

Greedy Motif Search

Muhammed Shajahan

S2 AIE ‘B’

AM.EN.U4AIE21144

Akshay Krishnan T

S2 AIE ‘B’

AM.EN.U4AIE21109

Navneeth Suresh

S2 AIE ‘B’ AM.EN.U4AIE21147

Niranjan Prasanth

S2 AIE ‘B’

AM.EN.U4AIE21148

Aswin US

S2 AIE ‘B’

AM.EN.U4AIE21119

***Abstract*—** **DNA motif discovery is an important problem in bioinformatics and it is essential for identifying transcription factor binding sites that play key role in the gene expression process. Motifs are generally short patterns which repeat among a set of DNA sequences. However, it is computationally expensive and impractical to find them with exhaustive search. Therefore, probabilistic and heuristic approaches can be used for tackling this problem. This study focuses on greedy construction algorithms for finding DNA motifs. First, we develop a GRASP algorithm with candidate list reduction with a one-sequence- change neighborhood based hill climbing local search. Then, we evaluate and discuss the performance of the developed algorithm on 3 common datasets, namely hm03r, yst04r, and yst08r. The results obtained show the effectiveness and usability of the proposed method for the DNA motif discovery problem..**

***Keywords—Motifs,*** ***transcription factor , k-mer***

1. **INTRODUCTION**

Due to expanding biological research needs and quick technology advances in gathering such data, the use of genetic data for biological research has been expanding quickly. Sequencing machines read DNA sequences (and other types of sequences) as part of the traditional method of creating raw genetic data.

The output of current sequencing machines is typically a very large in number (tens of millions) of short strings of characters.

Each of these strings, referred to as "reads," corresponds to a brief segment of a lengthy DNA sequence. Since the DNA is made up of pairings of four nucleic acids, the reads that represent its many components are strings made up of the four potential characters/letters A, C, G, and T, each of which stands for a different nucleic acid.

A species's individual members each have a unique DNA sequence (moreover, each human has two versions of most of his or her chromosomes and, in fact, small variations may exist among the many cells in a human body). The following scenario is frequently seen in the analysis of such differences: one is interested in comparing the DNA of one specific member of the species to the reference, and one possesses some "reference" .

‘Motif discovery’ (or ‘motif finding’) in biological sequences can be defined as the problem of finding short similar sequence elements (building the ‘motif’) shared by a set of nucleotide or protein sequences with a common biological function. The identification of regulatory elements in nucleotide sequences, like transcription factor binding sites (TFBSs), has been one of the most widely studied flavors of the problem, both for its biological significance and for its bioinformatic hardness [1, 2].

This first step of gene expression, ‘transcription’, is finely regulated by a number of different factors, among which ‘transcription factors’ (TFs) play a key role binding DNA near the transcription start site of genes (in the ‘promoter’ region), but often also within the region to be transcribed or in distal elements like ‘enhancers’ or ‘silencers’ [3, 4]. The actual DNA region interacting with and bound by a single TF (called TFBS) usually ranges in size from 8–10 to 16–20 bp. TFs bind the DNA in a sequence-specific fashion, that is, they recognize sequences that are similar but not identical, differing in a few nucleotides from one another.

The basic idea of the greedy motif search algorithm is. to find the set of motifs across a number of DNA. sequences that match each other most closely.Here the greedy algorith has some qualities like and it does no search for all possibilities of a particular thing we are searching. Greedy will only consider some initial conditions and will always persue in that direction whilre searching.Offcourse it may or may or may not lead us to the global solution, but definetly we will getan important local solution which can resemble the global solution.Especially when it comes to DNA, it doesn’t matter whether it is a global solution or not, both are going to be similiar.

Basically greedy chooses the best in each case, and there is no backtracking, it is local solution and not an exhaustive search.Exhaustive search means it does not go and seach for all possibilities, rather it will just consider one or two initial possibilities and come at a solution.

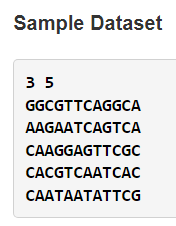
The major advantage is that this method uses very less time and results closely resembles the known motifs.Remaining algorithms, which search for all possibilities which we will get global solution takes hours or even days to get results since genomes are really really long.So that is the reason why we are using motif discovery using Greedy since it takes very less time and results closely resembles the known motifs. We will be getting set of patterns which are extremely similar and that’s more than enough since we are only looking for motifs which we do not know how they look in prior.

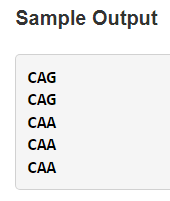
1. **PROBLEM STATEMENT**

**Given:** Integers k and t, followed by a collection of strings Dna.

**Return:** A collection of strings BestMotifs resulting from running GreedyMotifSearch(Dna, k, t). If at any step you find more than one Profile-most probable k-mer in a given string, use the one occurring first.

1. **SAMPLE DATASET & OUTPUT**

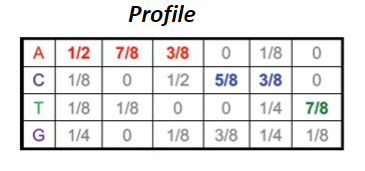
****

****

1. **METHODOLOGY**
2. *Forming the DNA matrix:-*

*If a set of DNA or Single string DNA, Then to make the process easier we will be converting that into a matrix by dividing DNA at equal intervals.*

1. *Selecting the 1st seen k-mers to create profile matrix:-*



1. *Selecting the kmer (motif\_1) with highest score from 1st row.*

*The profile obtained earlier is used to evaluate each kmer in the 1st row of the DNA matrix*

*Eg: let a sequence be ACGTACG AND k=3*

*From these sequence ACG,CGT,GTA,TAC…ie, all possible 3-mer from sequence is taken and sent to evaluation using the Profile matrix, we will be getting a value.*

*From the values obtained, one with the highest score is taken and that will be our 1st Motif and is added to the motif matrix*

1. *Updating the Motif matrix with a new motif (i):-*

*1st motif will get added, ie the motif matrix now have only one entry*

*For Eg; let TAC be the the one with highest score from the above stated example*

*Then matrix will be [TAC].*

*Now we will be creating the profile matrix for [TAC]*

*Ie,we have a new profile matrix now.*

*This profile matrix will be used to evaluation of the kmer with highest score in 2nd Row of the DNA matrix just like we did for the 1st row.*

*Thus we will get a motif and that is added to the motif matrix.*

*Now using these two, we will be creating new count matrix and profile matrix. Using this updated profile matrix, we will be evaluating the best k-mer from the 3rd row and added to motif matrix.*

*This process will keep on adding.*

*So the number of rows the DNA matrix have, that many entries will be there in the motif matrix. Ie that many motifs we will get.*

1. *Repeating this step for every other row:-*
2. *Returning the motif matrix which will contain*

*Motif\_1*

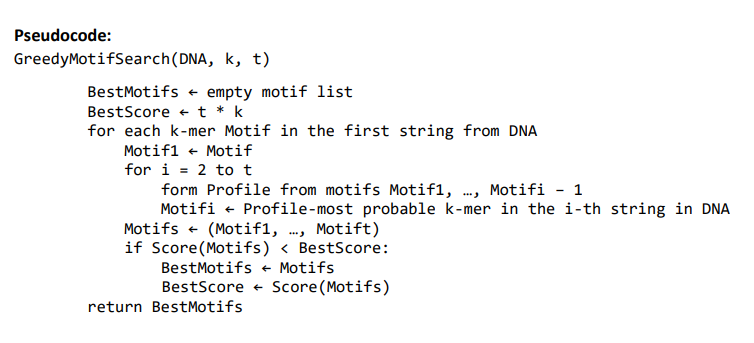
*Motif \_2*

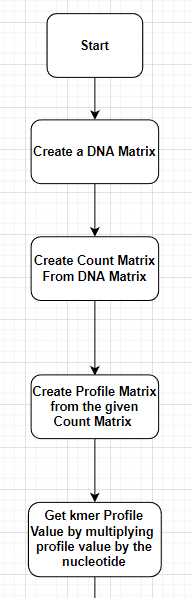
*….*

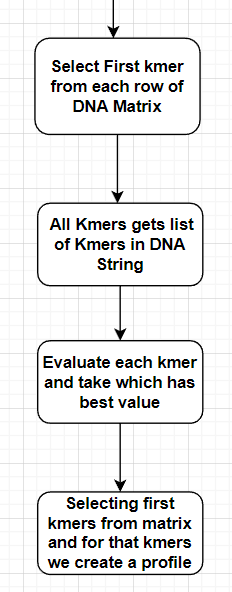
1. *Algorithm idea:-*

Greedy Motif Search Algorithm Our proposed greedy motif search algorithm, GreedyMotifSearch, tries each of the k-mers in DNA1 as the first motif. For a given choice of k-mer Motif1 in DNA1, it then builds a profile matrix Profile for this lone k-mer, and sets Motif2 equal to the Profile-most probable k- mer in DNA2. It then iterates by updating Profile as the profile matrix formed from Motif1 and Motif2, and sets Motif3 equal to the Profile-most probable k-mer in DNA3. In general, after finding i − 1 k-mers Motifs in the first i − 1 strings of DNA, GreedyMotifSearch constructs Profile(Motifs) and selects the Profile-most probable kmer from DNAi based on this profile matrix. After obtaining a k-mer from each string to obtain a collection of Motifs, GreedyMotifSearch tests to see whether Motifs outscores the current best scoring collection of motifs and then moves Motif1 one symbol over in DNA1, beginning the entire process of generating Motifs again.

1. *Pseudo Code:-*

**

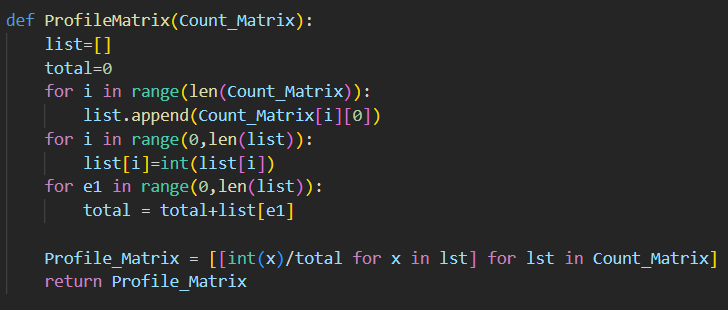
1. **PROGRAM FLOW**



1. **DISCUSSION**

At first, a code was made that was successfully able to generate the neighbourhood of the given string. However, a drawback was found in the code. The time complexity of the code was large when implemented on large datasets.

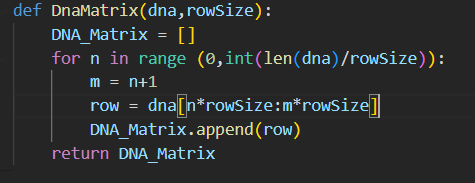
So, another code was made to reduce the time complexity for the same given logic of the first code that was made. In this code, the code size was reduced as much as possible. The program once again successfully generated the required

output. This code matched the time complexity for all available codes across the internet.

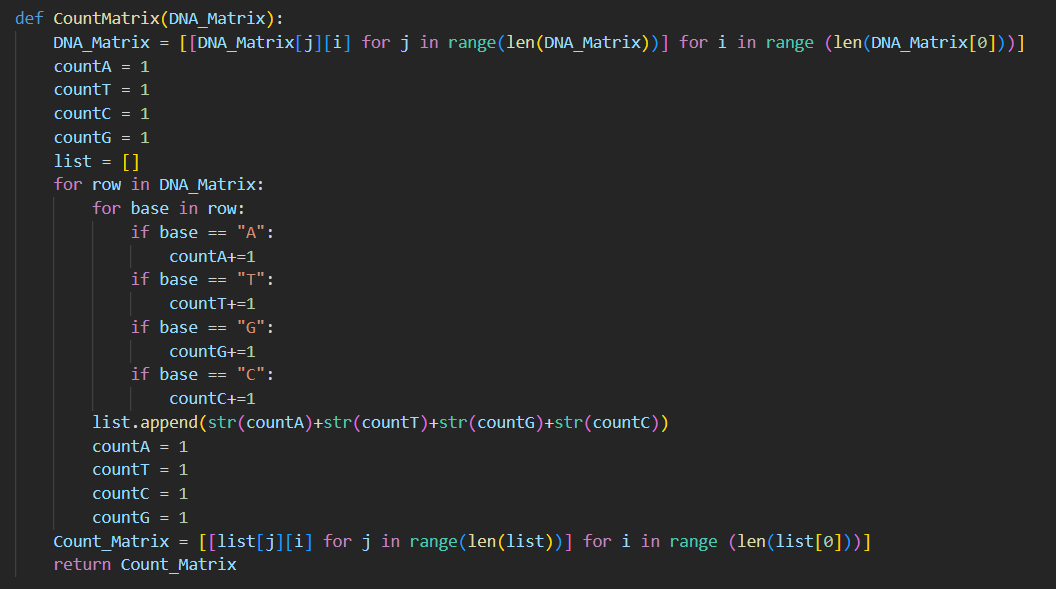
Another challenge was put up to check whether a new code could be made to become the best code for our current problem. The only found was to cut down the function to generate all the k-mers. However, in order to do this a new and more efficient system had to be made to generate said k- mers0.

1. **IMPLEMENTATION**
2. *Code 1:-*

At first, the functions were defined.



Above is the function to create a DNA matrix and it will loop through that cuts the DNA at a desired rowsize and will append to the rows

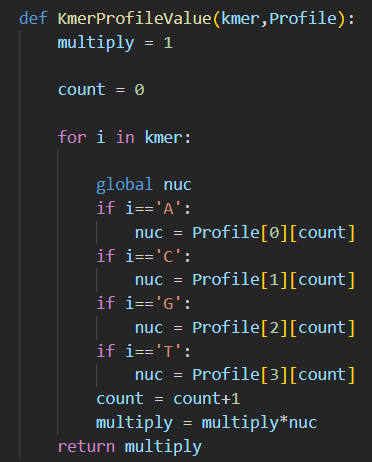
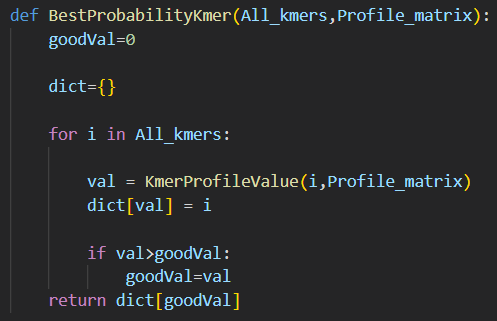


Then we create a function CountMatrix, in which the matrix can

Only be created only if a DNA Matrix exists. We initialize counters and increment values accordingly and append to list, then we take transpose and return the count matrix

Then we create a function Profile Matrix which is a sum

total of rows of the count matrix and divide by the total.  
 Here we convert to integer as we will get count matrix as string.



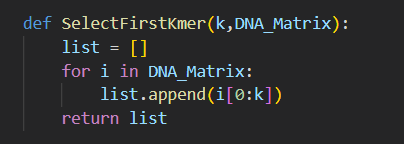


Then we create a function KmerProfileValue, which basically

is, if there is a kmer of a specific k value and a profile matrix

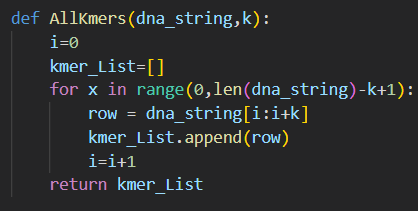
and it takes value from profile matrix and it multiplies and store

the value and returns the result for that specific kmer



Here is where the greedy part starts, we have to select the first

kmer from each row of DNA Matrix



Here if we give a dna string and length k which are all possible

kmers , here to get the last kmer we use the above code length of

dna string - (k+1)

If we all kmers and a profile matrix , we evaluate each kmer and take the best value

This is the main function where we create a main DNA matrix

We select the first kmers from that matrix and we create a profile

for that kmers

1. **CONCLUSION**

An

Algor for quickly aligning DNA readings to rerence genome has been developed. The method handles incompatibilities by default, and it has been proved that extra heuristics allow it to handle additions and deletions as well.

We have successfully created a base algorithm to generate the neighbourhood of a string. Also, in the process we have bettered the code along with an improved algorithm to obtain the required output.

However, the logic for computing the neighbourhood of a string still remains the same throughout the various codes produced.

1. **ACKNOWLEDGEMENT**

We would lito thank Dr. Manjusha Nair M and Dr. Aswathy R Nair for their assistance in this project. We would also want to thank our friends and family for sharing their insightful ideas with us for this study.

Furthermore, we would like to thank Amrita University and our Chancellor Amma for allowing us to benefit from their presence and grace as they guided us through this research.

1. **REFERENCES**

[1] [https://pubmed.ncbi.nlm.nih.gov/20493676/#:~:text](https://pubmed.ncbi.nlm.nih.gov/20493676/" \l "%3A~%3Atext%3DAbstract%2Cnuclear%20lamina%2C%20etc)

[=Abstract,nuclear%20lamina%2C%20etc.](https://pubmed.ncbi.nlm.nih.gov/20493676/" \l "%3A~%3Atext%3DAbstract%2Cnuclear%20lamina%2C%20etc)).

*(about Genomic Neighbourhood)*