CISC 471: Computational Biology, Homework 3

EVELYN YACH and DANIEL OH

In this report, we aim to explore a number of algorithms that solve the Motif Finding problem. We explore a Greedy search, Randomized search and the Gibbs Sampling Method. Overall each algorithm demonstrates the following situational strengths. Greedy gives the fastest but most approximate result, Randomized offers a longer execution time but better approximation and Gibbs offers a solution that lies in between the two.

ACM Reference Format:
Evelyn Yach and Daniel Oh. 2021. CISC 471: Computational Biology, Homework 3. In *Computational Biology '21: Homework 3; February 26, 2021; Kingston, ON*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The Motif Finding problem is an important biological problem to both solve and implement. The motivation for this is that gene clustering is highly dependent on both gene expression and dependencies. If we assume that dependent genes are co-regulated, then we can examine their promoter regions for conserved motifs, thus confirming their relation.

2 ALGORITHMS

2.1 Greedy Motif Search

When our algorithm was passed the sample dataset on Rosalind, it returned the sample output.
Time Complexity: $\theta(nlk^2)$ where n is the number of DNA strings, l is the length of each string of DNA, and k is the length of k-mer motifs(the k parameter).
Space Complexity: $\theta(2n + 3m)$ where n is the number of DNA strings and m is the length of each string of DNA.

2.2 Randomized Motif Search

Running our algorithm on the Rosalind sample data, we found that it returned the correct answer consistently when running the second data set, but produced varying results on the first and third. The results from these runs alternated between two distinct sets of motifs. One of these sets matched the correct answer for every base, and another seemingly chose at random. We believe that this variation can be attributed to the parameters that are passed into the algorithm causing a varying result to be produced. This is an expected result, as the Randomized Motif search we are using is based off Monte Carlo algorithms which are known for being fast in finding approximate solutions.
Time Complexity: $\theta(1000 * nk^2)$ where n is the number of DNA strings and k is the length of k-mer motifs(the k parameter).
Space Complexity: $\theta(5n + m)$ where n is the number of DNA strings and m is the length of each string of DNA.

Computational Biology '21, February 26, 2021, Kingston, ON

ACM ISBN 716-5-7490-XXXX-X/21/26…$69.69
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2.3 Gibbs Samplers

Our algorithm returns a different output each time it runs on the Rosalind sample data and any other dataset. We suspect that this is caused by the aspect of random number generation in the algorithm. As such, despite not getting exactly what the Rosalind sample output suggests, we believe that the output is in fact correct. Further evidence is provided in the fact that various test runs on the Rosalind input set occasionally generated a portion of the expected output. As an example, we observed 'TCTCGGGG' being returned as a best motif by Gibbs. This string is also found in the sample output on Rosalind

Time Complexity: $\theta(n * 4m * k^2)$ where n is the number of iterations(N parameter) and k is the length of k-mer motifs(the k parameter).

Space Complexity: $\theta(n + 3m)$ where n is the number of DNA strings and m is the length of each string of DNA.

3 ALGORITHM COMPARISON

3.1 Time Complexity

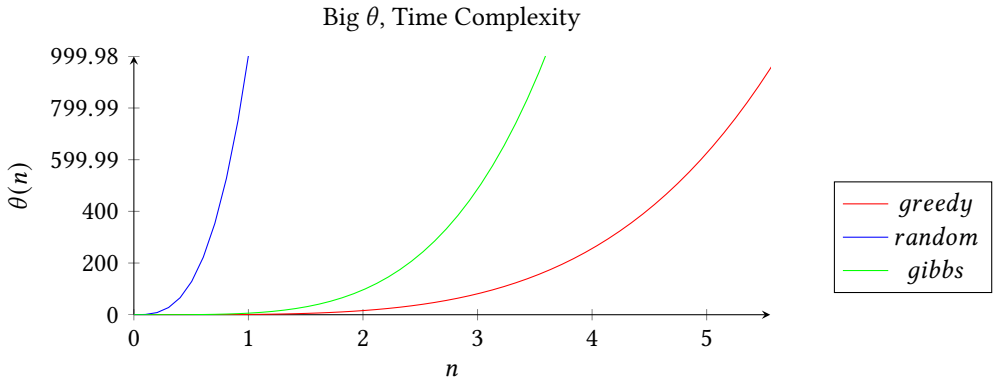

Fig. 1. Comparison of the time complexity of Greedy Motif Search, Randomized motif search and Gibbs Sampling.

3.2 Space Complexity

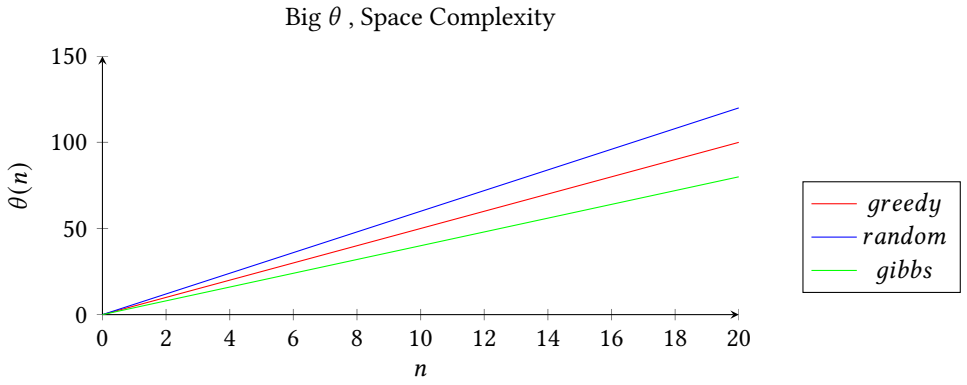

Fig. 2. Comparison of the space complexity of Greedy Motif Search, Randomized motif search and Gibbs Sampling.

3.3 Runtime Comparison

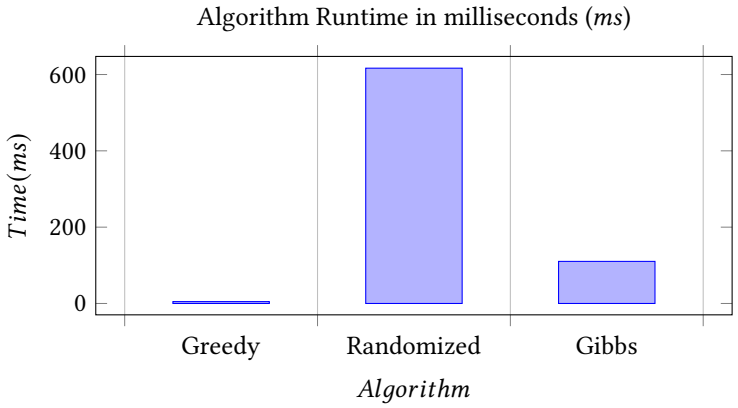

Fig. 3. Comparison of the time taken to complete of Greedy Motif Search, Randomized motif search and Gibbs Sampling. The parameters for this test were a dataset $'GGCGTTCAGGCA', 'AAGAATCAGTCA', 'CAAGGAGTTCGC', 'CACGTCAATCAC', 'CAATAATATTCG'$, along with k and t parameters where $k = 3$ and $t = 5$.

4 DISCUSSION

As expected, Greedy Motif search is the fastest of the three algorithms, followed by Gibbs Sampling and Randomized Motif search respectively. Randomized Motif search is an algorithm designed to be run over many iterations to produce an optimal solution which when doing so, increases the execution time significantly. It is important to note that our results for Randomized Motif search take this step into account.

It is once again not surprising to find that Randomized Motif search exceeds the other algorithms in terms of space complexity. Since our algorithm saves both the best motifs and current iteration motifs in both the loop which runs the algorithm many times and in the main loop within the algorithm itself, it is natural that it would take up more space. Fortunately, the space complexities for all three algorithms are linear, meaning that space is not a large concern when deciding which of the three algorithms to use.

5 CONCLUSION

A Greedy Motif algorithm will select the most attractive alternative at each iteration which despite executing very quickly, will generally fail to produce an exact solution. In situations where you need a single quick run and speed is a high priority, this can be good choice.

A Randomized Motif search is similar to Greedy in that they are both quick and mediocre at producing an exact solution. As discussed earlier, Randomized Motif search is ideal if you have the time to do the large number of iterations it needs to arrive at a good answer.

Gibbs Sampling is more cautious in its iterative approach. It will discard a single k-mer from a set of motifs at each iteration, and then decide whether to keep or replace it. Each step here will change a single motif as opposed to Random Motif search which has the possibility to change every k-mer. A risk with using Gibbs Sampling is that it may converge on a sub-optimal solution, particularly for difficult search problems. Best used if speed is not a high priority and the problem isn't overly difficult.