6.047 Computational Biology Final Report

Improving The Runtime Of Nussinov's Algorithm By Partitioning

Nicole Power and Lily Seropian

December 9, 2013

Abstract

RNA is a significant molecule in biology which is not yet fully understood. Many RNA molecules are noncoding RNAs, which are not translated into proteins, but have a distinct form and function of their own, akin to proteins. RNA has a complex three-dimensional geometry which is closely tied to its function; thus, scientists have been developing predictive RNA folding algorithms to increase our understanding. Currently, there are a few very complex algorithms to predict tertiary folding, and many others which focus on secondary structure. The secondary structure RNA folding algorithms have to-date all shared an expected runtime of $O(n^3)$ where n is the length of the RNA sequence. We've sought to improve upon the existing RNA folding algorithm of Nussinov using partitioning on the RNA sequence. Our partitioning function uses an experimentally determined partition length to provide an optimal runtime and folding score. The RNA is then partitioned into subsequences of the optimal partition size, and run through Nussinov's algorithm as independent sequences. The expected runtime of our algorithm is $O(n^2)$. Our Nussinov's algorithm with partitioning did not achieve an effective speed-up on unpartitioned Nussinov's, but for some partition lengths, the folding score was an improvement. These results inform us that partitioning an RNA sequence may not be an appropriate method for decreasing the runtime of RNA folding, due to the effect of large constants on the effective runtime. Had our partitioning function improved the runtime effectively, then it would have been a novel result, beneficial to the biological research community in allowing a faster analysis of RNA structure. However, it is possible that we have enabled Nussinov's algorithm to account for pseudoknots, which bears further investigation.

Background

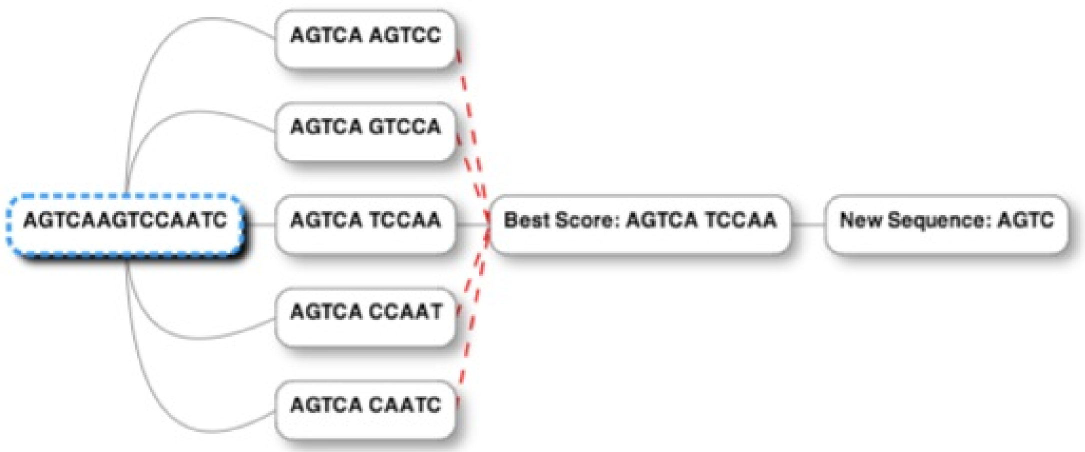
RNA is a very biologically important molecule. It is the intermediate between DNA and proteins, thus vital to gene expression. It also plays catalytic roles in the cell, most notably in ribosomes. The many different types of RNA—mRNA, tRNA, rRNA, piRNA, microRNA, and others—have diverse functions as a result of their diverse forms. Unlike DNA, RNA is capable of forming single-stranded double helices, a result of bonding with itself. RNA function would be better understood if we knew its form, but the tertiary structure is complex to predict. Secondary structure prediction, however, is within the grasp of our current tools, and can greatly expand our understanding of RNA. From properties of the secondary structure, functional attributes of the specific RNA molecule can be gleaned. As with protein function, RNA function follows form, so looking at RNA's form helps researchers understand how specific RNA molecules bind and interact with other molecules. RNA secondary structure is also useful to study because it is evolutionarily well preserved, allowing us to better find and understand non-coding RNAs.

Existing algorithms that have sought to predict RNA secondary structure have been based either on thermodynamics or probabilities. Nussinov's Algorithm uses Dynamic Programming to recursively calculate the substructure of subsequences of the RNA that minimize free energy. The Zuker Algorithm is also structured to minimize free energy, but it also factors in the contribution of stacking energy. The existing algorithms which approach the RNA folding problem probabilistically are McCaskill, CYK, and Inside-Outside Algorithms. The McCaskill Algorithm uses a partition function and pair probabilities, while the CYK and Inside-Outside Algorithms use Stochastic Context-Free Grammars. These algorithms all run in $O(n^3)$ time. Our work improves upon Nussinov's algorithm for the RNA folding problem by using partitioning to reduce its runtime from $O(n^3)$ to $O(n^2)$.

Our algorithm differs from previous work in that it divides the RNA into smaller pieces and assumes independence between pieces. This assumption is necessary for the decrease in expected runtime from partitioning. Unlike Nussinov, we allow for pseudo-knots to form, based on this sequence-independence. In the future we hope to score these pseudo-knots against known pseudo-knots to determine correctness and accuracy (see **Future Goals** for further discussion).

Results And Discussion

Method: We first developed a partitioning algorithm to be used to optimally break the RNA down into constant-sized segments that could be folded in constant time. This enabled us to reduce the runtime of our folding algorithm from the classic $O(n^3)$ to $O(n^2)$, simply by changing the way we modelled the problem. Given that Nussinov's algorithm only takes a scoring metric into account, and does not impose any additional constraints to ensure biological relevance, this means that we are still trying to solve the same optimisation problem following the same rules. The size of the partitions used is a parameter that we experimented with (see **Accuracy Of Results** for discussion of good parameter values). For a given partition size, a sequence is broken up into all possible subsequences of that size where the first half of the subsequence is taken from the beginning of the sequence, and all possible second halves are considered. See *Figure 1* for an example partitioning. After we have our list of possible partitions, we use Nussinov's Algorithm to fold them and compute a score for the folding. The partition with the best score is then added to our overall folding for the sequence, and the bases involved are removed from the input sequence. This process is repeated until all bases have been either paired or classified as unpaired bases, giving us a folding of the whole sequence. Throughout, the sequence is represented as a list of bases, where each base is associated with the index of its position in the original sequence. This enables us to keep track of which bases are paired in the context of the input sequence, instead of in the context of the partition being worked on.



t

Figure 1: An example partitioning. In the implementation, each base is tagged with its index in the original sequence, so the final pairing is known even though the sequence is broken up.

Existing Tools Used: We used the RNA Strand Database to get all of our information for this project. We used the querying feature of the database's website to learn about the information the database had to offer, and used the .dp files containing the sequences that make up the database to get the sequences that we used to test our code. We implemented our algorithm in Python, and used the unittest module to test our implementation before running it on actual sequences.

Challenge Undertaken: The RNA folding problem is a computationally complex problem that has yet to be solved to satisfaction. Algorithms are forced to make tradeoffs in terms of tractability and biological accuracy. Our original plan was to use machine learning to develop a whole new approach, and see if we could improve upon the biological accuracy while remaining tractable. While this approach did not work out, we were still able to take an existing algorithm and make it more tractable, while improving the results.

Obtaining Results: We ran both the unpartitioned version and the partitioned version on all of the sequences from the SRP database, a subset of the RNA Strand Database. This gave us 383 data points, containing sequences whose lengths ranged from approximately 50 to 550 bases.

Accuracy Of Results: We measured our results in terms of the "folding score", which refers to the score given to a particular folding by the scoring metric Nussinov's algorithm optimises. This metric gives one point to Watson-Crick pairs and the G-U Wobble, and no points to other pairs or unpaired bases. *Figure 2* displays the relationship between partition length and folding score, averaged across all the sequences tested. We obtained better folding scores with smaller partition lengths. This is probably because smaller partition lengths enable a more fine-grained examination of the possibilities. The folding at one end is less constrained by how the sequence folded at the other end, and the method of selecting partitions from anywhere in the rest of the sequence enables loops and knots to form that are impossible in Nussinov's Algorithm (see **Future Goals** for a discussion of the next steps in analyzing the impact of this knotting capability). What is curious is that as sequence length increases, the score decreases and becomes worse than the average score obtained by using the unpartitioned version of Nussinov's, when we would expect that using partitions of the same size as the input sequence would result in the same folding and thus the same score. However, we did not use partitions large enough to encompass the whole sequence, as they would defeat the purpose of the runtime speedup, so it is possible that the graph would in fact be parabolic with the vertex around 250 bases were we to continue testing partitions of greater size. We also must keep in mind that we are only looking at average results, and results for individual sequences may more closely approximate the expected trend.

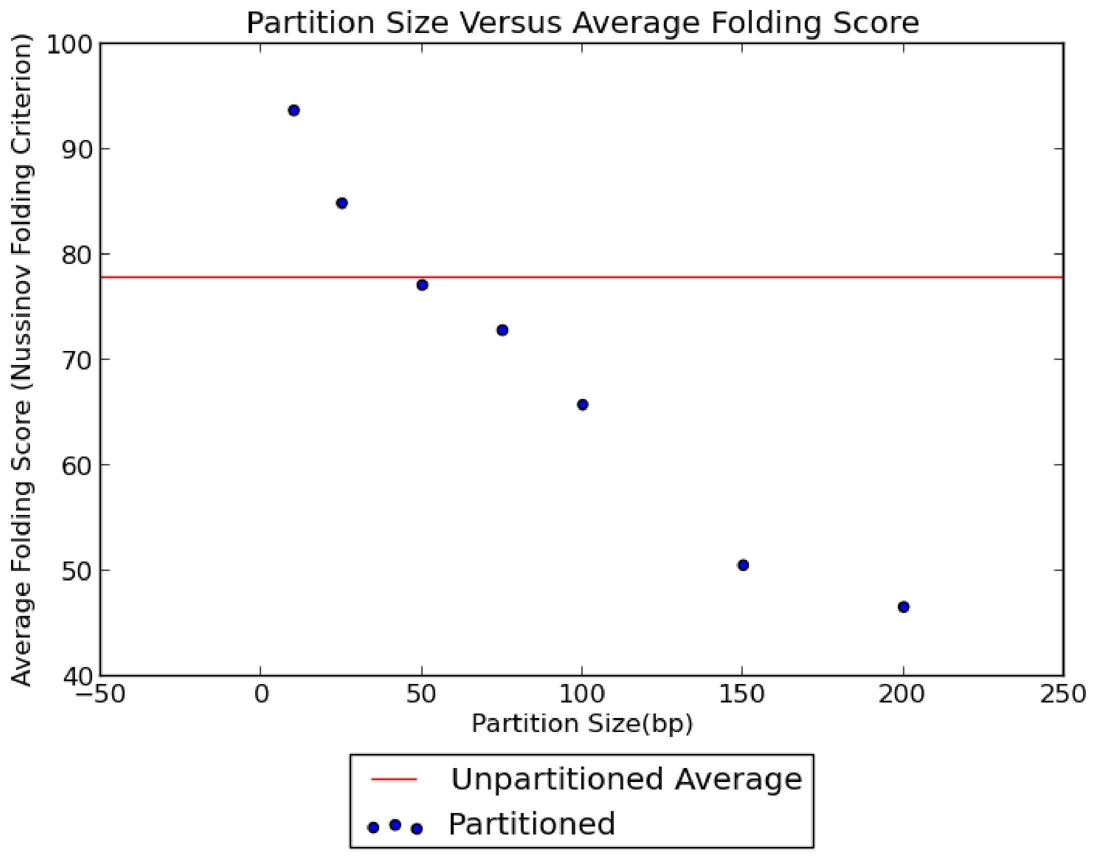


Figure 2: The score given by Nussinov's Algorithm decreases monotonically with partition size.

Runtime Results: Although we know our algorithm to be theoretically faster, we analyzed the actual time it took to run our algorithm to see if the theoretical speedup translated into a real-life speedup. *Figure 3* shows the time it took for the algorithm to run as a function of the length of the input sequence, for different partition sizes as well as for the unpartitioned code. We found that the constant of our $O(n^2)$ algorithm was large enough that the $O(n^3)$ algorithm outperformed it on our trial dataset. Additionally, larger partition sizes generally seem to take longer to run, but the trend is not very defined. *Figure 4* plots the average time it takes to fold each of the partition sizes we tried, in order to make this questionable trend easier to see.

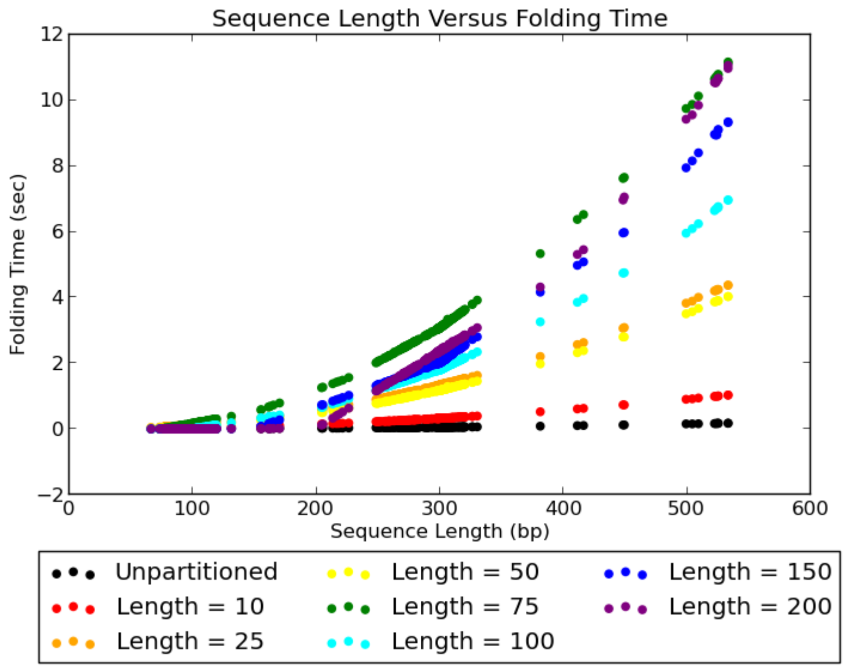


Figure 3: The effects of increasing sequence length can be seen on the partitioned algorithm. Unpartitioned Nussinov's is relatively unaffected.

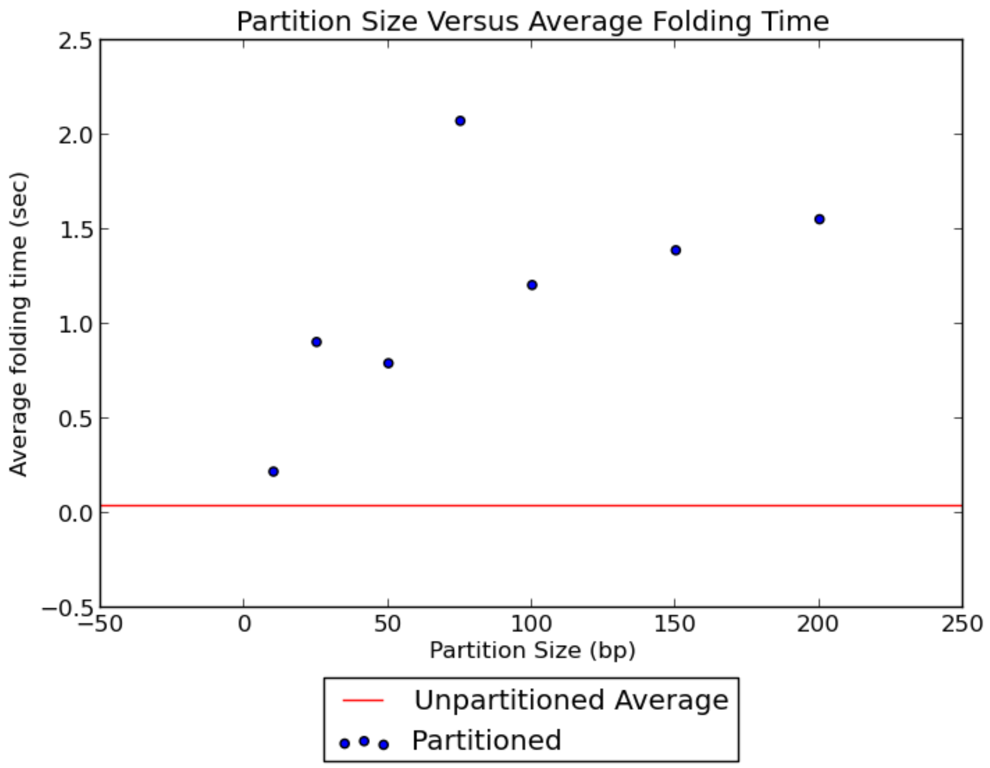


Figure 4: Increasing partition size seems to increase the folding time, but the trend is not clear enough to be conclusive.

Conclusion: Smaller partition sizes result in a better score and quicker runtimes, and are thus preferable. All work is available at github.com/lilyseropian/rna-folding for examination or further experimentation.

Future Goals

Although the theoretical runtime of our algorithm is a big improvement over Nussinov's algorithm, in practice our partitioned code runs slower than the unpartitioned code on the same input. We would like to work on making our algorithm actually run faster in practice. One way to approach this is to use multithreading. Once the data has been partitioned, Nussinov's can be run on each partition in parallel. One of the drawbacks of Python is that it does not support true multithreading (it does have multiprocessing, but that is slower than multithreading), so we would rewrite the code in Java to take advantage of this potential speedup. We would then run it on a machine more powerful than a personal computer, to increase the speedup gained by multithreading, and see if that reduces the overhead enough to make our algorithm a viable alternative to Nussinov's.

Another important step to take before these results are publishable is a more thorough analysis of their biological relevance. We have compared our results to Nussinov's results using the scoring metric that Nussinov's uses to compute the folding, but this doesn't guarantee that our foldings will be accurate to the actual RNA foldings, or even biologically feasible. All we know now is that the scores we obtained are better because we bypassed Nussinov's limitations about knot-like structures, but we have yet to determine whether we have managed to correctly identify pseudoknots or if our results are nonsensical. Some possible ways to test this would be to use a different scoring metric on the foldings our algorithm produces, such as the free energy model used by Zuker's algorithm, or to take some RNAs whose real foldings have been determined by x-ray crystallography and develop a distance metric to compare them to our foldings. Putting our foldings into a visualizer would also likely be helpful in determining biological practicality, if not accuracy.

Comparison With Original Proposal

Our original plan was to use our partitioning algorithm to create input to a classification or clustering algorithm, which would attempt to classify partitions as composing different structural elements. For reasons detailed in our midcourse report, this proved to be unfeasible. Essentially, there was insufficient information and too many limitations for a machine learning approach to be successful in actually learning enough to differentiate between different types of secondary structures, as well as the issue of inability to compare findings based on a list of secondary structure types alone, without a base to base mapping.

For the original proposal, the advice we received from reviewers was to:

- develop a plan for partitioning the sequences
- do a more thorough analysis of the theoretical speedup of runtime
- find a mentor
- elaborate on the plan for classifying/clustering

In response, we successfully developed our partitioning algorithm, did a thorough runtime analysis, and found a mentor. When it came time to flesh out the classification and clustering aspects, we ran into the difficulties discussed above, implying that our original proposal had been too vague in this area, and that we should have figured out the details earlier in the process. Because of this, the focus of our project shifted from developing a completely novel method to improving upon an existing one. Although we did not accomplish our original goal, we did succeed in discovering that the original plan was untenable, despite the validation of the core concept by our reviewers, feedback from course staff, and mentor. We also were able to take part of the original core idea - the partitioning - and apply it in such a way that is still relevant and useful to the field of RNA folding.

Commentary On Experience

If we were to start over, we would do a lot more planning in the earlier stages of the project, considering that we ran into a fundamental problem with the original idea that forced us to change the direction of the project. Being as specific as possible, even to the point of pseudocode would have been greatly beneficial in the earlier stages, because it would have helped us realize sooner the problems with the classifier. We're both happy with the datasets that we used in that we were able to get all the information we needed out of them (and they also came with a nice website that made finding information relatively painless), although we do wish the datasets themselves had had better documentation (that wasn't a large enough problem that we would change the dataset we used, however).

The most challenging part of the project was simply figuring out what to do (which we ended up having to do twice!). It seems like a lot of people were able to build off of ideas from previous projects or their current research, but this class is our first exposure to both computational biology and original research, which is part of what made this experience so rewarding in addition to the challenge. However, it did feel disadvantageous that both of us were undergraduates, as all of the graduate students seemed to be much better equipped to hit the ground running. We most enjoyed the excitement of working on a real problem that nobody had tried to solve in the same way before, and we least enjoyed discovering that our original hopes for the project would not be realised. However, both the encouraging and frustrating aspects were great opportunities for us to learn about what the research process is like.

Commentary On The Peer-Review Process

The Peer-Review process turned out to be largely disappointing. Nicole was unable to attend the review meeting, but spent a good amount of time editing her assigned papers and typing up specific critique. Lily was able to attend the meeting, but found that her time was not used well amongst classmates who arrived late and unprepared to review papers seriously. When we received our peer reviews, it seemed as though others had not put as much effort into reviewing us as we had them--only one of the three reviews was truly helpful. Based on this review's advice, we altered our proposal to contain a more detailed description of our algorithmic approach, including a partitioning diagram and runtime analysis. However, more serious reviewers may have caught the flaws in our plan that led to us having to redetermine the direction of our project.

Division Of Labor

Labor was divided in an attempt to play on each of our strengths, with opportunity for growth and further learning from each other. Because Lily is more comfortable and experienced with computer science, she was primarily responsible for the idea of the algorithm, and for writing the partitioning and folding functions, as well as the functions to graph the results. Nicole, stronger in biology, analyzed the algorithm for biological accuracy and wrote the functions for parsing the data to be used to test the algorithm. Both of us reviewed the overall code, and evenly split the writing of reports.

The main benefit of collaboration was that we were able to check each other's logic, and lessen the burden of the project by splitting up tasks. The challenges of collaboration came from finding time to meet up, and occasional misunderstandings about the workings of the algorithm. If we were to start over, we would try to schedule more time in the beginning to meet with our mentor, and also to begin coding sooner. It wasn't until we tried to write the code for our classifier that we realized its faults, so writing more code sooner, or being more detailed and concrete about our plans from the beginning (eg. what specific attributes would be used to classify and what data is available for that) would have been helpful.

References

Washietl, S. *et al.* (2012) RNA folding with soft constraints: reconciliation of probing data and thermodynamic secondary structure prediction. *Nucleic Acids Res.,* **40**, 4261-4272.

Nussinov R, Jacobson AB, Fast algorithm for predicting the secondary structure of single-stranded RNA.Proc Natl Acad Sci U S A. 1980 Nov; 77:(11)6309-13.

Zuker M, Stiegler P Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. Nucleic Acids Res. 1981 Jan; 9:(1)133-48.

McCaskill JS The equilibrium partition function and base pair binding probabilities for RNA secondary structure. Biopolymers. 1990; 29:(6-7)1105-19.

Dowell RD, Eddy SR, Evaluation of several lightweight stochastic context-free grammars for RNA secondary structure prediction. BMC Bioinformatics. 2004 Jun; 5:71.

Do CB, Woods DA, Batzoglou S, CONTRAfold: RNA secondary structure prediction without physics-based models. Bioinformatics. 2006 Jul; 22:(14)e90-8.

Andronescu M, Bereg V, Hoos HH, and Condon A: RNA STRAND: The RNA Secondary Structure and Statistical Analysis Database. BMC Bioinformatics. 2008;9(1):340.

Westbrook J, Feng Z, Chen L, Yang H, Berman H: The Protein Data Bank and structural genomics. Nucleic Acids Res 2003, 31:489-491.

Cannone J, Subramanian S, Schnare M, Collett J, D'Souza L, Du Y, Feng B, Lin N, Madabusi L, Muller K, Pande N, Shang Z, Yu N, Gutell R: The comparative RNA web (CRW) site: an online database of comparative sequence and structure information for ribosomal, intron, and other RNAs. BMC Bioinformatics 2002, 3:15.

Andersen ES, Rosenblad MA, Larsen N, Westergaard JC, Burks J, Wower IK, Wower J, Gorodkin J, Samuelsson T, Zwieb C: The tmRDB and SRPDB resources. Nucleic Acids Res 2006, 34(Database issue):163-168.

Sprinzl M, Vassilenko K: Compilation of tRNA sequences and sequences of tRNA genes. Nucleic Acids Res 2005, 33(Database issue):139-140.

Brown J: The Ribonuclease P Database. Nucleic Acids Res 1999, 27:314-314.

Griffiths-Jones S, Moxon S, Marshall M, Khanna A, Eddy S, Bateman A: Rfam: annotating non-coding RNAs in complete genomes. Nucleic Acids Res 2005, 33(Database issue):121-124.

Berman HM, Olson WK, Beveridge DL, Westbrook J, Gelbin A, Demeny T, Hsieh SH, Srinivasan AR, Schneider B: The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids. Biophys J 1992, 63(3):751-759.