RNA structure prediction by dynamic programming

Group 35 Shenxin Jiang, Grace Lang, Simin Liu

Abstract

Ribonucleic acid (RNA) is a polymeric molecule made up of four bases: A, U, G, and C. Pairing of these bases fold an RNA molecule into a functional secondary structure. Decoding the secondary structure is essential for understanding the biological function of an RNA molecule. In this paper, we compare the performance of three implementations of two classical dynamic programming algorithms (Nussinov algorithm and Zuker algorithm) on predicting structures of different length RNA and discuss the differences among three widely used metrics for measuring similarity between RNA secondary structures.

1 Problem background

Ribonucleic acid (RNA) is a linear polymeric molecule majorly made up of nucleotides containing four types of bases: adenine, uracil, guanine, and cytosine (denoted by the letters A, U, G, and C). Although RNA is a single-strand molecule, it can fold into a structure called the secondary structure following the rule of complementary base pairing (A with U, and G with C)[1]. Figure 1 shows an example of a linear RNA molecule and the secondary structure it forms:

Human tRNA-SeC

GCCCGGAUGAUCCUCAGUGGUCUG
GGGUGCAGGCUUCAAACCUGUAG
CUGUCUAGCGACAGAGUGGUUCA
AUUCCACCUUUCGGGCG

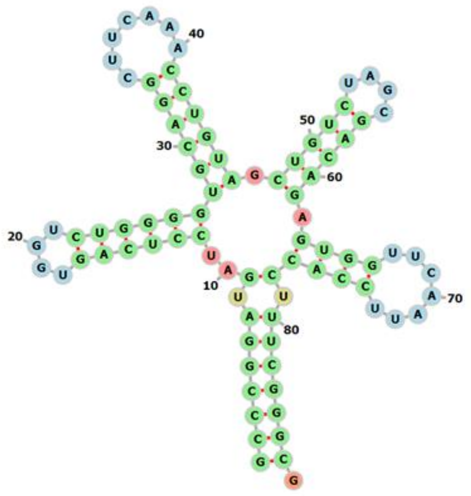

Figure 1: Example RNA sequence and structure (structure diagram made by Forna[2])

By forming secondary structures, RNAs play important roles in many physiological processes, including protein synthesis, gene expression regulation, and catalyzing biological reactions. Therefore, decoding the secondary structure is essential for understanding the biological function of an RNA molecule.

In addition to experimental approaches, scientists have developed a series of computational methods to predict the structure of an RNA molecule from its sequence. In this project, we aim to implement two classical dynamic programming algorithms, the Nussinov algorithm[3], [4] and the Zuker algorithm[5], to predict RNA structures. We compare the performance of three implementations of the two algorithms in terms of how accurate the predictions are to the ground truth and how fast the algorithms run.

2 Algorithms and Implementations

2.1 Nussinov algorithm (our implementation)

Pioneered by Ruth Nussinov and her colleagues, the Nussinov algorithm aims to find the secondary structure that contains the most base pairs[3], [4]. The algorithm is based on dynamic programming. To calculate the optimal structure that has the maximum number of base pairs, the algorithm calculates the max-base-pair structure of every sub-sequence (from short to long), until getting the optimal structure of the whole molecule. We implemented this algorithm from scratch on our own.

To illustrate the algorithm, consider the following case:
Let R be an RNA sequence with length L. Let A be the length of the subsequence S_{ij}, and let R_i, R_j be the left and right end of this subsequence (j = A + i). The 2D matrix DP is built to store the result and reduce redundant calculations, and DP[i][j] is the maximum number of base pairs in the RNA subsequence S_{ij}.

To calculate DP[i][j], we consider two cases:

Case one:
For i < k < j, if there is any base R_k in S_{ij} that R_j and R_k can form a base pair $P_{k,j}$ (Figure 2), then this base pair divide the subsequence S_{ij} into two smaller subsequences, $S_{i,k-1}$ and $S_{k+1,j-1}$. Therefore, the maximum number of base pairs of S_{ij} should be the sum of the maximum numbers of base pairs of $S_{i,k-1}$ and $S_{k+1,j-1}$, plus 1 ($P_{k,j}$):

$$DP[i][j] = DP[i, k - 1] + DP[k + 1, j - 1] + 1 \tag{1}$$

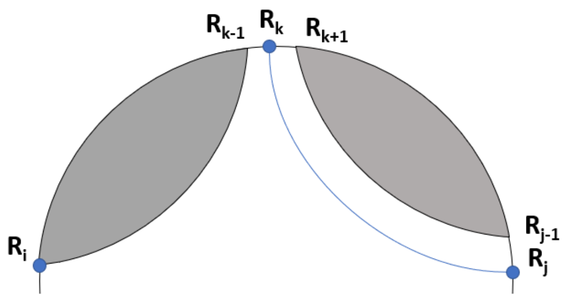

Figure 2: Nussinov algorithm, case one

If there are multiple R_k's that can pair with R_j, we calculate each of them and take the one that gives the largest number of base pairs as the optimal solution.
(Since we calculate subsequences from short to long, DP[i, k - 1] and DP[k + 1, j - 1] should be known.)

Case two:
If R_j does not pair with any other bases in the subsequence S_{ij} (Figure 3), then:

$$DP[i][j] = DP[i][j - 1] \tag{2}$$

(Note that in this case it does not necessarily mean that R_j CAN NOT pair with any other bases in the subsequence S_{ij} - it is always possible that some R_k can pair with R_j. Here we are discussing the case that an optimal structure prefers R_j not to pair with any R_k in the subsequence.)

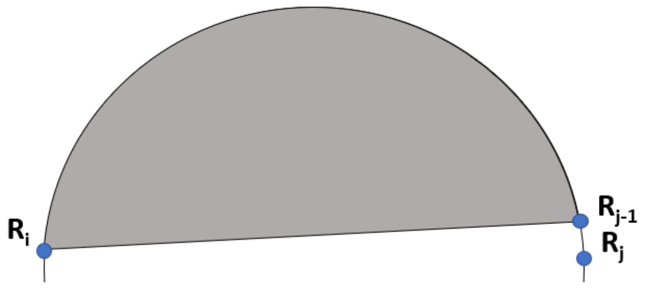

Figure 3: Nussinov algorithm, case two

At the end for the subsequence S_{ij}, we compare the two cases and take the one that has more base pairs:

$$DP[i][j] = \ max[case\ one, case\ two] \tag{3}$$

By storing the index k of R_k that pairs with R_j (0 if R_j does not pair with any base) in a new matrix K (by setting K[i][j] = k or 0), we can easily store and retrieve the base pairs in the optimal structure. Since $i < j$, the part of matrix DP below the main diagonal always remains empty. Therefore, we can let DP[j][i] = K[i][j] to save space.

2.2 Zuker algorithm (our implementation)

To get a more biologically relevant structure, it is important to consider topological and thermodynamic rules to find a structure that is energetically most favorable. Based on previous algorithms (including the Nussinov algorithm) and published free energy data of RNA[6], [7], Michael Zuker and colleagues developed the Zuker algorithm[5], aiming to find the RNA structure that has the minimum free energy. This algorithm is also a dynamic programming algorithm and can calculate the biologically reasonable structure. We implemented this algorithm from scratch on our own.

To illustrate the algorithm, we still consider an RNA sequence R with length L, and its subsequence S_{ij}, where $i < j$. Let R_i, R_j be the left and right end of this subsequence. We use two 2D matrices V and W for memorizing the energy of subsequences. W[i][j] records the energy of substructure S_{ij}, where V[i][j] records the energy of substructure S_{ij} when R_i and R_j form a base pair $P_{i,j}$. By this definition, the minimum free energy of the whole RNA will be stored in W[1][L].

We first consider the case where R_i and R_j pair with each other (Figure 4). To calculate V[i][j], we consider three different cases about the structure between R_i and R_j:

Case one (Figure 4, left):
Hairpin Loop: There are no other base pairs between R_i and R_j.

Case two (Figure 4, middle):
Let $P_{ii,jj}$ formed by R_{ii} and R_{jj} the closest base-pair to $P_{i,j}$. $P_{i,j}$ and $P_{ii,jj}$ can form three types of structures:
Stacking region: (if $ii = i + 1$ **and** $jj = j - 1$)

$$V[i][j] = Stack\ Energy(Pi, j, Pii, jj) + V[ii, jj] \tag{4}$$

Bulge loop: (if ii = i + 1 **xor** jj = j - 1)

$$V[i][j] = Bulge\ Energy(Pi, j, Pii, jj) + V[ii, jj] \tag{5}$$

Interior loop: (if ii != i + 1 **and** jj != j - 1)

$$V[i][j] = Interior\ Loop\ Energy(Ri, Rii) + Interior\ Loop\ Energy(Rj, Rjj) + V[ii, jj] \tag{6}$$

Here the Stack Energy, Bulge Energy, and Interior Loop Energy are calculated based on experimental data.

Case three (Figure 4, right):
Bifurcation Loop: As shown by the diagram, bifurcation Loop could be regarded as the composition of two substructures, namely $S_{i+1,\,k}$ and $S_{k+1,j-1}$, where R_k separates the two subsequences. We test every k that satisfies i +1 < k < j - 2, and let

$$V[i][j] = W[i + 1, k] + W[k + 1, j - 1] \tag{7}$$

have the minimum value.

At the end for the subsequence S_{ij}, we compare the two cases and take the smallest one:

$$V[i][j] = min[case1, case2, case3] \tag{8}$$

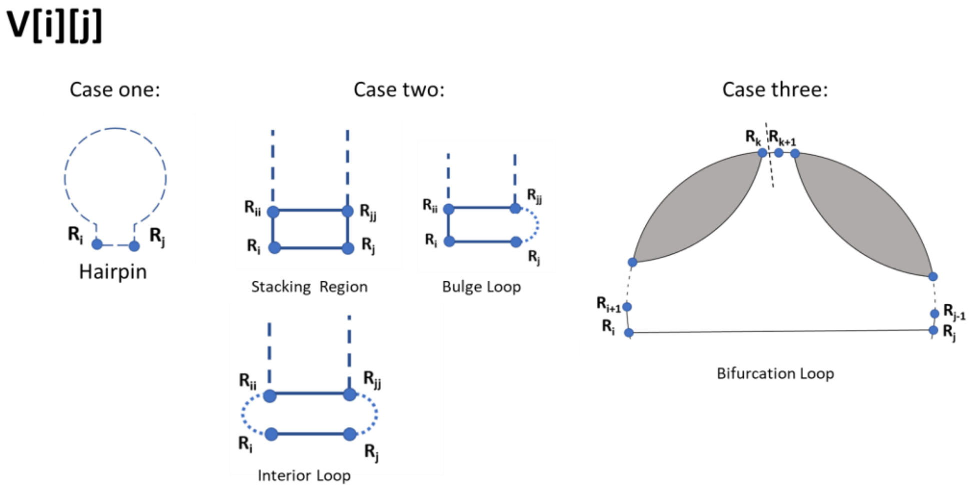

Figure 4: Zuker algorithm, V[i][j]

Then we consider the more general case where R_i and R_j do not have to pair with each other. To calculate W[i][j], we consider three different cases about the base pairing status of R_i and R_j:

Case one (Figure 5, left):
If none or only one of R_i and R_j pairs with another base, then

$$W[i][j] = min(W[i][j - 1], W[i + 1][j]) \tag{9}$$

Case two (Figure 5, middle):
If R_i and R_j form a base pair, then

$$W[i][j] = V[i][j] \tag{10}$$

This indicates that we need to calculate V[i][j] first.

Case three (Figure 5, right):

If R_i and R_j both form base pairs but not with each other, we then find an interior base R_k to separate the sequence into two subsequences. In this case,

$$W[i][j] = W[i][k] + W[k+1][j] \tag{11}$$

We scan through every k to find the one that gives the minimum energy.

At the end for the subsequence S_{ij}, we compare the three cases and take the smallest one as shown in equation 12.

$$W[i][j] = min[case1, case2, case3] \tag{12}$$

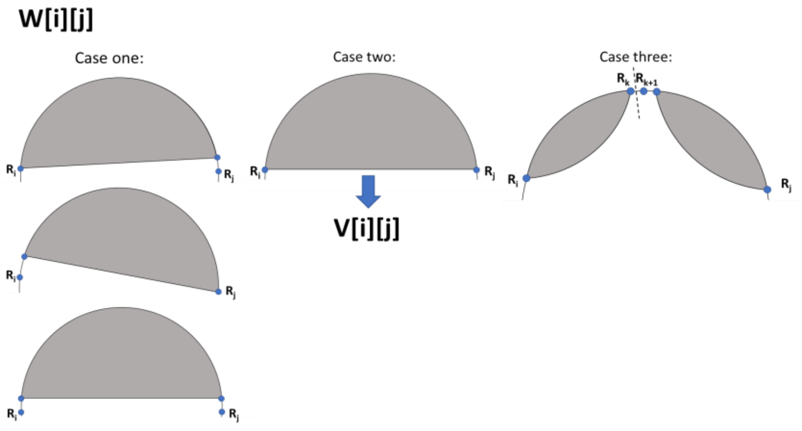

Figure 5: Zuker algorithm, W[i][j]

In this way, the minimum free energy of the whole RNA will be stored in W[1][L] by the definition of matrix W, and the structure can also be retrieved by storing the choices among cases and base pairs during the whole process.

2.3 Zuker algorithm (the implementation from Lattice Automation, "seqfold")

In addition to implementing the Zuker algorithm by ourselves, we also looked for open-source implementations, and the seqfold package from Lattice Automation is one of them (https://github.com/Lattice-Automation/seqfold)[4], [5], [8]–[10]. Seqfold is an implementation of the same Zuker algorithm but is more delicate and implements a more comprehensive and detailed set of free energy data to help calculate the free energy of substructures (Table 1).

Table 1. Energy types included in our Zuker implementation and in Seqfold

Energy type	Our Zuker implementation	Seqfold
Hairpin loop energy	Energy data are available for loops <= 10 nt; for loops > 10 nt, extrapolation	Energy data are available for loops <= 30 nt; for loops > 30

Energy type	Our Zuker implementation	Seqfold
bulge loop energy	is needed.	nt, extrapolation is needed.
Interior loop energy		
Stacking base pair energy	Only AU and GC pairs are considered	GU pairs are included
	All stacking base pair energy are calculated in the same way	Interior and terminal stacking base pair energy are calculated separately
Multibranch energy	Set as 0	Energy data provided

In addition, seqfold introduces the concept of "energy penalty", proposed by SantaLucia and colleagues[8], to correct the calculation error when the substructure is AT closing. A function used to adjust the free energy based on a pre-set value temperature is also included to increase the algorithm's versatility. Seqfold also separates the pre-calculations from their recursive functions. They regarded the Stack and Bulge structure as a special situation of internal loops, which increase the reusability of the code.

A major difference between our implementation and the seqfold implementation is that seqfold calculates W[1][L] in a recursive way with memorization of intermediate results, thus only the subsequences that are relevant will be calculated in the recursive steps, while our implementation calculates every single subsequence from short to long, which may include many subsequences that are not useful for the result. Therefore, it is not surprising that the time it takes to execute seqfold is faster than our implementation of the zuker algorithm. We will discuss to what extent the runtimes of the two implementations differ in the experiment section of this paper.

3 Dataset

We compiled a set of 419 RNAs from the RNAcentral database (https://rnacentral.org/)[11]. Based on previous studies, dynamic programming based secondary structure prediction algorithms are more accurate for short RNAs than for long RNAs[4], [5]. To test the performance of these algorithms on RNAs with different lengths, our dataset covers the length from 17 bp to 410 bp with about 20 sequences in each 20 bp bin (Figure 6).

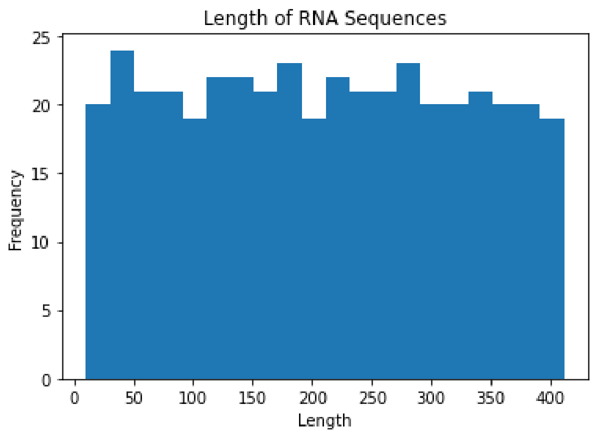

Figure 6: Histogram of full dataset

For each row of data, we collect the RNA sequence, length, and its "ground truth" secondary structure, denoted by a dot-bracket sequence. With this notation, each dot denotes an unpaired base, and each pair of left and right brackets denote a base pair. Table 2 is one sample of data from the set we compiled.

Table 2. Example sample from dataset

Name:	Human tRNA-SeC
Sequence:	GCCCGGAUGAUCCUCAGUGGUCUGGGGUGCAGGCUUCAAACCUGUAGC UGUCUAGCGACAGAGUGGUUCAAUUCCACCUUUCGGGCG
Secondary Structure:	(((((((.(..((((((....))))))((((((.......)))))).(((((....))))).((((.......))))).))))))).

The RNAcentral database is an integrated database of a comprehensive and up-to-date collection of non-coding RNA information. Secondary structures available on this database are from multiple sources, including biochemical analysis, structural biological studies, and bioinformatic analysis such as phylogenetic analysis of conserved structures across different species[11]. Therefore, we think it is reasonable to take secondary structures from this database as a biologically relevant ground truth to test the performance of our algorithms.

4 Metrics

After running the whole dataset with the three implementations (our Nussinov implementation, our Zuker implementation, and seqfold) and getting predicted structures, we compared these predicted structures with ground truth structures, and used three different metrics (BP, RBP, and tree alignment distance) to judge how similar the predicted structures are to the ground truth.

4.1 Base-pair score (BP)

The base-pair (BP) score between two secondary structures is the total number of base pairs that exist in one structure but not in the other[12]. For example, in the following two structures, only the bold base pair exists in both structures, while the blue and red base pairs are unique to one of the two structures. In this case, BP score = 2 (blue) + 2 (red) = 4.

…((....))..()

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

The BP score is straightforward but also too stringent, thus it has obvious drawbacks - it essentially does not consider the shape of the structures. Two topologically very similar structures can be formed by a set of very different base pairs, while two structures that share a decent amount of base pairs can look completely different.

4.2 Relaxed base-pair score (RBP)

To overcome the drawbacks of the BP score, the relaxed base-pair (RBP) score is introduced[12]. Instead of finding base pairs that are the same, the RBP metric provides a flexible range around one base pair for it to "move around" and "find" another base pair in the other structure. The flexible range, namely how far a base pair needs to travel until finding another base pair, becomes the RBP score. Two identical structures have an RBP = 0, and the bigger the RBP score is, the more different the two structures are. Under the RBP metric, the same example structures shown in the BP section has an RBP = 1, which is very close to 0 and reflects their structural similarity well. Figure 7 below illustrates how an RBP score of 1 is determined:

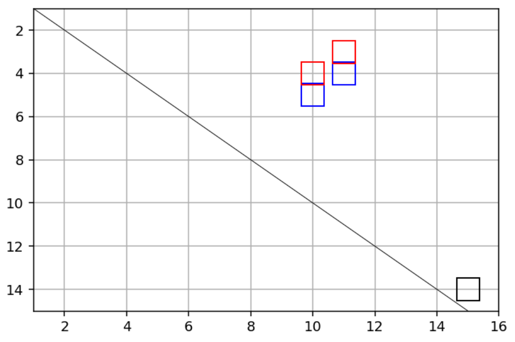

Figure 7: RBP score via structure dot plot

Starting with an index of 1, the first structure has pairs at positions [(4,11), (5,10), (14,15)] and the second structure has pairs at [(3,11), (4,10), (14,15)]. A structure dot plot is created that corresponds to these pairs. For example, on the plot above, the pair (4,11) is represented by the rightmost blue square. Instead of just plotting a single point to represent each pair, we instead start with a 1x1 box. The requirements for the RBP score are that every red or blue box touches a box of a different color in order to be considered close. We keep expanding the box size and excluding pairs until this condition is met. For this simple example, we only need a 1x1 box to satisfy the requirement, so the RBP score is 1.

4.3 Tree alignment distance (TD)

RNA structures can be represented as trees, where internal nodes represent base pairs, and leaves represent unpaired bases[13], [14]. By converting a dot-bracket sequence into a tree-node structure, we consider not only the difference between number and 1D position of base pairs, but also the 2D topology of the whole structure[15], [16]. Specifically, for two RNA tree structures T_1 and T_2 with root nodes N_1 and N_2, let T_{11} to T_{1N} be T_1's subtree structure with parent node N_1, and T_{21} to T_{2M} be T_2's subtree structure with parent node N_2.

Assuming that N_1 has more child nodes than N_2 ($N > M$), The tree alignment distance Distance(T_1, T_2) should be:

$$Distance(N1, N2) + Distance(T1a, T2a)\ for\ a\ in\ [1, M] + Distance(T1b, empty)\ for\ b\ in\ [N - M, N] \quad (13)$$

We could recursively calculate the distance of each subtree structure and at the end get the distance between the whole trees. In this way, even if two RNA structures share a decent amount of base pairs, their tree alignment distance will be huge if they do not share similar 2D shapes and topology.

5 Experiments and Results

5.1 Accuracy Analysis

We were able to obtain predicted structures for the full dataset using all algorithms and implementations discussed above and generate 3 scores (BP, RBP, and Tree Alignment Distance) for each prediction. Below (Figure 8, 9) are examples of the predicted structures compared to their ground truth for a long and short RNA sequence from the dataset.

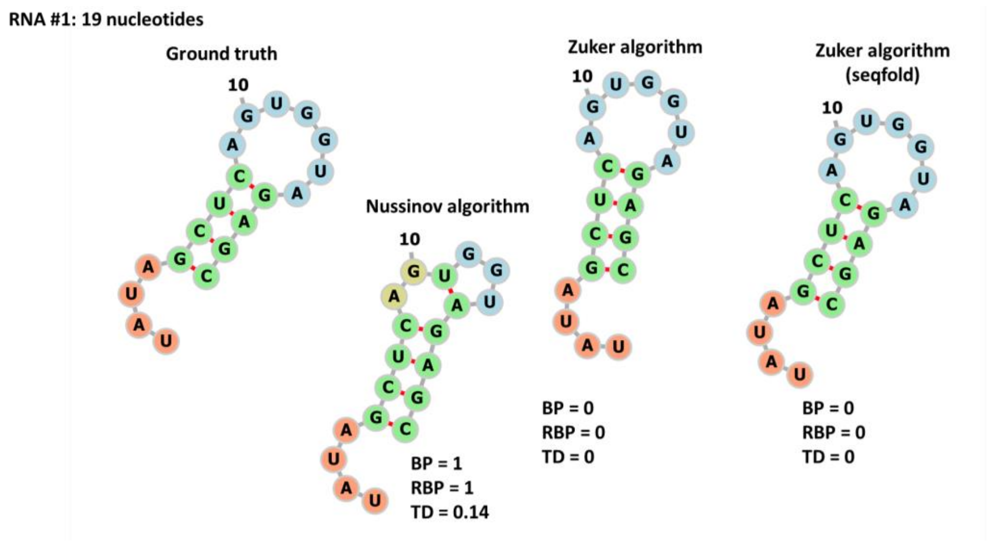

Figure 8: Predicted Structures and Scores of Short RNA

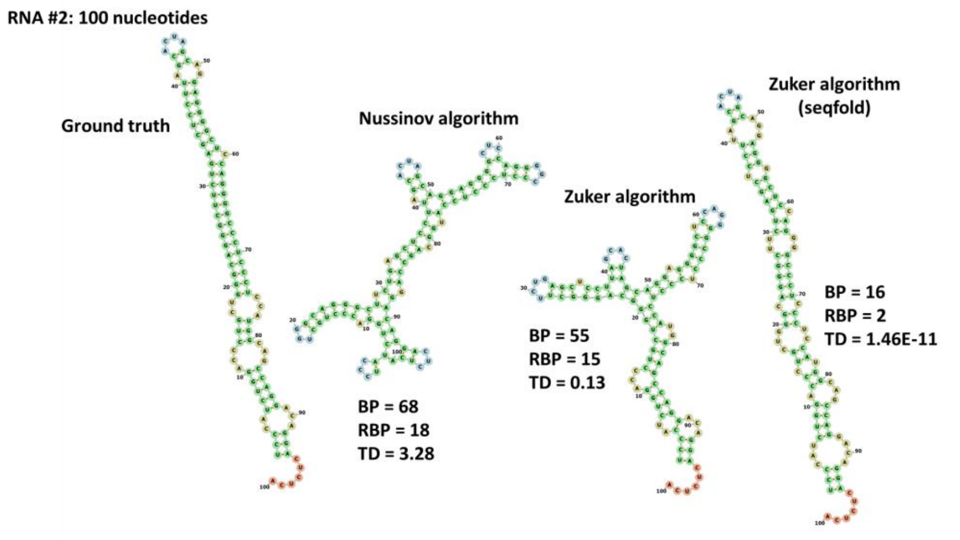

Figure 9: Predicted structures and scores of long RNA

For these two examples, it is clear the scores and distances go up as the structure gets longer. Keeping this in mind, we can analyze just how close the predicted structures are to the ground truth as RNA sequence length changes. To do this, we separated the data into 20 bins depending on length: 10-30, 31-50, 51-70, 71-90, 91-110, 111-130, 131-150, 151-170, 171-190, 191-210, 211-230, 231-250, 251-270, 271-290, 291-310, 311-330, 331-350, 351-370, 371-390, and 391-410. Figure 10 shows how the algorithms performed under the BP metric:

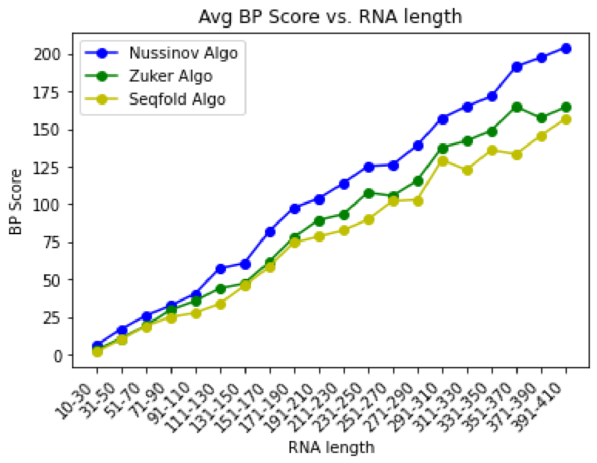

Figure 10: Average BP scores for different length RNA

Figure 10 shows the visual impact of RNA length on prediction accuracy. Additionally, under the strictness of this metric, the Nussinov algorithm performs the worst and the seqfold implementation of the Zuker algorithm performs the best. The Nussinov algorithm performing the worst may have to do with the number of base pairs predicted. By design, the Nussinov algorithm maximizes the number of base pairs. Figure 11 below shows the average number of base pairs for each sequence length bin for the three algorithms. As expected, Nussinov has the largest number of pairs for each bin. Since the BP score counts the number of pairs that do not exist in the other structure to determine the score, the Nussinov algorithm may have the highest score because it has more pairs to be counted.

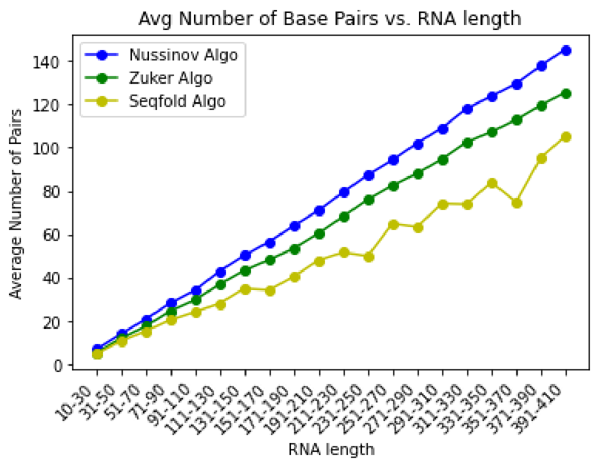

Figure 11: Average number of predicted base pairs for different length RNA

Figure 12 shows how the algorithms performed when looking at the RBP score. Implementing a relaxed version of the scoring results in less differentiation between the three algorithms. They all perform similarly throughout the entire range of RNA lengths. We can also see just how much the RBP implementation brings down the score when compared to the average BP score. The BP score ranges from 0-200, while the RBP ranges from 0-50. The less strict measure of similarity between the predicted structures and the ground truth resulted in much lower scores overall.

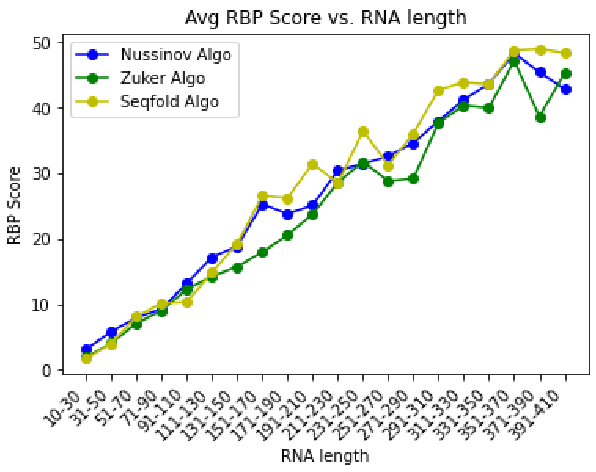

Figure 12: Average RBP scores for different length RNA

Figure 13 shows the average tree alignment distance for each algorithm. Compared with the previous two metrics, the tree alignment distance emphasizes the similarity of the 2D shapes and topology of two RNA secondary structures, and in the real world, the shape and topology of an RNA secondary structure (i.e., our ground truth) must obey the thermodynamic rules. Therefore, algorithms that consider as many different types of thermodynamic free energy as possible are more likely to get a biologically and thermodynamically reasonable predicted structure, and such a predicted structure is more likely to be similar to the ground truth, thus getting a better score for the tree alignment distance.

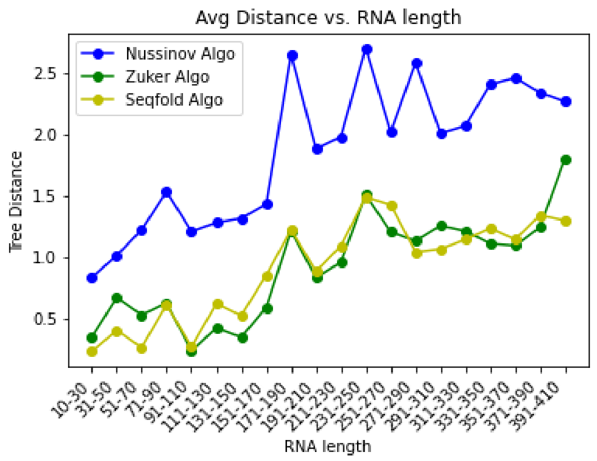

Figure 13: Average tree alignment distances for different length RNA

As expected, implementations of the Zuker Algorithm are relatively similar along the entire range of sequence lengths. Since the Nussinov algorithm does not consider any free energy, it is noticeably worse for every single range, particularly for the 171-190, 231-250, and 271-290 range. Figure 14 takes a closer look at the 171-190 range.

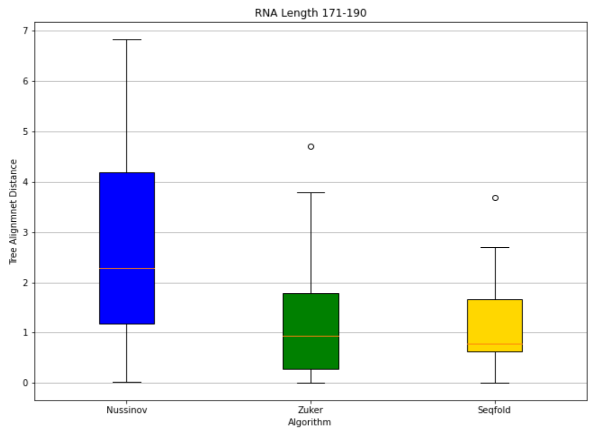

Figure 14: Tree Alignment Distances of 171-190 nt length RNA

The Nussinov algorithm has tree alignment distances over a much larger range than the other two algorithms. Although the two Zuker implementations have similar means, seqfold has less variance and may be more reliable in predicting the structure of other RNA sequences outside this dataset.

5.2 Runtime Analysis

A subset of the 419 RNA was used to determine the average runtime of the three algorithms/implementations. 5 sequences were taken from the 20 bins of different sequence lengths from 10-410 nucleotides for a total of 100 data points. Each algorithm was executed 5 times to determine the average runtime. It is important to note that all three algorithms are written in Python. We can deduce a true comparison between our two implementations because they were developed by the same people, in the same language at the same time. The runtime of our Zuker implementation can also be directly compared to the seqfold implementation to determine if there is room to improve the efficiency of our implementation. Table 3 shows these results.

Table 3. Runtime Results

Algorithm (Implementation)	Average Runtime (minutes)	Standard Deviation
Nussinov (Ours)	2.32	0.03
Zuker (Ours)	238.05	1.31
Zuker (seqfold)	65.60	1.30

There is a large discrepancy between the runtime for all three algorithms. Our implementation of the Nussinov algorithm is extremely fast, taking on average only 2 and a half minutes to calculate predicted secondary structures for the 100 RNA sequences. On the other hand, our implementation of the Zuker algorithm took almost four hours. The seqfold implementation of the Zuker Algorithm took a little over an hour. This shows it is possible to reduce the runtime of our Zuker Algorithm to make running it on large datasets more feasible by removing unnecessary calculations that are part of the recursion. However, it is unlikely that we could reduce it enough to have a similar runtime to the Nussinov algorithm.

6 Code Implemented

As discussed previously, one Zuker implementation and the Nussinov implementation were written from scratch by us. The open source seqfold library was integrated into our existing framework to transform the results into the correct format to allow us to measure accuracy via scoring metrics. The BP and RBP metrics were implemented from scratch. RNAlib-2.5.0 library was imported to convert dot bracket RNA sequence into "tree string", which was then converted into calculable tree structure. The alignment distance calculation algorithm was implemented with Java and Jscheme. Data analysis of the results was implemented in Python with the help of the numpy, pandas, and matplotlib python libraries in order to generate the figures throughout this paper. Most of our code are available on github. (https://github.com/gracelang15/Algorithms-Project)

7 Discussion

Based on the results above, the two algorithms and their implementations have different characteristics that make choosing one or the other to predict secondary structure situationally dependent. Zuker is more accurate than Nussinov in terms of the tree alignment distance and the BP metric. But, when taking a less strict approach to scoring with the RBP metric, the results are almost indistinguishable between the algorithms. Runtime is also an important characteristic of the algorithms/implementations. Nussinov is more than 28 times faster than the seqfold Zuker implementation and more than 100 times fast than our implementation.

Based on these results, it is advised to choose the algorithm on a case-by-case basis. Generally speaking, if structure predictions are required for long RNA sequences and/or a large dataset while accuracy is not as important, it is recommended to use the Nussinov algorithm to trade off accuracy with runtime performance. On the other hand, if greater accuracy is the most important factor, or if a biologically reasonable structure is required, then the seqfold implementation of the Zuker algorithm should be used since it incorporates biologically relevant information, runs in a quarter of the time, has the same tree alignment distance and RBP score as our implementation, and a better overall BP score.

RNA secondary structure prediction has long been a very critical yet not fully solved problem in biology. The gold-standard method for determining the structure of an RNA are structural biological approaches, including X-ray crystallography, nuclear magnetic resonance (NMR), and cryogenic electron microscopy[17]. While by these methods scientists can obtain the precise structure of an RNA, these methods are time and labor consuming and can be very difficult to perform, and currently there are only <250 non-redundant, high-resolution RNA structures available which covers only a tip of the iceberg. Computationally, in addition to using algorithms with thermodynamic, statistical, and probabilistic data to predict structure, comparative sequencing analysis is also used to predict the structure of an RNA based on its homologous RNA sequences whose structures are available. The cutting-edge machine learning and deep learning methods make the prediction of RNA secondary structure even easier and more accurate, which also allow improvement in RNA 3D modeling. It is foreseeable that the rapid development of both experimental and computational methods will provide a deeper understanding on RNA structure and function[17], [18].

8 References

[1] D. L. Nelson and M. M. Cox, *Lehninger principles of biochemistry*.

[2] P. Kerpedjiev, S. Hammer, and I. L. Hofacker, “Forna (force-directed RNA): Simple and effective online RNA secondary structure diagrams,” *Bioinformatics*, vol. 31, no. 20, pp. 3377–3379, Oct. 2015, doi: 10.1093/bioinformatics/btv372.

[3] R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman, “Algorithms for Loop Matchings,” *SIAM J. Appl. Math.*, vol. 35, no. 1, pp. 68–82, Jul. 1978, doi: 10.1137/0135006.

[4] R. Nussinov and A. B. Jacobson, “Fast algorithm for predicting the secondary structure of single-stranded RNA,” *Proc. Natl. Acad. Sci.*, vol. 77, no. 11, pp. 6309–6313, Nov. 1980, doi: 10.1073/pnas.77.11.6309.

[5] M. Zuker and P. Stiegler, “Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information,” *Nucleic Acids Res.*, vol. 9, no. 1, pp. 133–148, Jan. 1981, doi: 10.1093/nar/9.1.133.

[6] W. Salser, “Globin mRNA Sequences: Analysis of Base Pairing and Evolutionary Implications,” *Cold Spring Harb. Symp. Quant. Biol.*, vol. 42, pp. 985–1002, Jan. 1978, doi: 10.1101/SQB.1978.042.01.099.

[7] H. Jacobson and W. H. Stockmayer, “Intramolecular Reaction in Polycondensations. I. The Theory of Linear Systems,” *J. Chem. Phys.*, vol. 18, no. 12, p. 1600, Dec. 2004, doi: 10.1063/1.1747547.

[8] J. SantaLucia and D. Hicks, “The Thermodynamics of DNA Structural Motifs,” *Annu. Rev. Biophys. Biomol. Struct.*, vol. 33, no. 1, pp. 415–440, 2004, doi: 10.1146/annurev.biophys.32.110601.141800.

[9] D. H. Turner and D. H. Mathews, “NNDB: the nearest neighbor parameter database for predicting stability of nucleic acid secondary structure,” *Nucleic Acids Res.*, vol. 38, no. suppl_1, pp. D280–D282, Jan. 2010, doi: 10.1093/nar/gkp892.

[10] M. Ward, A. Datta, M. Wise, and D. H. Mathews, “Advanced multi-loop algorithms for RNA secondary structure prediction reveal that the simplest model is best,” *Nucleic Acids Res.*, vol. 45, no. 14, pp. 8541–8550, Aug. 2017, doi: 10.1093/nar/gkx512.

[11] RNAcentral Consortium, “RNAcentral 2021: secondary structure integration, improved sequence search and new member databases,” *Nucleic Acids Res.*, vol. 49, no. D1, pp. D212–D220, Jan. 2021, doi: 10.1093/nar/gkaa921.

[12] P. Agius, K. P. Bennett, and M. Zuker, “Comparing RNA secondary structures using a relaxed base-pair score,” *RNA*, vol. 16, no. 5, pp. 865–878, May 2010, doi: 10.1261/rna.903510.

[13] S. Schirmer, Y. Ponty, and R. Giegerich, “Introduction to RNA Secondary Structure Comparison,” in *RNA Sequence, Structure, and Function: Computational and Bioinformatic Methods*, J. Gorodkin and W. L. Ruzzo, Eds. Totowa, NJ: Humana Press, 2014, pp. 247–273. doi: 10.1007/978-1-62703-709-9_12.

[14] R. Mikhaiel, G. Lin, and E. Stroulia, “Simplicity in RNA Secondary Structure Alignment: Towards biologically plausible alignments,” in *Sixth IEEE Symposium on BioInformatics and BioEngineering (BIBE’06)*, Oct. 2006, pp. 149–158. doi: 10.1109/BIBE.2006.253328.

[15] T. Jiang, L. Wang, and K. Zhang, “Alignment of trees — an alternative to tree edit,” *Theor. Comput. Sci.*, vol. 143, no. 1, pp. 137–148, Jul. 1995, doi: 10.1016/0304-3975(95)80029-9.

[16] M. Pawlik and N. Augsten, “Compare your trees with ease.” http://tree-edit-distance.dbresearch.uni-salzburg.at/

[17] J. Singh, J. Hanson, K. Paliwal, and Y. Zhou, “RNA secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning,” *Nat. Commun.*, vol. 10, no. 1, p. 5407, Nov. 2019, doi: 10.1038/s41467-019-13395-9.

[18] J. Fallmann, S. Will, J. Engelhardt, B. Grüning, R. Backofen, and P. F. Stadler, “Recent advances in RNA folding,” *J. Biotechnol.*, vol. 261, pp. 97–104, Nov. 2017, doi: 10.1016/j.jbiotec.2017.07.007.