CSIT (December 2019) 7(4):295–297
https://doi.org/10.1007/s40012-019-00235-w

ORIGINAL RESEARCH

High performance computing approach for DNA motif discovery

Deepti D. Shrimankar[1]

Received: 28 November 2018 / Accepted: 18 May 2019 / Published online: 17 August 2019

Abstract Unraveling the mechanisms that regulate gene expression is a major challenge in biology. An important task in this challenge is to identify regulatory elements, especially the binding sites in deoxyribonucleic acid (DNA) for transcription factors. These binding sites are short DNA segments that are called motifs. The motifs are short, recurring patterns in DNA sequences that are presumed to have a biological function. Motif discovery has been one of the most widely studied problems in bioinformatics ever since genomic sequences have been available. Recent advances in genome sequence availability and in high throughput gene expression analysis technologies have allowed for the development of computational methods for motif discovery. As a result, a large number of motif finding algorithms have been implemented and applied to various motif models over the past decade. Since regulatory elements are frequently short and variable, their identification and discovery using computational algorithms is difficult. However, significant advances have been made in the computational methods for modeling and detection of DNA regulatory elements. The detection of regulatory elements from a large set of regulatory regions is a challenging problem in computational genomics. However, computational methods to extract this biological meaningful information suffer from high computational requirements. High performance computing appears as a magic bullet in this challenge. Designing a parallel algorithm to detect regulatory elements using correlation with gene expression data and its implementation with openMPI and openMP will leads to significant runtime savings on distributed system. Solving computationally intensive problems on high performance computing architecture can significantly improve and speedup the run time of the problem solution when proper task distribution, scheduling strategy and suitable parallel computing paradigms are used. Deploying more and more cluster computers can bridge the gap of speed difference between architectures and will result in fewer numbers of concurrent jobs that can be allocated to the system.

✉ Deepti D. Shrimankar
dshrimankar@cse.vnit.ac.in

[1] Department of Computer Science and Engineering, VNIT, Nagpur, India

1 State-of-the-art

Motif finding is a well-studied problem in computing. Various motif search algorithms have been developed, falling into two main categories: heuristic and exact. Heuristic algorithms perform an iterative local search, for instance by repeatedly refining an input sampling or projection until a motif is found. Gibbs sampling and Expectation Maximization (EM), used in the motif-finding tool MEME both use probabilistic computations to optimize an initial random alignment. [An alignment is simply a vector (a_1, a_2,…,a_n) of n positions, which predicts that the motif occurs at position a_i in the given sequence S_i.] Gibbs sampling tries to refine the alignment one position at a time; in contrast, EM may recompute the entire alignment in a single iteration. Projection combines a pattern-based approach with EM's probabilistic approach, trying to guess every successive character of a tentative motif and using EM to verify its guesses. GARPS uses a random version of projection, in tandem with the genetic algorithm (GA), for yet another iterative approach. These are just some of many successful heuristic algorithms [1, 2]. However, heuristics are non-exhaustive, and thus not always guaranteed to find a solution. Exact algorithms, on the other hand, perform an

exhaustive search of possible motifs and so always find the planted motif.

WINNOWER and its successor MITRA are exact algorithms that look at pairwise *l*-mer similarity to find motifs. In a set of DNA sequences, there are numerous pairs of "similar" *l*-mers, which come from different sequences and have Hamming distances of at most *2d* from each other (meaning that they could be two d neighbors of the same *l*-mer). WINNOWER represents these pairs in a graph, with *l*-mers as nodes and edges connecting *l*-mer pairs. It then prunes the graph to identify "cliques" of pairs that indicate a motif. MITRA refines this graph representation into a mismatch tree containing all possible l-mers, organized by prefix [3, 4]. The tree structure allows MITRA to eliminate entire branches at a time, making it faster than WINNOWER at removing the spurious edges that are not part of any motif clique.

The current state-of-the-art in exact motif search is qPMS9, the most recent in a series of Planted Motif Search algorithms. It performs a sample-driven step, which generates a *k*-tuple of *l*-mers from each of *k* input strings, followed by a pattern-driven step, which generates the common d-neighborhood of the tuple and then checks whether any of the *l*-mers in this common neighborhood is a motif. To identify neighbors, qPMS9 efficiently traverses the tree of all possible *l*-mers, using certain pruning criteria explored by predecessors PMSPrune and qPMS7 to quickly discard non-neighbor branches. Sampling in qPMS9 is an improvement on its predecessor PMS8; in building a *k*-tuple, qPMS9 intelligently prioritizes *l*-mers that have fewer matches with the *l*-mers already selected, such that the common d-neighborhood becomes smaller and thus faster to check through. Finally, PMS8 and qPMS9 have been implemented to run on multiple processors, allowing them to solve instances with (*l, d*) as large as (50, 21) in a few hours.

2 Work done till now

We have proposed and implemented the distributed parallel computing algorithm for motif discovery problem. We have implemented a simple scalable and efficient parallel openMP and openMPI implementation for Planted Motif Search problem using cluster computer. Also we have presented the method for creating Beowulf cluster [5, 6]. The efficiency of the algorithm is validated by testing it on both simulated as well as real biological databases.

2.1 Experimental result on simulated data set

The input sequences are generated by using simulated data sets with parameters t = 20 sequences and m = 600 characters, where the characters are A, C, G, T. Each (*l, d*) input instance dataset is generated as follows: We generate random strings with length (m − l) each, where the characters appear randomly with equal probability [7, 8]. Then we generate randomly an *l* length string M and plant a copy of it in each sequence at random position after mutating it with at most *d* random mutations (Fig. 1; Table 1).

Figure 2 shows the scalability results for our algorithm where

$$speedup = \frac{Time\ on\ 1\ node}{Time\ on\ C\ nodes}$$

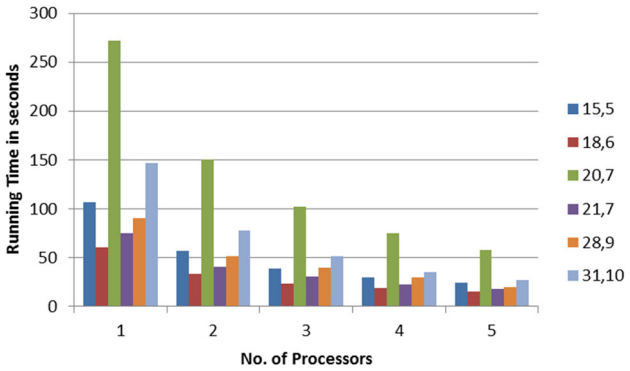

Fig. 1 Column chart for some instances of simulated data set

Table 1 Running time in seconds for different challenging instances [9, 10]

Instances	C				
	1	2	3	4	5
(15, 5)	106.9	56.51	38.60	29.64	24.14
(18, 6)	60.75	33.55	23.54	18.35	15.4
(21, 7)	271.75	150.6	102.3	75.2	57.78
(23, 8)	75	40.5	30.6	22.5	18.2
(26, 9)	950	449.2	330.8	255.1	194.9
(28, 10)	90.93	51.6	40.1	30.5	20.92
(30, 11)	147.47	78.4	51.6	35.1	27.62

C, number of computers in cluster

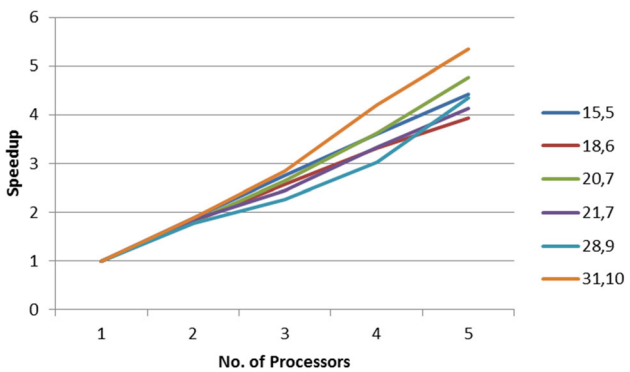

Fig. 2 Scalability plot of our algorithm for some instances

Table 2 Motifs detected in real biological dataset [9, 10]

Gene family	Instance (*l, d*)	Detected motif	Published motif
Insulin	(10, 1)	CCTCAGCCCC	CCTCAGCCCC [32]
Growth hormone	(9, 2)	TATAAAAAG	TATAAAAAG [32]
c-fos	(10, 2)	CACAGGATGT	CACAGGATGT [32]
PDR3	(8, 1)	TCCGTGGA	TCCGTGGA [32]
Histone H1	(10, 1)	AAACAAAAGT	AAACAAAAGT [32]
ECB	(16, 3)	TTTCCCATTAAGGAAA	TTtCCcnntnaGGAAA [32]
c-myc	(8, 1)	GTTTATTC	GTTTATTC [32]

We note that, our proposed method reduces the running time and the speedup achieved scales well with the increasing in number of cluster nodes.

2.2 Experimental result on real data set

We test PMS on a set of real biological data which are used in the literature. The data for this set contains the upstream DNA regions of a set of genes from different species (Table 2).

3 Future work

Solving computationally intensive problems on high performance computing architecture can significantly improve and speedup the run time of the problem solution when proper task distribution, scheduling strategy and suitable parallel computing paradigms are used. Deploying more and more cluster computers can bridge the gap of speed difference between architectures and will result in fewer numbers of concurrent jobs that can be allocated to the system. Future work may include the use of different scheduling strategies and intelligent selection criteria to choose the best scheduling strategy to solve a given computationally intensive problem. We believe that this paper is a step towards a complete system to solve computationally intensive problems on heterogeneous architectures.

3.1 Research papers submitted

1. "GENOME WIDE IDENTIFICATION OF CIS–REGULATORY MOTIF USING BEOWULF CLUSTER" submitted in "IETE Journal of Research" on 6th April 2017
   Status Under Review
2. "REVIEW OF REGULATORY MOTIF DISCOVERY ALGORITHMS" submitted in "IETE Technical Review" on 15th August 2017
   Status Under Review

Acknowledgements Funding was provided by Ministry of Electronics and Information technology (Grant No. YFRA).

References

1. Nicolae M, Rajasekaran S (2014) Efficient sequential and parallel algorithms for planted motif search. BMC Bioinform 15:34. https://doi.org/10.1186/1471-2105-15-34
2. Ikebata H, Yoshida R (2015) Repulsive parallel MCMC algorithm for discovering diverse motifs from large sequence sets. Bioinformatics. https://doi.org/10.1093/bioinformatics/btv017
3. Fan Y, Wu W, Liu R, Yang W (2013) An iterative algorithm for motif discovery. Procedia Comput Sci 24:25–29
4. Huo H, Zhao Z, Stojkovic V, Liu L (2010) Optimizing genetic algorithm for motif discovery. Math Comput Model 52:2011–2020
5. Kaya M (2009) MOGAMOD: multi-objective genetic algorithm for motif discovery. Expert Syst Appl 36(2):1039–1947
6. Huo H, Zhao Z, Stojkovic V, Liu L (2010) optimizing genetic algorithm for motif discovery. Math Comput Model 52:2011–2020
7. Witte De et al (2015) BLSSpeller: exhaustive comparative discovery of conserved cis-regulatory elements. Bioinformatics 31:3758–3766. https://doi.org/10.1093/bioinformatics/btv466
8. Nicolae M, Rajasekaran S (2015) qPMS9: an efficient algorithm for quorum planted motif search. Sci Rep 5:7813
9. Mohantyr S, Sahu B, Acharya AK (2013) Parallel implementation of exact algorithm for planted (l,d) motif search. In: Proceedings of the international conference on advances in computer science, AETACS
10. Liu FM, Tsai JJ, Chen RM, Chen SN, Shih SH (2004) FMGA: finding motifs by genetic algorithm. In: The fourth IEEE symposium on bioinformatics and bioengineering, p 459