$.

e.g. Taskar *et al.*, 2003), the dual optimization problem is

$$\underset{\boldsymbol{\alpha}\in\boldsymbol{\Lambda}}{\text{maximize}} \sum_{i=1}^{m} \sum_{y'\in\mathcal{Y}_C^{(i)}\cup\{y^{(i)}\}} \alpha_{i,y'}\Delta(y^{(i)},y') - \frac{1}{2}C||\mathbf{w}(\boldsymbol{\alpha})||^2$$

where

$$\boldsymbol{\Lambda} = \left\{ (\alpha_{i,y'}) : \alpha_{i,y'} \geq 0, \sum_{y'\in\mathcal{Y}_C^{(i)}\cup\{y^{(i)}\}} \alpha_{i,y'} = \frac{1}{m} \right\}$$

$$\mathbf{w}(\boldsymbol{\alpha}) = \frac{1}{C}\sum_{i=1}^{m} \sum_{y'\in\mathcal{Y}_C^{(i)}\cup\{y^{(i)}\}} \alpha_{i,y'}\Big(\mathbf{F}(x,y^{(i)}) - \mathbf{F}(x,y')\Big).$$

By strong duality, for any solutions $(\mathbf{w}^*,\boldsymbol{\xi}^*)$ and $\boldsymbol{\alpha}^*$ of the primal and dual optimization problems, respectively, the values of the primal and dual objectives must be equal, i.e.,

$$\frac{1}{2}C||\mathbf{w}^*||^2 + \frac{1}{m}\sum_{i=1}^{m}\xi_i^* = \sum_{i=1}^{m} \sum_{y'\in\mathcal{Y}_C^{(i)}\cup\{y^{(i)}\}} \alpha_{i,y'}^*\Delta(y^{(i)},y') - \frac{1}{2}C||\mathbf{w}(\boldsymbol{\alpha}^*)||^2. \tag{C1}$$

Now, suppose that $D_i \in \mathbb{R}$ for $i = 1,\ldots,m$ satisfy

$$D_i \geq \max_{y'\in\mathcal{Y}_C^{(i)}\cup\{y^{(i)}\}} \Delta(y^{(i)},y'). \tag{C2}$$

In the case of the RAF loss function, for example, we can use

$$D_i = (|a|+|b|)\Big(\gamma^{\text{FP paired}} + \gamma^{\text{FN paired}} + \gamma^{\text{FP aligned}} + \gamma^{\text{FN aligned}}\Big).$$

Then the KKT optimality condition $\mathbf{w}^* = \mathbf{w}(\boldsymbol{\alpha}^*)$, the primal constraint that $\xi_i^* \geq 0$ for $i = 1,\ldots,m$, and (C1) imply that

$$\begin{aligned} C||\mathbf{w}^*||^2 &= \sum_{i=1}^{m} \sum_{y'\in\mathcal{Y}_C^{(i)}\cup\{y^{(i)}\}} \alpha_{i,y'}^*\Delta(y^{(i)},y') - \frac{1}{m}\sum_{i=1}^{m}\xi_i \\ &\leq \sum_{i=1}^{m} \sum_{y'\in\mathcal{Y}_C^{(i)}\cup\{y^{(i)}\}} \alpha_{i,y'}^* D_i = \frac{1}{m}\sum_{i=1}^{m} D_i. \end{aligned}$$

Therefore, $||\mathbf{w}^*|| \leq \sqrt{\frac{1}{m}\sum_{i=1}^{m} D_i/C}$. ■