Proc. of Int. Conf. on Advances in Computer Science, AETACS

ELSEVIER

Parallel Implementation of Exact algorithm for Planted (l,d) Motif Search

Satrupa Mohantyr[1], Prof Biswajit Sahu[1], and Anuja Kumar Acharya[1]
[1] School Of Computer Engineering,KIIT University, Bhubaneswar
Satarupamohanty2006@gmail.com
sh_biswajit@yahoo.co.in

***Abstract*—Identifying meaningful patterns known as motif from voluminous amount of biological data is a big challenge. Three versions of motif search problem have been identified in the literature. One of this is planted (l,d) motif problem. Numerous algorithms have been proposed in the literature that addresses that challenge. Many of these algorithm fall under the category of approximation algorithm. In this paper, we present algorithms for planted (l,d) motif problems that always find the correct answer. The algorithm proposed here is parallelization of PMSP Rajsekharan et.al. [3]. Our algorithm is very simple and is based on ideas that are fundamentally different from this one employed in literature. Most of the sequential motif finding algorithms contains lots of repeated, data-independent operations. In this paper, we propose an improvement over the existing exact planted (l, d) motif search algorithm using a bit-vector mapping technique to avoid storing of huge number of l -mers generated by different sequences. We also develop a parallel algorithm for this modified sequential algorithm using both coarse grained and fine grained parallelism. Synthetic data sets from the public domain are used to implement the algorithm, for analyzing the efficiency and accuracy of the proposed solution. We observe that our parallelization method on four SMP cluster systems with each of 2.4GHz Intel Pentium-IV having 4GB RAM running under Red Hat Linux improve the existing sequential programs.**

***Index Terms*—Planted Motif Search (Pmsp), Symmetric Multiprocessor (Smp), Message Passing Interface (Mpi), Bit Map Vector**

I. INTRODUCTION

A gene is a segment of DNA that is developed to produce different proteins to provide a particular function. In order to start the decoding process (gene expression), a molecule called transcription factor will bind to a short region (binding site) preceding the gene. One kind of transcription factor can bind to the binding sites of several genes to cause these genes to co-express. These binding sites have similar patterns called motifs. Motif may occur repeatedly either within the same molecule or in the many molecules. Motifs are short; typically 5 to 20 base pairs long [12] repeated patterns that occur either within the same molecule, or in many molecules [13]. Examples of motifs are: regulatory motifs, restriction recognition sites, splice sites, promoter regions, potential drug target sites etc.
Suppose a secret pattern M of length l is implanted at random positions of given set of DNA sequences. Without knowing what the pattern M is, or where in each sequence it has been implanted, we have to

reconstruct M by analyzing the DNA sequence. It is assumed that each input sequence contains a variant of M. The variants of interest are sequences that are at a hamming distance [7] of at most d from M.

Given a set S of t sequences S = {S1,S2,…,St} of each length n, from a fixed alphabet $\sum$, and integers l, and d, with $0 \leq d < l < n$, we define the (l, d) motif search problem as that of finding a string x with length l such that Si has a substring xi of length l such that x differs from xi in at most d places for i = 1,…, n. We will call x a planted (l, d) motif for S. Searching planted motif problem is known to be NP-Complete [14].

Three versions of motif search have been identified in the literature: planted (l,d) motif search (PMS), edit distance based motif search(EMS) and simple motif search (SMS). In this paper we focus on PMS.

Simple motifs search problem is to identify all the patterns in the given set of sequences such that each pattern is of length at most l, together with a count of how many times each pattern occurs. Optionally a threshold value for the number of occurrences could be supplied. Determining this threshold is a challenging task. A pattern is a string of symbols (also called residues). The length of a pattern is the number of characters in it . The edit distance based motif discovery problem is to find all the patterns in the given set of sequences such that each pattern is of length l and it occurs in at least q of the t sequences. If a pattern V is output, it will mean that V is a string of length l present in one of the input sequences and also V occurs in at least q of the input sequences. (A string x with $|x| = l$ is considered an occurrence of pattern V as long as the edit distance between x and V is at most d).The planted (l, d) -motif search problem is to find a motif (i.e., a sequence) M of length l in the given set of sequences. It is assumed that each input sequence contains a variant of M. The variants of interest are sequences that are at a hamming distance of d from M [7]. "The Challenge Problem" was parameterized as finding a planted (15, 4)-motif in t = 20 sequences each of length n = 600.

Despite extensive studies and research, the exact solution for motif finding problem leads to exponential algorithm. Thus the efficiency is low in serial implementation of the exact solution. Usually the sequential algorithms contain lots of repeated, data-independent operations e.g. the extraction of all possible l -mers (u) from string S1, generation of all relevant neighbors of these l-mers (v) at a hamming distance of d from each u, generation of all relevant neighbors of these l-mers (v') at a hamming distance of 2d from each u sort these v' to store in different sets and then to find each v which present in all set at hamming distance d[3]. On examination of the algorithm for PMS problem, it is observed that planted motif search problem developed by Rajsekharan[3] is one among the best of sequential algorithms. In this paper, we propose modification of existing exact PMSP algorithm developed by Rajsekharan[3] with the help of a bit-vector array to avoid storing huge number of l -mers (v) generated by different sequences and then parallelize the algorithm. We have used synthetic data sets from the public domain for implementing the algorithm, and analyzing the efficiency as well as accuracy of the proposed solution.

Next section presents works related to motif finding algorithms. Existing sequential exact PMSP algorithm is explained in section 3. Motivation in section 4.The proposed parallel algorithm along with its implementation details are presented in section 5. Performance evaluation, experimental results, and comparison with sequential algorithm are discussed in section 6. Section 7 concludes the paper.

II. Related Work

The existing motif discovery algorithms may be classified based on: consensus and alignment [15], the underlying biology [16], specific techniques [17], specific genomes [18], revealing the differences and similarities between existing methods ([12] [19]), using deterministic or statistical pattern models [2], using a class of grammars, and rating function (motif score) etc. [4]. Furthermore, methods also differ in aspects like average running time, need for manual parameter-tuning, exhaustiveness of results, general usability and so on. Individual methods may also perform better on one type of genomes compared to others, making it difficult to compare performance on a general scale. Given t sequences S1, S2, ..., St of average length n from a fixed alphabet $\sum$, and integers l, and d, the motif finding algorithms can be classified as planted (l, d) motif search problem, edit distance based motif discovery problem, and Simple motifs search problem based on the type of sequence information employed [19].

Algorithms for PMS can be categorized into two groups depending on the basic approach employed, namely, profile-based algorithms and pattern-based algorithms (see, e.g., Price et al.[12]). Profile-based algorithms predict the starting positions of the occurrences of the motif in each sequence. On the other hand, pattern-based algorithms predict the motif (as a sequence of residues) itself.

Several pattern-based algorithms are known. Examples include PROJECTION [5], MULTIPROFILER [9] and Pevzner, 2002), MITRA [6], and PatternBranching[12]. PatternBranching (due to Price, Ramabhadran and Pevzner [12]) starts with random seed strings and performs local searches starting from these seeds.
Examples of profile-based algorithms include CONSENSUS[7],, GibbsDNA[10], MEME [1], and ProfileBranching [12]. The performance of profile-based algorithms are specified with a measure called the "performance coefficient," which gives an indication of how many positions (for the motif occurrences) have been predicted correctly. For the (15, 4) challenge problem, these algorithms have the following performance coefficients (respectively): 0.2, 0.32, 0.14, and 0.57. The run times of these algorithms for this instance are: 40, 40, 5, and 80 (respectively, in seconds).
Numerous algorithms have been developed to solve PMS. Algorithms can be categorized into two, namely, exact algorithms and approximation algorithms. An exact algorithm (also called an exhaustive enumeration algorithm) always finds the correct answer(s). on the other hand, an approximation algorithm may not always output the correct answer. Some examples of approximation algorithms are Random Projection [5], Winnower [7], Pattern Branching [8] and exact algorithms are CENSUS [10], MITRA [6], PMS1[1], PMSP[9],and Voting [11].
Certain instances of PMS have been identified to be challenging instances. In particular, an (l, d)-motif instance is said to be challenging if the expected number of motifs that occur in the input by random chance is greater than or equal to one (when the input sequences themselves are generated randomly). For example, the instances (9,2), (11,3), (13,4), and (15,5) are examples of challenging instances. It is customary in the literature to show the performance of PMS algorithms only on challenging instances. It is always a challenge to solve as large a challenging instance of PMS as possible. The largest instance reported solved in the literature thus far is (19, 7) [9].Before presenting details on our algorithm we summarize some of the prior algorithms.
Pevzner and Sze [7] have proposed the WINNOWER algorithm. It generates a list of all the l-mers present in the database and constructs a graph out of them. Each l-mer represents a node and there is an edge between two nodes if the hamming distance between these two nodes is at most 2dand they belong to different sequences. The problem of finding motifs is then reduced to finding large cliques in this graph. The problem of finding a maximum clique in a graph is known to be NP-hard. The authors use some novel pruning techniques to find large cliques.
WINNOWER [7] uses a technique to generate Extendable Cliques. It is an iterative algorithm. The run time of this algorithm is O(N2d+1), where N=nm. SP-STAR uses a technique which eliminates more edges than WINNOWER and hence is faster. It uses less memory as well.
PatternBranching [8] performs a local search. It generates the neighbors of all l-mers present in the sequences and uses a scoring method to assign scores for these neighbors. There are n(m-l+1) l-mers present in the sequences. Each l-mer has $\binom{l}{d}$ 3^d neighbors. The scores of all the neighbors of all the l-mers are computed and the best scoring neighbors are identified. Since it searches through only a selected set of l-mers, it is very efficient. This algorithm takes a u, and then computes the best neighbor $u_1$of u. In the next step it identifies the best neighbor $u_2$of u_1 and so on till it computes u_d. The u_d values for all possible u are computed and then the best u_d value becomes the output. In every step it keeps only one value of the best neighbor but there could be a set of l-mers which could be the best neighbors for u.
MITRA [6] is based on WINNOWER [7]. It uses pair wise similarity information. It uses a mismatch tree data structure and splits the space of patterns into subspaces which start with a given prefix. Pruning is applied to each of these subspaces. MITRA performs very well in practice and is one of the best performing algorithms designed for Planted Motif Search Problem.
Several algorithms for PMS have been proposed in [1]. These are exact algorithms based on radix sorting. The first algorithm proposed was PMS1. PMS1 works as follows: It considers each l-mer in each input sequence and, for each such l-mer q, it constructs a list of neighbors (that is, l-mers) that are at a distance of d from q. Neighbor lists of the input sequences are then intersected using radix sort to identify the planted motif. This algorithm works well in practice for values of $d \leq 3$; however, as d increases, the memory requirement tends to be large. Another algorithm proposed is PMS2 [1], which is capable of solving challenging instances up to $d = 4$. For instance, it could not solve the Challenging instance (15, 5). The third algorithm proposed in [1] was PMS3 which was never implemented or investigated. In this paper we extend and test the performance of PMS3.
Two new algorithms PMSP and PMSi were proposed in [3] which were built upon PMS1 and perform a lot better than PMS1. PMSP is based on the idea of exploring the neighbor-hoods of the l-mers of the first

sequence and checking whether the elements of such neighborhoods are (l d) motifs. Even though the theoretical bound is worse than that of PMS, it is able to solve challenging instances such as (15, 5) in 35 minutes and (17, 6) in 12 hours with small requirements of memory.

III. EXISTING PMSP

PMSP follows the following simple idea: For every l-mer x in s1, it generates the set of neighbors of x and tries to guess if an l-mer y in that neighborhood is a motif by checking whether there are l-mers in si for i=2, ... ,n that are at distance ≤ d from it. This algorithm has been given in [1].However, this has been modified in a critical way in [3]. The key observation is to notice that here it is not needed to evaluate the distance in all l-mers of si but rather in the l-mers of si which are at distance ≤ 2d from x.
Before we describe PMSP [3], we consider the following definitions.

*Definition 1.*Given a set of strings $S = \{s_i\}_{i=1}^{t}$ over an alphabet $\sum$, with $|s_i|$ = m and l, d with 0≤ d < l < m, we define the (l,d) motif search problem as that of finding a string x with $|x|$ =l such that s_i has a substring x_i of length l such that x differs from x_i in at most d places for i = 1,...,n.

Definition 2. Given a string s with $|S|$=m and a string x with $|x|$ = l with l<m. We say $x \lhd_l s$ if x is a subsequence of s. Equivalently, we say that x is an l-mer of s.

Definition 3. For any string x, with $|x|$ = l, let $B_d(x) := \{y: |y| = l \text{ and } d_H(y, x) \le d\}$, where d_H is the "Hamming" distance, that is, the number of places where the two strings differ.

Notice that $N(l,d) = \sum_{i=1}^{d} \binom{l}{i} (|\sum| - 1)^d = |B_d(x)|$

Definition 4. Given s, with $|s| = m$ and $0 \le d < l \le m$, let $L_s := \bigcup_{x \lhd_l s} B_d(x)$

Algorithm PMSP.

1) Let M = Ø.
2) For each $x \lhd_l s_1$:
   - 1.1 Let N(i) = $\{y \lhd s_i : d_H(x,y) \le 2d\}$ for i=2,… ,n.
   - 1.2 For each x' Є $B_d(x)$:
     - 1.2.1 Check for **every** i($2 \le i \le n$), there exists a y_i Є N(i) such that $d_H(x', y_i) \le d$.
     - 1.2.2 In the affirmative, add x' to M.
3) Output M.

A. Limitation.

This algorithm has been applied to 20 numbers of sequences to get the result in reasonable time for motif of size (11,3), (13,4), (15,5), (17,6), (19,7). Finding planted motifs for larger Hamming distance i.e for 'd' and longer length of sequence becomes impractical because, huge number of l-mers will be generated at 2d distance for all string S_i (i=2 to n) corresponding to each l-mer of string S_1. In this simple PMSP algorithm the space for storing huge will be very high and searching, storing this huge l-mer will take more space and time.This also generates large number of l-mers at distance d for each l-mer of string S_1.Hence time for a sequential searching is increasing frequently for larger d. This searching time has been reduced by using a parallel searching. Improvement of space required for huge l-mers is done using bit vector technique. Additionally a larger memory is also available by multiprogramming environment to store these huse l-mers.

IV. MOTIVATION

Despite extensive studies and research, the motif finding problem still remains a challenge. We performed an exhaustive survey, their biological significance, and relevant computational methods for finding planted (l, d) motifs. The efficiency is low in serial implementation of the exact solution by single computer. In most of the sequential algorithm contains lots of repeated, data-independent operations. . This motivated us to propose a

parallel way of implementing existing exact planted (l, d) motif search algorithm developed by Rajsekharan et al. [3] by modifying the sequential algorithm. For this work we use a bit-vector mapping technique to avoid maintaining sets N(i) of different length for l-mers which are at 2d distance in each sequence w.r.t to each l-mer of first sequence and also for l -mers (v) generated from l-mer of first sequences. We have used synthetic data sets from the public domain to implement the algorithm, for analyzing the efficiency and accuracy of the proposed solution.

V. IMPROVED ALGORITHM BASED ON PMSP

1) Let M = Ø.
2) For each $x_j \lhd_l s_1$, do parallel using process where j=1 to n-l+1

Par Begin

2.1 For each i=2 to t do parallel using thread

Find N(i) = {y $\lhd s_i : d_H(x,y) \leq 2d$ }

2.2 Find $B_d(x_j)$

Par End

2.3 For each x' Є $B_d(x_j)$ do parallel using thread

2.3.1 Check for **every** i($2 \leq i \leq n$), there exists a y_i Є N(i) such that $d_H(x', y_i) \leq d$.

2.3.2 In the affirmative, add x' to M_{th}.

2.4 $M_p = \bigcap M_{th}$.

3) $M = \bigcup M_p$.

A. Implementation of Proposed Algorithm

For this algorithm, a Client-Server (C/S) mode is applied to our LAN, i.e. one Server and multiple clients. The Server is responsible for the coordination of the clients, which distributes the PMS tasks to these clients and make sure of their synchronization as shown in the Figure 1.

The clients accept the task and implement it in parallel. We have used SSh key technique to avoid the requirement of password during the interaction among master system and clients (slave) system for the execution of program. This helps easy and fast communication between systems.

All given sequences (S2, …, St) are evenly distributed among computers used in the distributed system along with the sequence S1. Each process will determines possible planted (l,d) motifs from the allotted subset independently. We represent every l-mer, as a sequence of integers. If r1,r2, …,rl is an l-mer, it is represented as (i1, i2,…, iq), where i1,i2,…, iq are integers, each integer corresponding to a sequence of residues. When $|\sum| = 4$, we need two bits per residue. Thus a sequence of residues can be represented as an integer in usual way. For instance, when l=15, if the size of an integer(on the machine of interest) is 32bits, then each l-mer is represented as an integer.

In our algorithm we have used bit vector instead of list. For every possible l-mer of length l, there is a corresponding bit in the bit vector. The bit represents whether the l-mer is present (1) or not (0). For fixed length l, increasing d, gives rise to increase the list tremendously in existing algorithm [3], where as in our proposed algorithm the bit vector size will remain same irrespective of size of d. As all possible l-mer of length 15 requires $\frac{1}{8}$ GB of memory space to represent the whole bit vector. All l-mers generated by S1 is divided into number of subsets where each subset is allocated to each process. Now each process will generate N(i), where i = 2 to (t-1)/2 and i = ((t-1)/2)+1 to t to store l-mers at a distances 2d of corresponding Si respectively. At the same time third thread finds the all possible l-mers x' Є Bd(x), where x is the l-mer of S1. After completion of these three threads, then for each x' Є Bd(x) we find Yi parallel using two threads such that Yi Є N(i) for i=2 to t and dH(x', Yi) $\leq$ d. we store those x' in a bit vector M_k^t. To find all x' which are motifs in each process we use a bit wise AND operation between all M_k^t for k=1 to number of thread and store the result in M p bi t vector. Finally a bitwise OR operation among all bit vector Mp(from all

process) are computed and the combined final bit vector is found which contain all motifs generated from the subset of l-mers of S1 in each process.

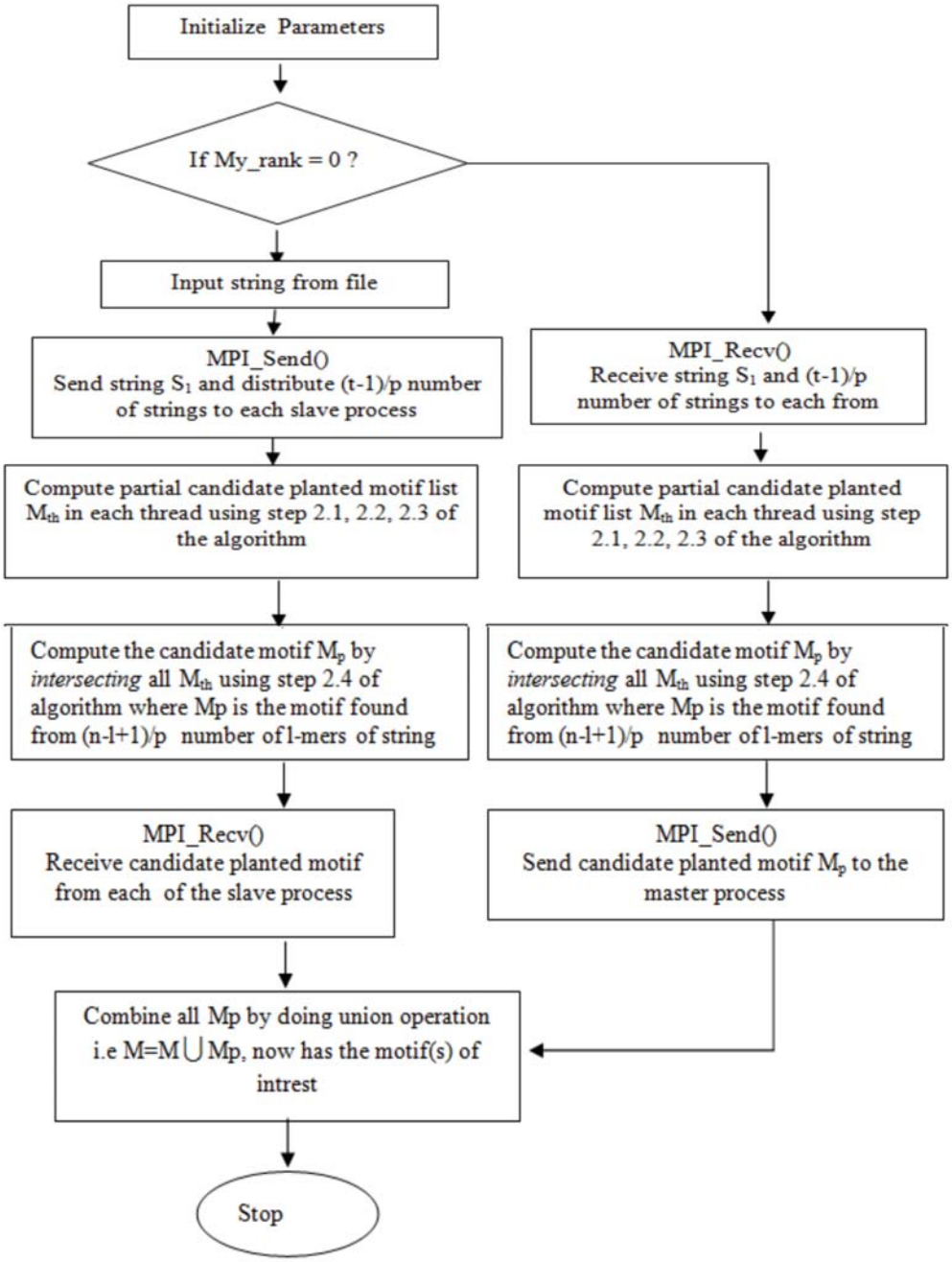

Figure 1: Flow chart of all message passing among master and slaves

B. Server Designing

The server is designed so as to perform the following functions:
Task distribution: In order to make the clients to work simultaneously, the server equally divides the task into smaller subtasks based on the number of files or sequences and distributes to the clients to compute. The clients upload the processing results.

Database sharing: We have primary database on the server. It stores all the related gene and protein information. Synchronization control: Because of the serial relationship between the modules, the server must control the synchronization of the clients. When the server distributes the tasks, it will send a Start message to the clients so that they can start to work. If one of the clients completes the work, it will send an End message along with the result to the server.
Planted (l, d) motif: After receiving End messages from all the clients, the server begins to calculate the planted (l, d) motif. The server is configured with REDHAT Enterprise version-5 OS.

C. Client Designing

Under control of the server, the clients simultaneously complete the task of "Getting DNA sequence into Database", and computing "Motif Finding" from a subsets of sequences. We have secondary database on the clients. When receiving the tasks from the server, the clients download the information from the server and store on the secondary database, which will avoid time delay due to frequently accessing the primary database. When the current task is completed, the clients need to submit the processing results and send an End message to the server. The clients are configured with REDHAT Enterprise version-5 OS.

VI. PERFORMANCE ANALYSIS

We implemented the proposed algorithm using C, MPI library, Pthread library. Our experiment environment is a four nodes cluster with each of 2.4GHz Intel Pentium-IV processor having 2GB RAM running under Red Hat Linux. Nodes are interconnected with one gigabits/s Ethernet switches. We tested our parallel algorithm using all four nodes and measured the wall clock time between the start and the end of the algorithm as the running time. The running time includes the execution time, the communication delay and reading the input data from a file. The algorithm allocates the sequence pairs based on the number of symm6etric multiprocessor (SMP) nodes available. Because the system is implemented under LAN, the transfer distance between the server and the clients is very short, and may be ignored.
Synchronization delay (sd):- the server distributes the tasks based on the number of files or sequences, so the sizes of the subtasks are almost equal which results in the same completion time of the clients. Because the server has to wait for the End messages from all of the clients, 'sd' is the main factor to affect the accelerating rate.
Data transfer delay (dt): -the clients download data from the server and upload data to the server. Because the size of data is large, 'dt' is also an important factor. Analysis shows that "Synchronization delay" and "Data transfer delay" are the main factors to affect the processing speed. The running times of PMSP, the best sequential algorithm (SA) proposed by Rajasekharan et al. [3], and our parallel algorithm (PA) in Table 1.

TABLE 1: THE RUNNING TIME OF BOTH PMSP SEQUENTIAL (SA) AND OUR PARALLEL ALGORITHM(PA)ON A ON A TWO NODE CLUSTER FOR N=600, AND L=20

Length of the motif (*l*)	Hamming distance (*d*)	*Running time for SA (Sec)*	Running time for parallel algorithm	
			One CPU (*Sec*)	Two CPU (*Sec*)
11	3	*6.9*	4.2	3.1
11	4	-	4.9	3.6
11	5	-	6.2	4.1
13	4	*152*	86	63
13	5	-	104	78
13	6	-	156	123
15	5	*2100*	1092	666
15	6	-	1278	762
15	7	-	1320	852

A. Experimental Results and Discussion

We discuss below the running time of planted (l, d) motif algorithm obtained from data set having variable motif length and variable number of Hamming distance. Our parallel algorithm is executed by creating two numbers of processes on a two node cluster. This distributes one process to each node. In this experiment, motif lengths are (11,3), (13, 4), and(15,5) for 20 number of sequences of length 600 each. The first set of

experiments are aimed at observing how lengths of the planted motif affect the performance of our parallel algorithm shown in Figure 3 with respect to existing PMSP sequential algorithm [3] as shown in Figure 2. It is observed in all cases that correct planted motifs are found using proposed parallel algorithm successfully.

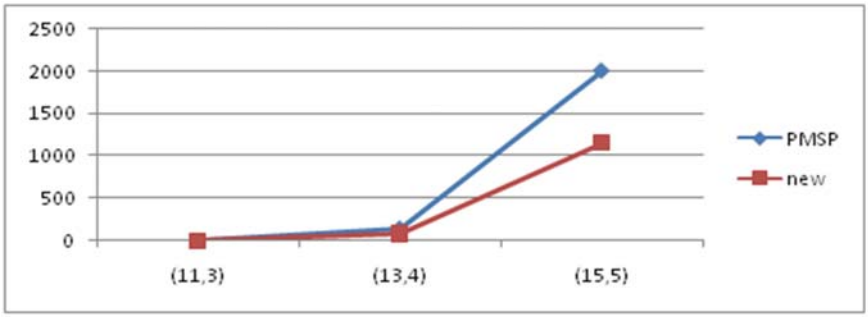

Figure 2: Running time of our parallel PMSP algorithm and existing sequential algorithm (SA) for t= 20, and n= 600 as a function of (l,d) motifs for different values of l and d.

The next experiment investigates how the processing time is affected by varying Hamming distance for length of motif 11, 13, 15 as shown in figure 4, 5, and 6 respectively. Running time of parallel algorithm on a two node cluster is compared with two nodes and one node, with the increasing Hamming distance. Curves indicate that the processing time has linear growth as the number of processors increase.

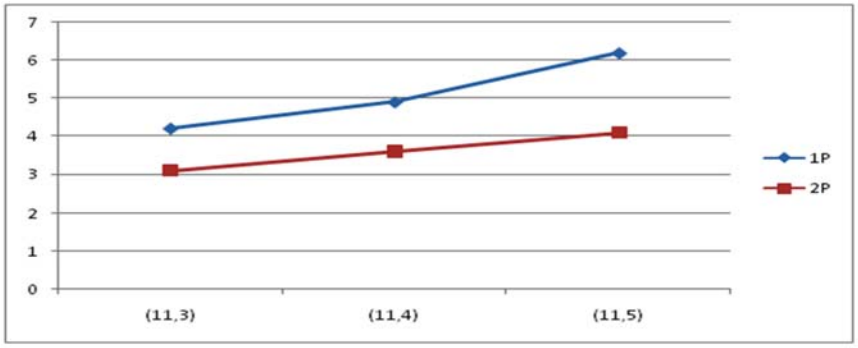

Figure 3: Running time of our parallel PMSP algorithm (1P, 2P) for t= 20, n= 600 and l= 11 as a function of Hamming distance of motifs.

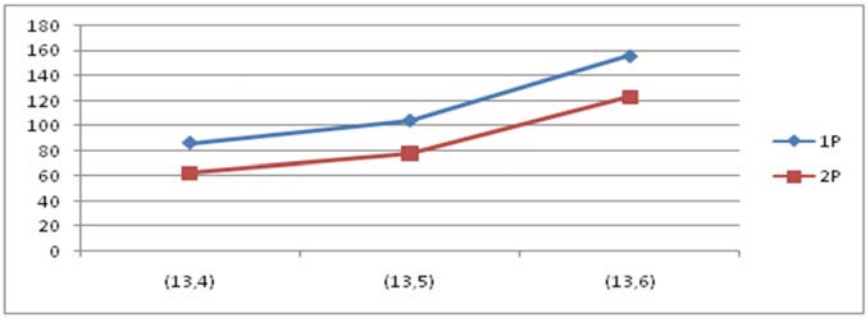

Figure 4: Running time of our parallel PMSP algorithm (1P, 2P) for t= 20, and n= 600 as a function of (l,d) motifs for different values of l and d.

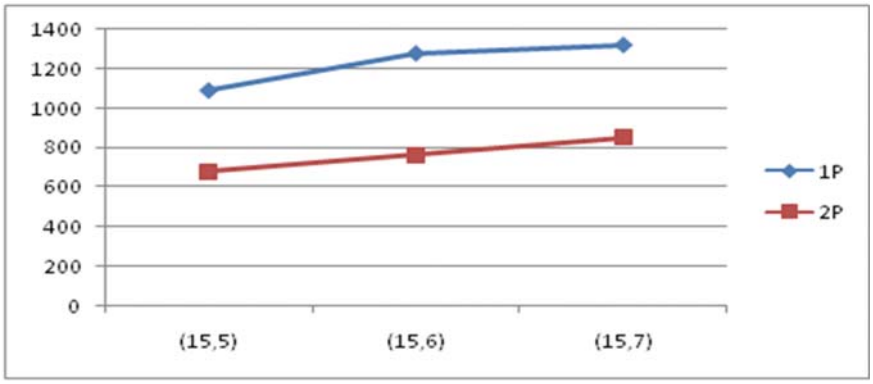

Figure 5: Running time of our parallel PMSP algorithm (1P, 2P) for t=20, and n=600 as a function of (l,d) motifs for different values of l and d.

VII. Conclusion

High performance computing is the challenging and important issue in computational biology. We have designed, implemented and analyzed our proposed algorithm for planted motif search problem on this computing environment. Firstly, we use a bit vector mapping technique in the sequential PMS3[3] algorithm to minimize storage of huge number of l-mers to be generated during intermediate processing. The bit vector implementation encourages the result for motifs of higher length (i.e $13 < l < 17$) with larger Hamming distance (d). Secondly, we also propose parallel algorithm which distributes the tasks of finding planted (l,d) motif from set of sequences evenly among all the processors in the cluster. Use of both coarse and fine grain parallelism in the proposed algorithm justifies our claim that our proposed parallelization method on SMP cluster system improves over existing sequential algorithm. We implement the proposed algorithm on two nodes SMP cluster system with each of 2.4GHz Intel Pentium-IV having 4 GB RAM running under Red Hat Linux. Nodes are interconnected with one gigabits Ethernet switches. The algorithm is also scalable i.e. by increasing the number of processors and number of sequences simultaneously maintains the speed up. Use of bit vector for storage and parallelization (both coarse grain and fine grain) makes it space optimal and cost optimal.

References

[1] S. Rajasekaran, S. Balla, and C.-H. Huang, "Exact Algorithms for Planted Motif Problems", Journal of Computational Biology, Oct 2005, Vol. 12, No. 8: 1117-1128.

[2] Brazrna, A., Jonassen, I., Eidharnrner, I.,and Gilbert, D., "Approaches to the automatic discovery of patterns in biosequences", J Comput Biol, 5(2), pp279–305, 1998.

[3] Jaime Davila, Sudha Balla, Sanguthevar Rajasekaran, "Fast and Practical Algorithms For Planted (l, d) Motif Search", IEEE/ACM Transactions TCBB, 2007.

[4] Brazrna, A., Jonassen, I., Vilo, J.,and Ukkonen, E., "Pattern Discovery in Biosequences", in Proc. 4th International Colloquium on Grammatical Inference London, Springer- Verlag, pp257–270, 1998.

[5] J. Buhler and M. Tompa, "Finding motifs using random projections", Proceedings Fifth Annual International Conference on Computational Molecular Biology (RECOMB), April 2001.

[6] E. Eskin and P. Pevzner, "Finding composite regulatory patterns in DNA sequences", Bioinformatics S1, 2002, pp. 354-363.

[7] P. Pevzner and S.-H. Sze, "Combinatorial approaches to finding subtle signals in DNA sequences", Proc. Eighth International Conference on Intelligent Systems for Molecular Biology, 2000, pp. 269-278.

[8] A. Price, S. Ramabhadran, P. Pevzner, "Finding subtle motifs by branching from sample strings", Bioinformatics supplementary edition, Proceedings of the Second European Conference on Computational Biology (ECCB-2003).

[9] J. Davila, S. Balla and S. Rajasekaran, "Space And Time Efficient Algorithms for Planted Motif Search", Second International Workshop on Bioinformatics Research and Applications (IWBRA 2006), May 2006.

[10] M. Li, B. Ma and L. Wang,"On the Closest String and Substring Problems", Journal of the ACM, Vol. 49, No. 2, March 2002.

[11] F Y.L. Chin and H C.M. Leung, "Voting Algorithms for Discovering Long Motifs", Proceedings of the Third Asia-Pacific Bioinformatics Conference (APBC2005), Singapore, 261-271, January 2005

[12] Das, M. K., and Dai, H. K., "A survey of DNA motif finding algorithms", BMC Bio informatics, vol. 8 Suppl 7, p. S21, 2007.

[13] Stormo, G.D., “DNA binding sites: representation and discovery”, Bioinformatics, pp16–23, 2000.
[14] Evans, P.A, Smith, AD., Wareham, H.T., “On the complexity of finding common approximate substrings”,Theoretical Computer Science,pp407–430, 2003.
[15] Ukkonen, E., “Finding approximate patterns in strings”, Journal of Algorithms6, pp 132–137,1985.
[16] Wasserman, W.W., Krivan, W., “In silico identification of metazoan transcriptional regulatory regions”, Naturwissenschaften,90(4), pp156–166,2003.
[17] Bulyk, M.L., “Computational prediction of transcription-factor binding site locations”, Genome Bioi, 5, pp201–211, 2003.
[18] Hannenhalli, S.,and Levy, S., “Promoter prediction in the human genome”, Bioinformatics, 17(Suppl 1), pp90–96, 2001.
[19] Sanguthevar, R., “Algorithms for Motif Search”, CRC Press, 2001