Received: January 6, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Intuitive Wizard to Identify Pattern in Similarity Expressed Gene Set of Rice

Preeti Ivy Barrow[1] **Swathi Jamjala Naryanan[1]***

[1]*School of Computer Science and Engineering, VIT University, Vellore, Tamil Nadu, 632014, India*
* Corresponding author's Email: jnswathi@vit.ac.in

Abstract: While the genome sequences of different crops have been published and annotated, relatively little is known about the transcriptional networks that regulate gene expression. There are different ways of finding significant motifs in genome sequences, evolutionary motif finding algorithms being one of them. The work presented in this paper uses Differential Evolution (DE) Algorithm to discover new motifs in rice genome sequences and aims to provide an intuitive wizard to analyse the micro-array analysis of expression patterns (motifs) for the queried genes. The used DE algorithm involves crossover operation with fine tuning. Further, to search for already known motifs parallel version of adaptive hash based pattern matching algorithm has been implemented. The study is conducted on rice genome sequences for twelve chromosomes each consisting of approximately 70000 genome sequences. In this paper, experimental results provide a comparison of serial and parallel version of the algorithm applied for the same dataset which shows that parallel implementation improves the performance by almost 25-30% to search the patterns in genome sequences and provide a list of newly discovered significant motifs.

Keywords: Differential evolution, Genome sequences, Motifs, Micro-array analysis, Pattern matching.

1. Introduction

Advancement of computational computer science greatly helps researchers to understand and explore the complex "omics" network in biological systems. The high-throughput – next generation sequencing (HT-NGS) technologies provides scope to generate huge amount of dataset to understand the central dogma (the coded genetic information hard-wired into DNA which transcribed into mRNA and synthesis a particular protein) in living cell. Development of user friendly computational tools and approaches using computer languages and algorithms offers opportunities to analyse such big data for the interpretation of biologists. In any genomic sequence it comprises with 4 codes of letters that are A, G, C and T which represents four molecules named as Adenine, Guanine, Cytosine and Thymine respectively. In recent years, numbers of enriched genomic databases of agriculturally important crops have been developed for the genome wide expression analysis, whole genome sequence study, genome annotation, promoter identification, cis regulatory elements (small motifs of sequences which play role in gene expression) etc. A pattern search algorithm of small motifs of sequences in promoter region facilitates in identification of master regulatory genes which control the expression of thousands of downstream genes. However, there is a need of user-friendly intuitive wizard to analyze the available data for the subject concern.

The problem of searching motifs from a given gene set is analogous to multiple string pattern matching algorithms. The aim of multiple search is to concurrently search presence of a set of consecutive strings know as patterns primarily present in large and noisy dataset. Various fields like data mining and ware housing [1], network IDS [2], DNA sequences matching [1,3] require innovative approaches to search, discover and analyze significant patterns from data.

The problem of motif finding is defined as finding the repeating patterns in a given biological

data set. This repetitive behavior of patterns provides an indication that these patterns hold some significant structural or functional jobs [4]. Numerous algorithms have been developed to solve the sequential motif finding problem. The pattern discovery is an elementary problem that is associated with finding new motifs in DNA sequences [5]. Generally, the task of finding new motifs starts with a set of regulatory zone of genes which hold small DNA patterns (motifs), present statically. The size of motifs is generally short (<=30 nucleotides) and they don't have gap between them which make them hard to discover amongst large amount of numerical noise. As one genome sequence can contain nucleotides from hundreds to thousands the problem of finding motifs in genome sequences becomes NP-complete problem. Therefore, to discover new motifs, evolutionary algorithms, provide a better way to solve the problem.

The proposed work is to develop a wizard to allow users to search a collection of motifs belonging to rice chromosomes in specific locus id of rice. The wizard shall take patterns to be searched in the gene set as input from the user along with the locus id where search operation need to be performed and provide the user; percentage of each motif found in queried locus ids'. It concentrates on the rice genome, being founded on full-length DNAs, and is intended to get cis-components related with locus ids' that users define. The details are given in Fig.1. The work also aims to discover new motifs present in the rice chromosomes using the DE Algorithm with fine tuning.

The novelty in this paper is the parallel implementation of the adaptive hash based pattern search algorithm (AHPS). The AHPS algorithm has sub-algorithms which involve pre-processing of patterns, memorization of discovered patterns, search operation. In this paper the search operation has been executed in parallel on different rice chromosomes locus id, thus reducing the time of searching patterns in different locus ids.

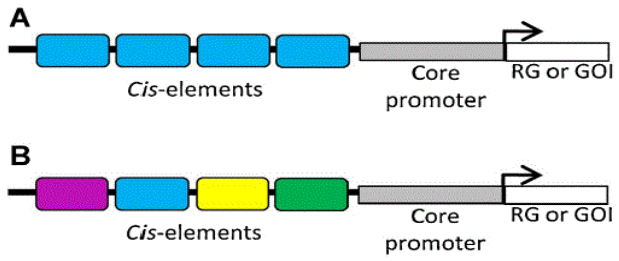

Figure.1 Cis-regulatory elements acting on promoter regions

With the help of multi-threading the execution time is reduced to a significant level. The final output is given as percentage of searched pattern present in each queried locus id. Not only this, the work has also focused on finding new motifs from rice chromosomes making use of DE algorithm. The DE algorithm evaluated the cis-elements by measuring the likelihood score. This score is defined by comparing the PSSM of user-defined gene set and a consensus motif. The wizard can search for a list of motifs defined by the user and also discover new motifs from gene sequences of rice.

The paper is organized as follows. Section 2 gives information about the work done in past to search patterns in strings and the algorithms used to discover motifs. It also talks about the limitations of the methods that led to discovery of new approaches. Section 3 provides the problem formulation. Further in Section 4, we have given the proposed architecture. In Section 5, the dataset and the proposed methodology is given in detail. In section 6, we present the results and discussion followed by conclusion, future work and references.

2. Literature review

Various researchers have implemented several approaches to extract or search motifs from sequences. The broad categories are pattern-driven approach, statistical based approach and machine learning approaches.

Pattern driven approaches enumerate all the patterns to find those appearing with a high frequency in the input sequence. Most commonly used techniques for pattern driven approaches include enumeration, suffix tree, graph, hash table and linked list. In this literature, we focus on algorithms that used hash functions for searching patterns. The reason behind choosing hash functions is its characteristics of fastness compared to other data structures like arrays and lists.

Rabin Karp [6] introduced pattern matching algorithms using hash functions in order to reduce the complexity of the naïve algorithms. But this technique searched for patterns one by one, Knuth-Morris came up with a pattern matching process named after him called as Knuth Morris Pratt (KMP) pattern processing where the focus is to reduce the number of comparisons while searching for patterns. The process followed in this concept is to skip the characters that are already matched.

The time complexity of this algorithm was relatively very less compared to Rabin Karp algorithm. The advantage of KMP algorithm was that the pointer is never decremented in the text. The

pre-processing of the algorithm took *O(m)* time and space. The algorithm was mainly applicable for one pattern that needs to be searched multiple times as like in text editor search [7]. KMP was very well accepted algorithm to search patterns.

Another algorithm proposed by Boyer Moore (BM) worked in similar way of scanning text in text editor but the only difference was in the start of pattern matching process which was from right to left in this approach [8]. As further extension, to simply BM approach, a Boyer Moore Horspool (BMH) algorithm was proposed where a different kind of heuristic was proposed to handle bad characters and it also helped in improving the accuracy of the search algorithm. In the pre-processing phase, the algorithm generates a table which stores details of characters and the number of characters that can be skipped while performing match operation [9]. As the BMH algorithm largely depended on the size of the alphabet and the pattern length, it was further improved by Raita [10] by changing the search procedure of BMH search. The proposed strategy for searching is to compare the last character followed by first character and then the middle character before actually comparing the other characters. This further reduced the unnecessary comparisons. The drawback of this approach was later found to be that it has quadratic time complexity. Another algorithm called quick search algorithm [11] which is yet another simplification of BM was also suffering from quadratic time complexity during search phase in worst case, though it was more suitable for short patterns and large alphabets.

To overcome these shortages, Boyer Moore Smith (BMS) [12] proposed an approach using shifts to overcome the quadratic time complexity. This approach was the fastest when compared to all BM variants. As extensions, Berry [13] proposed hybridization of quick search and Zhu-Takoka string matching algorithm which used fastest loop or otherwise termed as character unrolling cycle. This algorithm is more suitable for long patterns and it uses two consecutive text characters to compute the bad character shift. The complexity of the algorithm is *O(mn).*

Currently lot of motif finding algorithms were proposed to deal with small scale datasets such as PMS8 [14], and qPMS9 [15]. Few other algorithms like F-motif [16] used words to search for motifs instead of character wise search. The algorithm like MCES [17] adopted the process of mining the substrings in input sequences with high occurrence frequency and then combined them as motifs.

Later the research direction shifted towards parallel search algorithms to reduce the time complexity. To keep searching time minimum, algorithms like bit parallel algorithms were developed. These algorithms are capable to match multiple strings in parallel and hence these are categorized as multiple string matching algorithms.

Researchers found that most of the traditional methods are not scaling for large datasets, Salmela [18] proposed a variant of Horspool Algorithm to handle the shortcomings of traditional methods in accommodating large datasets. The extended backward oracle matching algorithm [19] was also proposed to handle long patterns and small alphabets. But the memory utilization of these algorithms was huge. Wu-Manber [20] implemented parallel versions of pattern matching algorithms and released as a tool called agrep tool. There were many other tools that were released to discover motifs. The details are given in Table 1 with different acceptable parameters as given Tran and Huang [21].It can be seen from the table that different tools support different formats, sizes but none of them allow to search for different locus ids from any chromosomes for any number of patterns. Besides the availability of web tools, it is also difficult to customize the problem of motif finding because every tool is specific to certain size of *BaseString* or certain format of genome sequences.

In the recent era, many evolutionary algorithms were used for motif discovery. The most commonly used algorithm are genetic algorithm and particle swarm optimization algorithm. In [22], an iterative approach was proposed to improve the computational efficiency of motif searching by using parallel random search.

Table 1. Web tools for motif discovery

Web tool	**Accept format**	**File max size**	**#of motifs option**	**Motif's size option**
MEME	Fasta	=60000 characters	Y	Y
RSAT peak-motifs	Raw, multi, fasta	Unlimited	Y	Y
GLAM 2	Fasta	=60000 characters	N	N
PScan Chip	Bed	100-150 bp	N	N
Cis Finder	Fasta, plain text	Un specified	Y	N

The proposed strategy was applied over genetic algorithm operations to find good starting positions and to find candidate motifs which help for identifying the best motifs.

Miriam proposed De-Novo [23] which was used to perform optimal search operations to find probable solution for HMM model. Using genetic algorithm, they proved that some local maximas can be overcome easily. The major limitation of this approach was the costly behaviour heuristics they used to calculate the fitness function.

In most cases, researchers who used genetic algorithm and particle swarm optimization technique [24] did not perform the fine tuning of parameters after cross over operations. This research gap is handled in our work by considering differential evolution algorithm which performs better for the research problem considered. The selection of DE algorithm over other optimization algorithms is due to DE's capability to always direct the search towards the most probable region in the feasible region by using its select property. Different mutation schemes like mixed distribution are used to check the performance of the algorithm and for maintaining. The selection of the balance of exploration and exploitations. There are other variants of DE [25-28] available in literature for various problem domains, where the variant differs interns of mutant strategy, objective functions and decision variables.

3. Problem Formulation

The project comprises of two kinds of problems: the first being searching for a particular pattern present in thousands of genome sequences and the second being DE [29] algorithm which involves crossover operation with fine tuning. The searching of motifs in genome sequences requires multiple pattern matching. Multiple pattern matching is a kind of string matching that is used to get all possible positions of a pattern present in an input string.

Definition: Given a string $S=s_0s_1s_2s_3....s_{n-1}$ of length n and a finite set of pattern $P= \{p^0,p^1,p^2,...,p^{k-1}\}$ of size k, has each pattern p^t of length m i.e. $p^t=p^t_0p^t_1p^t_2...p^t_m$ defined over a finite set of characters $\sum$, the size of alphabet is indicated as $|\sum|$ and the total number of patterns is denoted as $|P|$, the problem is to find all the occurrences of all the sequences in the given *BaseString*.

The second problem is the discovery of DNA motifs in the given rice genome data sequences. The motifs are described as nucleic acid sequence patterns that have significant biological importance. Usually, the length of motifs ranges from *~5 to 40* basepairs (bp) and are repeated across different genes or even in the same gene.

Definition: Given a set of DNA sequences (1000 bp upstream), the problem of motif finding is the job to detect overrepresented motifs and at the same time conserved motifs from orthologous sequences that are promising candidates for being DNA binding sites for a ruling protein.

4. Architecture Diagram

The proposed architecture is given in Fig. 2. The data set for the addressed problem is taken from online source http://rice.plantbiology.msu.edu/. The dataset comprises only of the upstream region and parsing of the dataset is done to extract genome sequence of each chromosome.

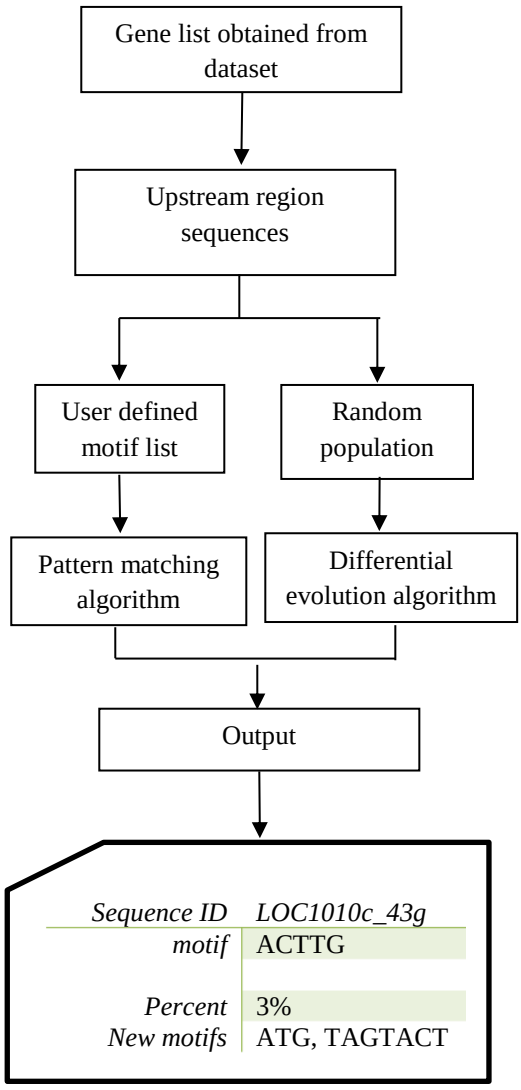

Figure.2 Architecture diagram

In the recent past there has been studies conducted to find the most activated nucleotides in stress conditions like drought, water flooding etc. Using those studies a list of already existing cis-regulatory elements is created to help user in search process. The search process is performed in the pattern matching phase where multi-pattern matching hash based algorithm has been applied. With the help of differential evolution algorithm, the new motifs are discovered.

The project comprises of two actions namely search and find motifs. The search phase takes a list of motifs from the user and then finds it all the chromosomes of rice. The final result is provided in form of present percentage of motifs in each chromosome. The discovery phase uses DE to find new cis-regulatory elements aka motifs. The DE algorithm focuses on maximizing the similarity of subsequences. The subsequence with maximum similarity is a potential motif candidate. The list of all such motifs is provided to the user in the end. The below diagrams shows the various processes involved in the project conducted on rice chromosomes.

5. Experimental setup

In this section, we present the sample dataset and the experimental set up of proposed methodology. The sample genome sequence of rice dataset is given in Fig. 3. From the given genome sequences, the objective is to find the user requested motifs in minimum time and to discover new motifs. The problem of finding user defined patterns on the series of genome sequences is solved using parallel adaptive hash based pattern matching algorithm.

```
>LOC_Os01g01019.1|13101.m00003|1000bp_UpstreamSeq|9798..10800
expressed protein
aaaatggaggaattctgccactttgttctttttttttcctggcacctgttctttttgctag
agggaaaaacagccctggctgggcactgggcagcatactttgcggcccaatacgatagag
cccaaccagggtaggttttcaactaggccggcgagtcatcttccttataatggattcctc
aaccacaatatcaaatgctctagatttcctaattattaaaaggaaaaagtcaaactttaa
ctttcttaaaagtaggccgaatctaattcgtaacccttaaccagaaaaccggataggacg
gattcttaaaagtttggcgtttatgtggcattagaattaaaaatatatgtggagctcaca
tgatcatctctctaatccttcttcctcctctctctcttatctctctcttttctttttcc
ccttcggccctggaggcggtgggtaggacgttgacaggtgctggagcggcggcggcggcg
agcgggtgagggccagagcggcagcggagggtgtgtcggccgattccccttccctttctt
ggctcgccgaaatccctaatccccatgccgcatttcacctcttcccgcgcacaacctccg
catcgcacgcggccggcctctgcacgttcggcgaaggcggatacagccggagcacctcgt
ttagcatgcaccccatcttggtgagctttcccatcgccgccgcgtccatggggccagacc
aaccggtcacctcctccacctcctcccgcaccgccgcccgccactccgggtgcgtcgcca
gcatcagtagcgtccacgacagcgccagcgccgtcgtaagcttcctctctgcggcggcgc
tcgcctcgttcccggccagcagcagccacaacaggtccgtcgtcgtcgtcttcctgccgc
cgccttcacaccgccgccgcgactcgatgatgtcgagcaggagcgcgtcgatcttgcggc
ccagcttccacgcctcgtaggtggcgcggtccgcgcggatATG
>LOC_Os01g01030.1|13101.m00004|1000bp_UpstreamSeq|10774..11776
monocopper oxidase, putative, expressed
gtaggtggcgcggtccgcgcggatatggaggaggctggcgagcgggacgctgacgagacg
cacgcgtggagcggaacagcatcgcctgcagctttgtggaagaccgtggcgtcgtcggcg
gcgatgctgaagctggccttggcgatgatctcggcggcgttccggacaacgcccttctcg
atgtcgatgcagctgtgtcccaacgccactatgtcgctccactcgccgagcatcttcgac
gtcgtctcctccatcacgccgatcatgtcctgcatcatcaacaacaggtacatatgtagc
```

Figure. 3 Dataset Representation

The algorithm consists of four sub-modules to solve the pattern matching problem. The experiment is conducted on 12 chromosomes of rice plant where each chromosome consists of approximately 70,000 genome sequences.

Let the genome sequences for each chromosome be represented as *BaseString*. The first step is to create a table to store the hash values of various unique pairs across all patterns and the table is named as *Match* table. The pairs across all patterns are taken as successive character pairs. Let these pairs be called as *PatPair*. The *Match* table is divided into two sections one part containing the hash value of each *PatPair* and second section holding the corresponding quadruples set values of each *PatPair*. The second section will have as many quadruples sets as many times of the *PatPair* occurs across all the patterns. The quadruples values are as follows:

- *Offset:* Every pattern is identified by it offset number that is nothing but 0, 1, 2 assigned to each pattern. Add that offset number to each *PatPair* corresponding in which pattern it is found.
- *Position: PatPair* last character position.
- *LDist:* Distance of last character of *PatPair* from extreme left of pattern.
- *RDist:* Distance of last character of *PatPair* from extreme right of pattern.

The pattern's first character is taken to start at zero position. Once the *Match* table is prepared then we need to have a window of size *p-1* characters to match hash values of *p-1* characters starting from the beginning of the *BaseString*. The variable *p* represents the length of the smallest pattern. Whenever a match is found processing of set values present in *Match* table will be done and the window slides after all quadruples sets for the corresponding *PatPair* have been processed.

For each match, each set value undergoes steps of checks which are described below:

1. *LPos* which is the start position of probable match is checked for its validity. The validity of *LPos* is defined that it must be greater than or equal to *LPosMin*, where *LPosMin=0* and denotes the start of the *BaseString*. *LPos* is calculated as *CurPos-LDist* where *CurPos* points to current position in the *BaseString*.
2. *RPos* which is the end position of a possible match is checked for its validity by *CurPos+RDist*. The maximum value of *RPos* is the length of the *BaseString*. Once the *RPos*

reaches above that value the processing of that set is stopped.

3. In order to avoid redundant data to enter into final *MatchRecord* table because of same match, a *lay-off* algorithm is used. If the set *LPos* and *RPos* is already present in *MatchRecord* table no more processing of that set is done.
4. The hash value of substring of *BaseString* of length *RPos-LPos+1* is matched with the hash value of the pattern in sync with the offset. If a match is found, then it signifies pattern is found otherwise the processing moves to the next available set.
5. Once a match is found, it is recorded in the *MatchRecord* table and this structure holds all the positions where the matched were found.

A set value in the *MatchRecord* table signifies the starting and ending positions of a pattern found in the *BaseString*. For instance, if a set value (23, 27) indicates that pattern of length 5 is present at location 23 to 27 in *BaseString*. The length of the window is intentionally kept one size smaller than the smallest pattern size to avoid elimination of possible matches of smallest pattern.

The algorithm provides dynamic as well as static search. The search can be dynamic in the sense that patterns can be added on the fly and be searched in ongoing search process. The pre-processing phase involved in the algorithm is used to create a *Match* table. This *Match* table plays a very important role for repetitive search on the same *BaseString* by the user with every time additional patterns. The match table stores the *PatPair* of already searched pattern so only newly added pattern's *PatPair* are added and thus the old *Match* table is used for subsequent searches. This provides adaptive nature to the algorithm and makes the search process dynamic.

The four sub-modules of the algorithm are:

1. Creation of hash table for each pattern. The table consists of offset value for each pattern and its corresponding hash value.
2. Creation of *Match table* which contains all the *PatPair* along with their quadruple sets as described earlier. The *PatPair* are stored in form of their hash value to increase the execution speed.
3. *Lay-off* algorithm to avoid duplication of match sets in *MatchRecord* data structure. This algorithm search for LPos, RPos pair in *MatchRecord* and returns 1 when no match is found.
4. *Search* algorithm which takes the *BaseString* as input and perform search operation on all the given input patterns while returning the *MatchRecord*.

Since the data set is huge and serial processing of files takes time, parallel implementation of the algorithm is done. The search on each file is done in parallel on dual core processor. This will greatly decrease the time of execution as the files are independent of each other. Multithreading concept is used to run the files parallel and apply search algorithm on each files concurrently.

The problem of multiple pattern matching is implemented in Java. Java language is selected as it has efficient data structures such as hash functions for implementation. After running the application several times, the mean execution time of all runs in search phase is taken as the execution time. The time of execution may differ with the nature of patterns to be searched and also based on the architecture of the machine on which the algorithm is implemented. The length of patterns considered lies between 5-40 characters as that is the approximate length of motifs (5-40 bp). As the search of motifs is done in promoter region, the length of *BaseString* is taken 1000 characters which is equivalent to 1000 bp upstream promoter region of each genome sequence.

The second aim of the project is to find the new motifs from the chromosome sequences using Differential Evolution algorithm along with fine tuning. The first step is to create the random population of individuals where each individual is represented by a matrix. The row corresponds to the letter {A, C, G, T} and the column {1,….*l*} represents the probability of each nucleotide at the respective index of the motif. After each individual is evaluated, a trial is created for each member of the population. The selection of the individual for next generation is done based on similarity fitness function. The individual which attains higher fitness value is moved to the next generation.

The aim is to select the individual with higher similarity of sub sequences that forms the final motif. To find the similarity, the first step is to compute the rule value *(rv)* of each nucleotide per motif position. Subsequently, for each motif position we select the highest value using Eq. (1):

$$rv(i)=max_a\{f(a,i)\} \qquad i=1,.....,l \tag{1}$$

where *f(a,i)* is the rank of nucleotide *a* in column *i* in the matrix of position and weight, *rv(i)*is the nucleotide that rules over other nucleotide in column *i*, and the length of the motif is defined by *l*. Similarity is then finally calculated as average of all the rule values as defined in Eq. (2).

$$Similarity\,(S)=\frac{\sum_{i=1}^{l} rv(i)}{l} \quad (2)$$

The novelty of the algorithm lies in the fine tuning added during the time of crossover. This helps in creating more diverse population and hence makes the subsequent population less in common to the initial population. This methodology helps in expansion of the solution space area. The process involves random selection of one column and row and selection of α in range of [0.2, 0.1]. Then α is added to the randomly selected cell and also $\frac{1-\alpha}{3}$ is subtracted from the rest of the cells of the selected column. This still leaves the individual valid as the sum of each column still adds up to one and each cell is in the range [0, 1]. The rand/2/exponential differential evolution scheme is chosen to make the convergence towards the solution faster. The scheme includes five different parameter vectors and two different weighting factors are used during mutation process for generating mutant vector form the parameter vector of current iteration. The strategy for generation of mutant vector is given in Eq (3).

$$V_i(G+1)_{rand/2}=X_{p1}(G)+F_1.[X_{p2}(G)-X_{p3}(G)] + F_2[X_{p4}(G)-X_{p5}(G)] \quad (3)$$

The implementation of differential evolution algorithm is done in MATLAB. The population size is kept as 150 individuals and number of generations as 3000. As per the literature, the length of motif is set between 6-50 bps. The crossover ratio is set to various values as 0.1, 0.25, 0.5, 0.75 and 0.9 but the most optimized solutions are observed at 0.25. The value of mutation factor is kept as 0.02 as studies show it is the most optimal value of *F*. Algorithms

5.1 Motif pattern matching algorithm

Input: The number of motifs to be searched
Output: The percentage of motifs present in each chromosome sequences
Start
Generation of pattern hash table for each motif.
Creation of *Match* table for each unique *PatPairs* found in motifs
Parallel execution of search algorithm on different chromosomes sequences
For each chromosome
Calculate the count of each motif
Calculate the percentage of each motif compared to other searched motifs.
End For
End

5.2 Differential evolution algorithm to find new motifs

Input: Genome Sequences
Output: Motifs with high score
Start
Initialize the population of size N
Population Evaluation
For *i*= 1 to *MaxGeneration*
For *j*=1 to *N*
Create a trial vector for each individual
Apply fine tuning after crossover operation
Evaluate Individual based on score function
Select best individual for next generation
End For
End For
Return best individual with maximum score
End

6. Results and discussion

In this section, we present both the results of searching motifs and generation of new motifs. For searching motifs, we conducted experiments in two different form. First one, by varying the number of patterns and keeping the *BaseString* constant. In the second experiment, the number of patterns is kept constant and the *BaseString* on which motif is to be searched is varied. These results are recorded in Fig. 4 and Fig. 5. In the first experimental set up, the *BaseString* constant is set to 1000 characters and the adaptive hashing algorithm is executed to search for the patterns specified by the user. The execution time taken by the serial and parallel approach is recorded in Fig. 4.

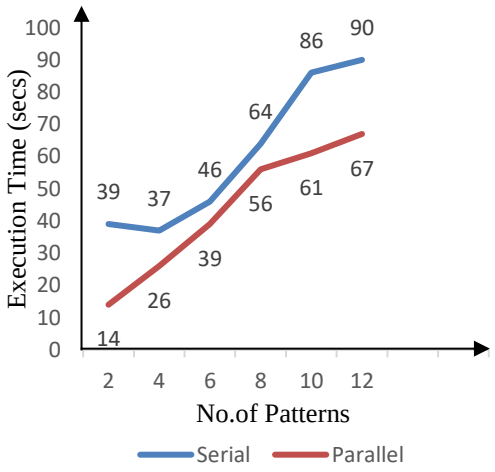

Figure.4 Adaptive hash algorithm's serial vs parallel computation time for searching patterns by varying numbers of patterns

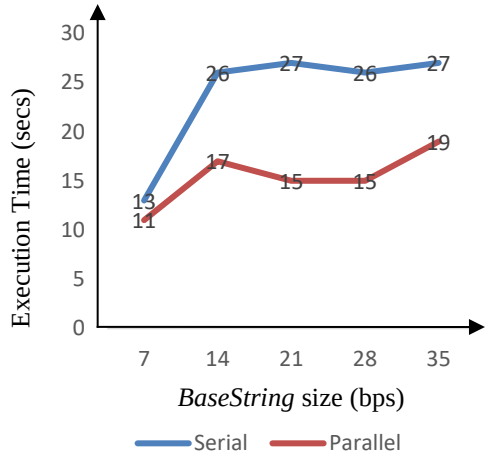

Figure.5 Adaptive hash algorithm's serial vs parallel computation time for searching patterns by varying base string size

The x-axis represents the number of patterns to be searched and the y-axis is the execution time of the algorithm. It is observed that parallel version of the algorithm finds the patterns much quicker when compared to the serial version of the algorithm. The graph in Fig. 5 also shows that even when *BaseString* is changed the parallel versions are able to search for the patterns quickly when compared to serial version.

The second set of results related to discovering new motifs by DE algorithm is taken by using subsequences ranging from length 5-40 bps and having maximum similarity. The results in Table 2 gives list of some newly discovered motifs of rice genome sequences. This information helps biologists to carry out their research in which motifs gets activated under different stress conditions.

Each individual in DE algorithm is denoted as a vector of locations of motif in each sequence. Two different methods are used to generate the initial population. One is the initialization method and second is completely random approach. The random approach explores the maximum solution space to generate initial population.

Table 2. List of discovered motifs

Discovered motifs	Total no. of occurrence
CCGGATAAC	80
CAACGGTC	1012
ATTCGGGC	298
CGTAACGT	202
AACGGTCG	580

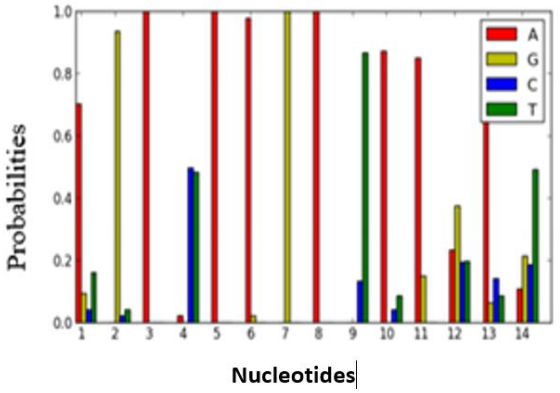

Figure.6 DE algorithm PWM presentation for file chr02.1kUpstream

But, this increases the convergence time. In initialization method, each individual is created by choosing a motif length *l*, obtaining its location through search algorithm and adding some random number α [1, 10] to that position to form new positions.Fig. 6 shows the probability of each nucleotide being present at each location in the motif identified as AGAAAGATAAGAT.

Further, to get the best DE configuration, experiments were conducted with different crossover factor values and the results are taken for 25 runs to make sure that the results are statistically correct. The population size is set as 200 individuals and generation limit as 3000 as per the findings from the literature. The threshold value is set to be 0.50 support for a motif from each sequence. The experiment was conducted with same mutation factors but with different crossover rates. Mutation factor F is set to be 0.5 and selection scheme used is best/1/binomial.

The motif which scored maximum is selected at the end of the process as it is found in maximum sequences and its similarity index is high and motif length is optimal too. To check on which crossover ratio will work best Table 3 provides the success rate of motifs found at rates ranging from 0.1 to 0.9. From Table 3 we conclude that the best results obtained in most of the cases is at crossover ratio (CR) set at 0.25.

Table 3. Crossover factor

	0.1	**0.25**	**0.75**	**0.9**
yst03r	66.80%	67.00%	67.35%	68.93%
mus07r	73.29%	73.81%	72.41%	71.25%
dm01r	75.09%	75.19%	75.26%	75.91%
chr02	71.19%	72.35%	71.44%	71.28%

Table 4. Similarity comparisons

		Similarity	
Dataset	**Motif length**	**DE**	**Genetic**
yst03r	10 15 8	0.85 0.86 0.96	0.82 0.84 0.84
chr02	8 12 15	0.85 0.87 0.89	0.82 0.79 0.80
hm03r	8 9 10	076 0.80 0.87	0.76 0.81 0.83

In Table 4, the performance of the algorithm in terms of similarity is shown for both genetic algorithm and DE algorithm when length of the motif is kept constant. The observation made is due to the tuning process of parameters after the cross over operation, the DE algorithm provides better population to find patterns which are most similar. Though the genetic algorithm works better in most optimization problem here DE outperformed due to several reasons. Firstly, unlike genetic algorithm, DE doesn't use any operators based on the representation of the solution, rather it evolves the set of the vectors during its own process itself. The other reason behind DE working well is its nature of stability whereas the GE suffers from premature convergence. The method of differential evolution can be applied to real-valued problems over a continuous space with much more ease than a genetic algorithm. The algorithm which is run on i5 processor with 2.33 GHz CPU and 4 GB RAM in Windows environment shows the execution time of DE with fine tuning is much less compared to general genetic algorithm. The strength of differential evolution's approach is that it often displays better results and can be easily applied to a wide variety of real valued problems despite noisy, multi-modal, multi-dimensional spaces, which usually make the problems very difficult for optimization.

7. Conclusion and future work

In the paper multiple string matching algorithm has been used for finding percentage of existing motifs that can be combined with the protein sequence to activate a gene. The paper also provides certain new motifs discovered by applying differential evolution algorithm on rice chromosomes. Using the percentage of motifs present in a particular chromosome the user will be able to find which gene can be activated using those motifs and mapping the motif to its transcription factor values will help in identifying what biological effects will occur if that motif is combined with particular gene. The experiment conducted show about approximately 30% reduction in time to search a particular motif using the parallel version of the algorithm. DE is also observed to converge faster and provide getter similarity index than genetic algorithm. As part of future work, we have the intention to compare the AHPS algorithm with other hash based algorithms and also DE with fine tuning with other evolutionary algorithms. Also, we would like to know the changes in behavior of the algorithm based on the selection schemes.

References

[1] L.A.E. Silva, "A Data Mining Approach for Standardization of Collectors Names in Herbarium Database", *IEEE Latin America Transactions*, Vol.14, No. 2, pp.805-810, 2016.

[2] M. Alicherry, M. Muthuprasanna, and V. Kumar, "High speed pattern matching for network IDS/IPS", In: *Proc. of 14th IEEE International Conf. On Network Protocols*, California, USA, pp.187-196, 2006.

[3] P. Kanuga, and A. Chauhan, "Adaptive hashing based multiple variable length pattern search algorithm for large data sets", In: *Proc. of International Conf. On Data Science & Engineering*, Kochi, India, pp.130-135, 2014.

[4] D. L.González-Álvarez, M. A. Vega-Rodríguez, and A. Rubio-Largo, "Finding patterns in protein sequences by using a hybrid multiobjective teaching learning based optimization algorithm", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, Vol.12, No.3, pp.656-666, 2015.

[5] R. Shamloo-Dashtpagerdi, H. Razi, M. Aliakbari, A. Lindlöf, M. Ebrahimi, and E. Ebrahimie, "A novel pairwise comparison method for in silico discovery of statistically significant cis-regulatory elements in eukaryotic promoter regions: Application to Arabidopsis", *Journal of Theoretical Biology*, Vol.364, No.1, pp.364-376, 2015.

[6] G.H. Gonnet, and R.A. Baeza-Yates, "An analysis of the Karp-Rabin string matching algorithm", *Journal of Information Processing Letters*, Vol.34, No. 5, pp. 271-274, 1990.

[7] D.M. Sunday, "A very fast substring search algorithm", *Communications of the ACM*, Vol. 33, No. 8, pp. 132-142, 1990.

[8] K. Al-Khamaiseh, and S. ALShagarin, "A Survey of String Matching Algorithms", *International Journal of*

Engineering Research and Applications, Vol.1, No. 4, pp.144-156, 2014.

[9] R.A. Baeza-Yates, and M. Régnier, "Average running time of the Boyer-Moore-Horspool algorithm", *Journal of Theoretical Computer Science*, Vol. 92, No. 1, pp.19-31, 1992.

[10] L. Bergroth, H. Hakonen, and T. Raita, "A survey of longest common subsequence algorithms", In: *Proc of IEEE Seventh International Symposium on String Processing and Information Retrieval*, Coruna, Spain, pp. 39-48, 2000.

[11] D. Cantone, and S. Faro, "Fast-Search: A new efficient variant of the Boyer-Moore string matching algorithm", In: *Proc of International Workshop on Experimental and Efficient Algorithms, Springer*, Berlin, Heidelberg, pp.47-58, 2003.

[12] R.S. Boyer, and J.S. Moore, *A Computational Logic Handbook: Formerly Notes and Reports in Computer Science and Applied Mathematics*, Vol.1, Academic Press, New York, N.Y.1988.

[13] J.M. Carlson, A. Chakravarty, C.E. DeZiel, and R.H. Gross, "SCOPE: a web server for practical de novo motif discovery", *Journal of Nucleic Acids Research*, Vol.35, No.2, pp.W259-W264, 2007.

[14] M. Nicolae, and S. Rajasekaran. "Efficient sequential and parallel algorithms for planted motif search", *BMC bioinformaticsi*, Vol.15, No. 1, pp.15-34, 2014.

[15] M. Nicolae, and S. Rajasekaran, "qPMS9: an efficient algorithm for quorum planted motif search", *Scientific reports*, Vol. 5, pp.7813, 2015.

[16] C. Jia, M. B. Carson, Y. Wang, Y. Lin, and H. Lu, "A new exhaustive method and strategy for finding motifs in ChIP-enriched regions", *PLoS One*, Vol. 9, No. 1, pp.e86044, 2014.

[17] Q. Yu, H. Huo, X. Chen, H. Guo, J.S. Vitter, and J. Huan, "An efficient algorithm for discovering motifs in large DNA data sets", *IEEE transactions on nanobioscience*, Vol. 14, No. 5, pp.535-544, 2015.

[18] L. Salmela, J. Tarhio, and J. Kytöjoki, "Multipattern string matching with q-grams", *Journal of Experimental Algorithmics*, Vol.11, pp.1-1, 2007.

[19] S. Faro, and T. Lecroq, "Efficient variants of the backward-oracle-matching algorithm, *International Journal of Foundations of Computer Science*, Vol. 20, No.6, pp.967-984, 2009.

[20] X. Ke, and C. Yong, "An improved Wu-Manber multiple patterns matching algorithm", In: *Proc. of 25th IEEE International Conf. On Performance, Computing, and Communications*, Phoenix, Arizona, pp.6, 2006.

[21] N.T.L Tran, and C.H. Huang, "A survey of motif finding Web tools for detecting binding site motifs in ChIP-Seq data", *Journal of Biology* Direct, Vol.9, No. 1, pp.4, 2014.

[22] Y. Fan, W. Wu, R. Liu, and W. Yang, "An Iterative Algorithm for Motif Discovery", *Procedia Computer Science*, Vol. 24, pp.25-29, 2013.

[23] M. Manevitz, and M. Samson, "De-Novo motif finding using genetic algorithm", In: *Proc of IEEE 28th Convention of Electrical & Electronics Engineers*, Eilat, Israel, pp.1-5, 2014.

[24] J. Serrà, and J.L. Arcos, "Particle swarm optimization for time series motif discovery", *Journal of Knowledge-Based Systems*, Vol.92, pp.127-137, 2016.

[25] M. Ramadas, A. Abraham, and S. Kumar, "ReDE-A revised mutation strategy for differential evolution algortihm *International Journal of Intelligent Engineering and Systems*, Vol. 9, No. 4, pp.51-58,2016.

[26]G. Balaji, R. Balamurugan, and L. Lakshminarasimman, "Fuzzy clustered multi objective differential evolution for thermal generator maintenance scheduling" , *International Journal of Intelligent Engineering and Systems*, Vol. 9, No. 1, pp.1-13, 2016.

[27] M. Zainudin, M. Sulaiman, N. Mustapha, T. Perumal, A. Nazri, R. Mohamed, and S. Manaf, "Feature Selection Optimization using Hybrid Relief-f with Self-adaptive Differential Evolution", *International Journal of Intelligent Engineering and Systems*, Vol. 10, No. 3, pp.21-29, 2017.

[28] A. M. Singh, and Singh Jatinder, "Hybrid Optimization Algorithm for Community and Fraud Detection in Complex Networks for High Immunity Towards Link and Node Failures", *International Journal of Intelligent Engineering and Systems*, Vol. 11, No.1, pp.211-220, 2018

[29] L. Shao, Y. Chen, and A. Abraham, "Motif discovery using evolutionary algorithms." In: *Proc. of International Conf. of Soft Computing and Pattern Recognition*, Malacca, Malaysia, pp.420-425, 2009.