Motif Finding

HERSHIL DEVNANI*, Queen's University, Canada

CCS Concepts: • **Applied computing** → **Computational genomics**.

1 INTRODUCTION

1.1 The Motif Finding Problem

1.1.1 The Significance of Motifs. Short transcription factor binding sites, know as regulatory motifs, regulate gene activity and DNA repair. The significant relevance of motifs is the fact that a single motif pattern may be present multiple times throughout DNA sequences, providing insight on proteins responsible for the regulation of a those genes. This information gives us tremendous potential for mapping transcription factors to help classify genes with commonly occurring motifs.

1.1.2 Comparing DNA Sequences in the Search for Motifs. The basic computation premise for finding motifs across sequences of DNA is as follows: given a set of DNA, find a k-mer that appears most frequently in all given DNA. Many factors that need be taken into consideration, such as the number of DNA string we are comparing, length of the motif we wish to search for and conservation of the motifs across the DNA set. Three algorithms discussed attempt to to compute an optimal set of motifs using a basis of scoring sets of motifs. Each algorithm is optimal in different use cases, based on factors such as number of DNA sequences and length of motifs we are searching for. This report will aim to compare both the time and space complexity of each of the algorithms and recommend the ideal use cases in which each algorithm is optimally suitable.

2 THE ALGORITHMS

2.1 Greedy Motif Search

2.1.1 Introduction. The premise for Greedy Motif Search uses basic principles of a greedy algorithm. The greedy aspect of the algorithm selects motifs based on the profile most probable motif of the previously visited DNA strands.

2.1.2 Input Vars. t = no. of DNA strs; DNA = DNA set; k = k-mer length; n = length of DNA strs.

2.1.3 Time Complexity. Time complexity estimate: $O(n-k+1)*(t^2)$.

2.1.4 Space Complexity. Space complexity estimate: $O(4*k+t)$.

2.1.5 Run Time Analysis. Outer loop is run t times number of k-mers in the first DNA string, i.e. loop over every k-mer in the first DNA string. The number of k-mers in the DNA string is $(n-k+1)*t$. The number of operations to generate a new profile each loop is t^2 since each iteration the profile factors one more motif, resulting in each iteration adding polynomially.

2.2 Randomized Motif Search

2.2.1 Introduction. The premise for Randomized Motif Search is to run many sets of selecting random k-mers from each DNA string, building a profile, then building a new set of profile based k-mers. This very quickly approximates k-mers and we can take advantage of this speed to run many iterations and keep track of the best score we see through the iterations.

2.2.2 Input Vars. t = no. of DNA strs; DNA = DNA set; k = k-mer length; n = length of DNA strs.

2.2.3 Time Complexity. Time complexity estimate:$O((n\check{}k+1)*t*t*k) = O(t*k*(n\check{}k+1)*log(t))$

2.2.4 Space Complexity. Space complexity estimate: $O(4*k+t)$.

2.2.5 Run Time Analysis. Outer loop is run t times, but we only run the loop until we don't find a better scoring set, so we can estimate $O(log(t))$t outer loop iterations. Next, we generate a new profile every iteration, which is $O(t*k)$. Building the new set of motifs off of profile is an operation that loops over each DNA string $(n-k+1)*t$ times.

2.3 Gibbs Sampler

2.3.1 Introduction. The premise for Gibbs Sampler is similar to Randomized Motifs Search, where a random set of k-mers is selected to start out with. However, instead of re-building the entire set of motifs, Gibbs Sampler selects one random k-mer to rebuild based off of the current profile.

2.3.2 Input Vars. t = no. of DNA strs; DNA = DNA set; k = k-mer length; n = length of DNA strs; N = no. of motif reconstructs.

2.3.3 Time Complexity. Time complexity estimate: $O(N*t*k*((n\check{}k+1)*t))$

2.3.4 Space complexity. Space complexity estimate: $O(4*k+t)$

*Sole author of this research.

2.3.5 Run Time Analysis. Inner loop is run $O(N)$ times. Selecting a set of random starting point k-mers is a constant operation, $O(1)$. Building a new profile every iteration is $O(t * k)$. Selecting a random k-mer, then re-building it based off the current profile matrix takes as many operations as there are k-mers in the selected k-mer's parent DNA string $(O(n - k + 1) * t)$.

2.4 Storage Analysis for all Three Algorithms

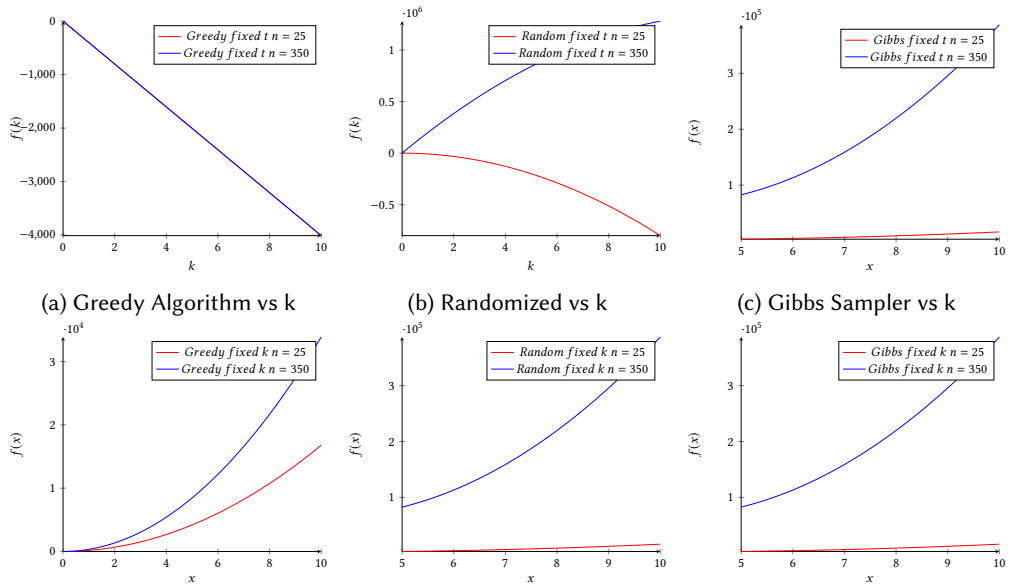
A $4 * k$ 2D array for the profile matrix is created every loop iteration. This array only needs to be counted once since it is re-created and the old one can be removed from the stack. We also create the list of k-mers of size t.

2.5 Analysis of Implementations (Why are the Algorithms Correct)

All data sets from the text and multiple from Rosalind procure correct results on my implementations of the three algorithms. With Randomized Motif Search and Gibbs Sampler, however, since both do not guaranteed an optimal solution each time, the Rosalind trials passed 6 out of 7 times.

3 COMPARATIVE ANALYSIS OF THE THREE ALGORITHMS

3.1 Time Complexity

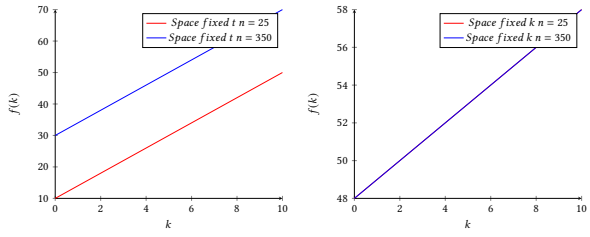


(a) Greedy Algorithm vs k (b) Randomized vs k (c) Gibbs Sampler vs k

(d) Greedy Algorithm vs t (e) Randomized vs t (f) Gibbs Sampler vs t

Fig. 1. Time Complexity Estimates Visualized

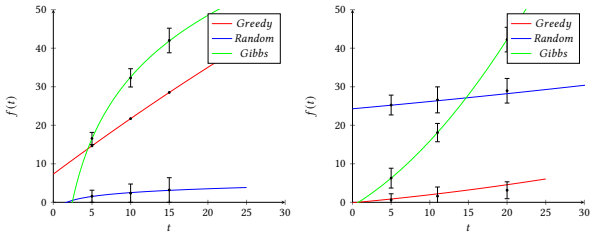
3.2 Space Complexity



(a) All vs k (b) All vs t

Fig. 2. Space Complexity Estimates Visualized

3.3 Real-Time Running Time



(a) All vs k (b) All vs k

Fig. 3. Average of 3 Trials of Real Time Complexity Estimates per Algorithm

4 DISCUSSION

Two trials were conducted to quantitatively measure the time complexity and performance of each algorithm. The first trial varied the number of DNA strings, t, while keeping the k-mer search length, k, constant at 12. The second trial varied the size of the k-mer, k, while keeping the number of DNA strings, t, constant at 25. Both trials were run 10 times; 5 times each for DNA string length of 25 and 350 to give a spread in efficiency in low vs high values for the DNA string length.

4.0.1 First trial – constant n of 25 and 350, constant k-mer of 12, varying t. The Greedy algorithm performs searches proportional to the ratio of k to t, so the number of operations are highly dependent on both t and number of k-mers per DNA string. With Random Motif Search, as the number of DNA strings increases, we have to increasingly build a new profile and set of motifs. We run the main loop t times in a given iteration; as we increase t, we are increasing the number of motifs we have to select. With Random Motif Search, we are able to make better choices in conserving computational operations and optimizing score, thus we see no drastic changes in one run, but over many runs get a much more accurate optimal motif set. Gibbs Sampler further improves on Random Motif Search by not relying as heavily on the given number of DNA strings. Gibbs Sampler only selects a single motif to reconstruct and repeats this for a given constant number of iterations.

4.0.2 Second trial – constant n=25 and 350, constant t of 25, varying k-mer. The Greedy Motif Search iterates over every k-mer in first DNA strand. Therefore, as k decreases, the number of k-mers in the first strand also increases, resulting in slower runtimes for short k value to n value (length of DNA string) ratios, i.e., small k value but large n value. This results in a faster runtime when n is constant and k grows since we have less k-mer options and loop iterations. With Random Motif Search, we have a decreasing pattern but in a polynomial fashion. The most time-consuming computation is rebuilding an entire set of k-mers on each main loop execution. As k increases, the k-mer search space per DNA strand decreases, resulting in the run time of the algorithm decreasing. For Gibbs Sampler, we also see a negative linear correlation as k increases. Gibbs Sampler decreases in runtime as k increases since the k-mer search space is reduced on each motif rebuild. Since k is not as significant of a constraint as it is with Random Motif Search, we only gain a linear decline in Gibbs sampler as k increases.

4.0.3 Space Complexity. If we consider the space complexity for the algorithms, each algorithm constructs profile matrices and conditionally rebuilds motif matrices. As a result, the time complexity is driven by the space allocations for the 2D $4 * k$ profile matrix and the $t * k$ motif matrix for all algorithms. We only need to modify the values, while the underlying structures are the same. Therefore, the space complexity is the same for all algorithms.

5 CONCLUSION

Considering t and k, Greedy Motif Search works well when both t and k are small. Randomized Motif Search works well when t is relatively small, with k up to 15, to generate reasonable approximations. Gibbs Sampling is ideal when we have larger data sets, we know we are looking for motifs or some patterns and want a high degree of accuracy.

Greedy Motif Search can generate reasonable approximations to give indication if we have potential basis to continue searching for motifs. This algorithm is ideal to offer quick identification of patters, with advantages of quick implementation, fast runtime and no need for multiple iterations.

Randomized Motif Search is ideal larger data sets and providing quick estimates. This algorithm can be used for a high-level pattern analysis of a set of DNA strings to gauge if a DNA set contains potential common k-mers. The caveat is that algorithm is random, so we have to run multiple iterations to sample different random starting points, upwards of 1000s of iterations, for reasonable estimated solutions. Randomized Motif Search adds time complexity versus Greedy Motif Search, but since we control the number of iterations, we can manually adjust the quality of our estimate versus time ratio to our liking.

Gibbs Sampler is ideal for large data sets, especially as t grows, and provides the best estimates amongst the three algorithms. Gibbs Samplers biggest trade-off is time for accuracy, however, with two variables available to control the iterative steps, we can balance our tolerances for speed versus accuracy. The accuracy comes from the fact that we are more cautious with our operations and selectively randomize motifs using calculated biases. Gibbs Sampler may converge at local optima, so we need to run Gibbs Sampler multiple times, upwards of 100s of iterations, to find the best possible set of motifs.