Memory Optimization for Global Protein Network Alignment Using Pushdown Automata and De Bruijn Graph Based Bloom Filter

MD. Sarwar Kamal
Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh
sarwar.saubdcoxbazar@gmail.com

Mohammad Ibrahim Khan
Computer Science and Engineering, Chittagong University of Engineering and Technology, Chittagong, Bangladesh
muhammad_ikhancuet@yahoo.com

Abstract—**Ongoing improvements in Computational Biology (CB) research have generated massive amounts of Protein-Protein Interactions (PPIs) data set. In this regards, the availability of PPI data for several organisms provoke the discovery of computational methods for measurements, analysis, modeling, comparisons, clustering and alignments of biological data networks. Nevertheless, fixed network comparison is computationally stubborn and as a result several methods have been used instead. It is very crucial to utilize the memory of computing devices for Protein-Protein Interactions (PPIs) data set. We have compared the memory uses using Pushdown Automata and de Bruijn graph based Bloom Filter for global proteins network alignment. De Bruijn graph is regularly used in Next Generation Sequencing (NGS) for large scale data set. De novo genome assembler utilizes the memory. Bloom filter and Pushdown Automat perform better to reduce memory. We have noticed that Pushdown Automata outperform Bloom filter in memory saving but it takes more time than Bloom filter. The result shows that Bloom filter software Mania implements full de novo assembly of human genome data set using 6.5 GB memory in 27 hours, on the other hand Pushdown Automat performs same results in 1 GB memory of 31 hours.**

Index Terms—**De Bruijn Graph, Bloom Filter, Pushdown Automata, Next Generation Sequencing.**

I. Introduction

System Biology incorporates the total environment to analyze the data set of plants, animals, human, molecules and insects DNA, RNA and Proteins. We know that majority parts of the biological systems are organized by a network of molecular interactions. Some popular networks are Protein- Protein Interactions (PPIs) network, Genome-Genome Sequencing (GGS), DNA-RNA interactions, genetic regulatory network and metabolic networks. Various sophisticated system can be represented by using network representation. Some gigantic systems are as biological networks, transportation networks, big data set networks and social media network. For biological data set measurement the most phenomenal network is Protein-Protein Interaction network. Proteins perform their activity in combined groups among various cells. A protein corresponds to a node of a graph and the relation between proteins refers as edges of the graph. The PPIs networks are modeled as graph structures [1, 2].

The de Bruijn graph is a mathematical tool which plays an increasingly important role in next-generation sequencing applications for DNA or RNA sequences. This graph was first introduced to perform *de novo* assembly of DNA sequences [3]. Now-a-days this methods massively implemented in *de novo* mRNA [4] and metagenome [5] assembly, genomic variants detection [6,7] and *de novo* alternative splicing calling [8]. But, an important real issue of this structure is its large memory utilization for large organisms. For example, the regular encoding of the de Bruijn graph for the human genome ($n \approx 2.7 \cdot 10^8$, k-mer size $k = 27$) requires 17 GB ($n \cdot k/4$ bytes) of storage area to store the nodes sequences alone. Graphs for much higher genomes and metagenome cannot be organized on a typical lab cluster, due to the prohibitive memory usage.

Current findings on de Bruijn graphs have been targeted on organizing more lightweight data representations. Li *et al.* remarkable research of minimum-information de Bruijn graphs, by not indicating of read locations and paired-end information [9]. Simpson *et al.* experiments a distributed de Bruijn graph to decrease the memory usage per node [10]. Conway and Bromage used sparse bit array format to store an implicit, immutable graph organization [11]. Targeted process compute local assemblies around sequences of interest, using simple memory, with greedy methods [12] or portions of the de Bruijn graph [13]. Ye *et al.* recently showed that a graph roughly equivalent to the de Bruijn graph can be obtained by storing only one out of *g* nodes ($12 \leqslant g \leqslant 27$) [14].

Conway and Bromage checked that the own information of the edges is a lower bound for exactly encoding the de Bruijn graph [11]:

$$\log_2 \binom{4^{k+1}}{|E|} bits$$

$k+1$= length of the sequence that uniquely defines an edge.
$|E|$= is the number of edges.

II. Related Work

A recent research [15] from Pell *et al.* defined the probabilistic *de Bruijn graph*, that is a de Bruijn graph stored as a Bloom filter. It is proved that the graph can be encoded with as little as 4 bits per node. An important limitation of this representation is that the Bloom filter introduces false nodes and false branching. Though, they noticed that the global format of the graph is approximately preserved, up to a limited false positive rate. Pell *et al.* did not implement assembly directly by searching the stochastic graph. Instead, they impose the graph to divide the set of reads into smaller sets, which are then combined in turns using a classical assembler. In the arXiv version of [15], it is unclear how much memory is required by the partitioning algorithm. Some algorithms have designed on Global Network Alignment (GNA). PATH [16], GA [17], NATALIE [18] and NetAlignBP, NetAlignMR [19] all focus on GNA.

I. De Bruijn Graphs And Bloom Filters

The directed graph which maintain edge from all nodes of the graph as a fashion of A=(a_1,a_2,a_3,……….a_n) to B = (b_1,b_2,b_3,……….b_n) with B is being a left-sided part of A, or b1=a2, b2=b3 and so on is called de Bruijn graph. To formal define de Bruijn graph depends on two set as nodes and their relation as dimensions. In the light of Bioinformatics the nodes set can be set of nucleotides of DNA and the dimensions is similar with k-mer length.

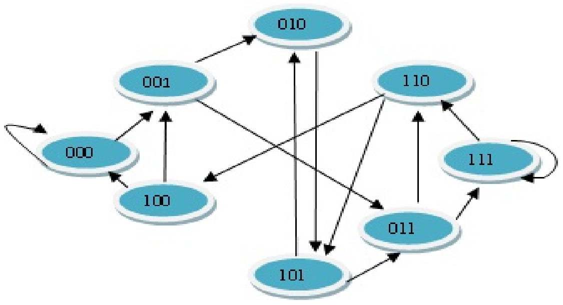


Figure 1: The complete de Bruijn Graph of two symbols 0 and 1.

Example:
Two symbol 0 and 1.
Dimensions =3, so the possible words= 2^3=8 as 00,001,010,011,100,101,110,111. Two nodes are only related if last two symbols of A are same with first two symbols with B (A→B). On the same it can be define that the de Bruijn graph maintain a relation only if last k-1 symbols of node A are similar as the first k-1 symbols of node B.

The **Bloom filter** [20] is a memory efficient hash-based data format, construct to test whether a components is in a set. It designed of a bit array of *m* bits, started with zeros, and *h* hash activity. To input or test the relationship of an element, *h* hash values are measured, yielding *h* array locations. The input operation corresponds to setting all these locations to 1. The relationship operation returns *positives* if and only if all of the bits at these positions are 1. A *negative* answer means the element is definitely not in the set. A *positives* answer indicates that the element may or may not be in the set.

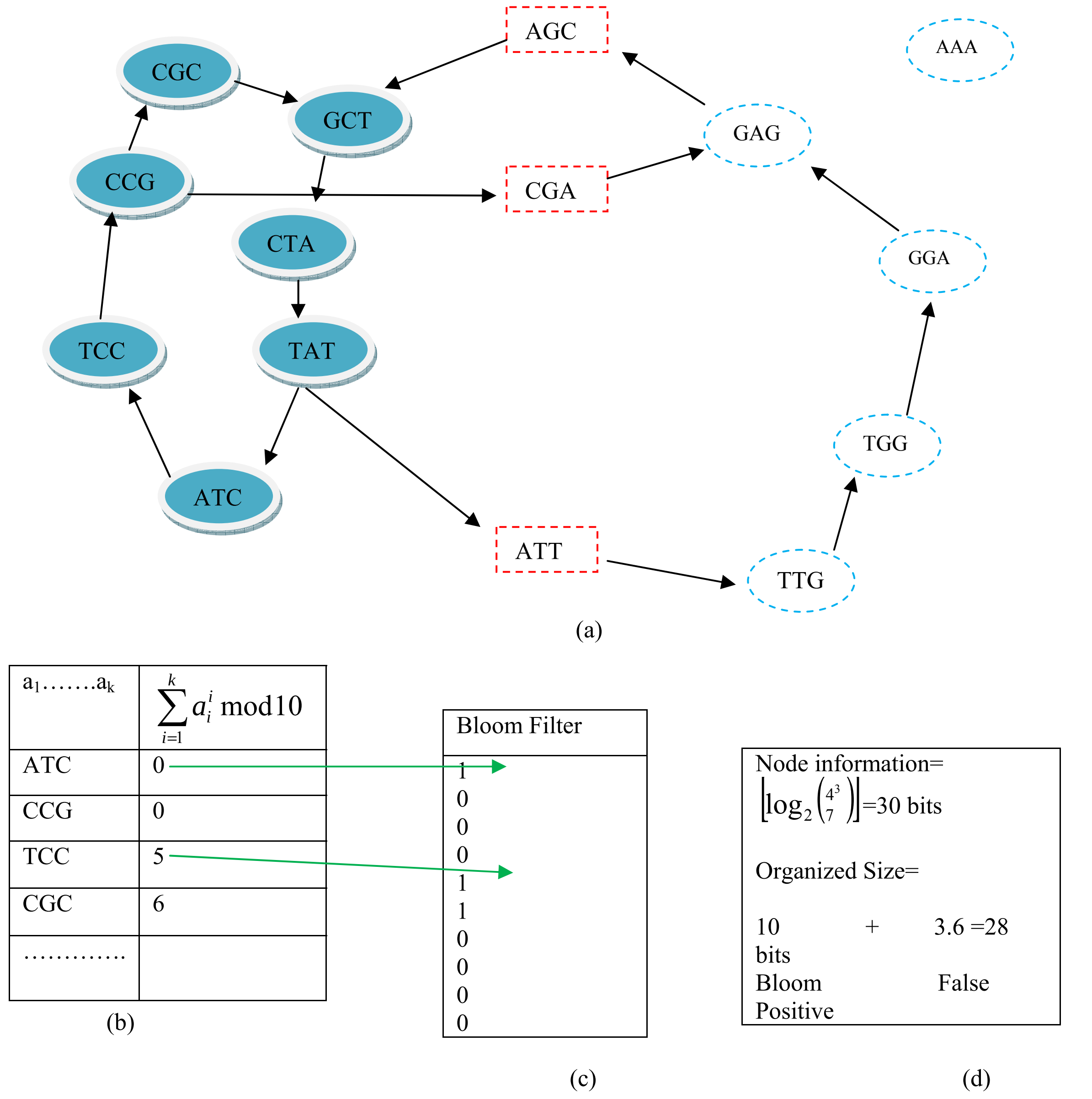
Removing critical false positives

The *cFP* structure

Here we have proposed a system that avoids false branching. To this end, we introduce the *cFP* structure of *critical False Positives k*-mers, implemented with a standard set allowing fast membership test.
Let *M* = true positive nodes
E= be the set of extensions of nodes from *M*.. Let *G*= be the set of all elements of *N* for which the Bloom filter answers *yes*. So **of critical false positives *cFP*** = *G* \ *M*. Figure 2 shows a simple graph with the set *M* of correct nodes in regular circles and *cFP* in dashed rectangles.

Algorithm 1 Constant-memory enumeration of critical false positives

```
1: Input: The set $M$ of all nodes in the graph, the Bloom filter constructed from $M$, the maximum number $M$ of elements in each partition (determines memory usage)
2: Output: The set cFP
3: Store on disk the set $P$ of extensions of $S$ for which the Bloom filter answers yes
4: Free the Bloom filter from memory
5: $D0 \leftarrow G$
6: $i \leftarrow 0$
7: while end of $S$ is not reached do
8:   $Pi \leftarrow \theta$
9:   while $|Gi| < M$ do
10:    $Gi \leftarrow Gi \cup \{$next $k$-mer in $S\}$
11:  for each $k$-mer $m$ in $Di$ do
12:    if $m \,/\!\in Pi$ then
13:      $Di{+}1 \leftarrow Di{+}1 \cup \{m\}$
14:  Delete $Di$, $Gi$
15:  $i \leftarrow i + 1$
16: cFP$\leftarrow Di$
```



$a_1 \ldots\ldots a_k$	$\sum_{i=1}^{k} a_i^i \bmod 10$
ATC	0
CCG	0
TCC	5
CGC	6
…………	

Bloom Filter
1
0
0
0
1
1
0
0
0
0

Node information= $\left\lfloor \log_2 \binom{4^3}{7} \right\rfloor$=30 bits

Organized Size=

10 bits Bloom + 3.6 =28 False Positive

Figure 2: Graphical representation of Bloom filter operations in the probabilistic de Bruijn graph. (a) Shows a network of seven nodes *M* de Bruijn Graph, and *Z*, its probabilistic representation from a Bloom filter. Red Dashed rectangular nodes are immediate neighbors of *M* in *Z*. These nodes are the critical false positives. Blue Dashed node define other nodes of *Z*; **(b)** Hash table of *M* **(c)** Bloom filter related to M; incidentally, **(d)** Space requirement of and saving measurements.

III. Pushdown Automata

Pushdown Automat (PDA) is very useful tool in computer science to free occupied memory space. Push Down Automaton (PDA) likes a finite automaton that has a single stack. A stored of data from stored item can be retrieved, also known as push down stack or push down list.PDA can write symbol in the stack and read back from later. A read-write head executes the POP operation by reading topmost symbols and erase from stack.

PUSH operation executes simply writing operation in stack. Stack offer the infinite memory and PUSH-POP the symbol in “”Last-In-First-Out” fashion. Pushdown Automata contains 3 elements: 1.Input Tape 2.Stack Input 3.Control Unit

PDA starting from initial state and reads the symbol from input tape. It switches to next state, where control unit reads symbol from input tape and matched with the stack symbol for each input tape symbol. When control unit matched with stack symbol, the symbol on the top of the stack removed and the remaining symbol moved up. Removing symbol from the stack referred to as popping symbol. When top symbol of stack and the input tape symbol not matched, the input does not belong to the languages or string, and PDA block or reject the input. If it reaches end of the input tape and stack is empty, then the input belongs the string and accept it.Fig-1shown schematic presentation of PDA.

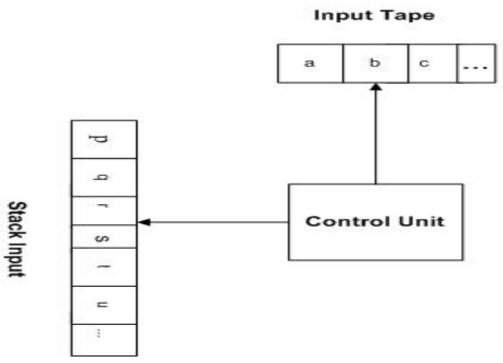


Figure 3: Schematic presentation of PDA

Memory Mapping for Anchor Selection

The complete view of the memory saving process is illustrated in figure 4 below. Here we see that there are three key units that manage memory efficiently. These units are Control Unit, Matching Unit and Count Unit. Along with these units two tables are available as Stack Table and Input data table. Stack Table contains the whole data set of proteins or DNA or RNA. Control unit restricts the matching with desired input from whole data set. When match or mismatches occurs it frees the stack. At the end of the input table, there is an end symbol that determines the last terminal of the input data position. It refers that a match is occurred. While a complete Anchor is matched from given data set count unit will counts the total value of these matches and returns the matching seeds. When a mismatch is occurred control unit pop the data set from stack but do not count the anchor found. The overall view of these processes is mathematically narrated at algorithm 2 below.

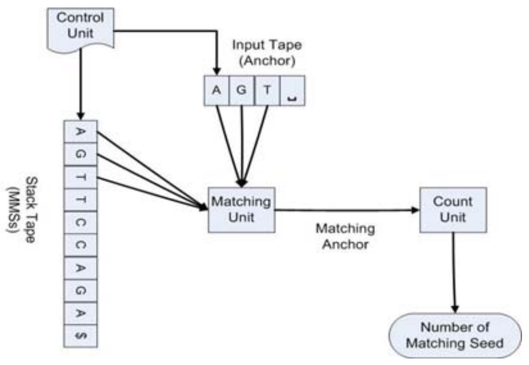


Figure 4: The Memory Mapping for Anchor Selection

Here this algorithm proposed the memory efficient for finding anchor from a local DNA sequence. This algorithm can handle more than 500 base pair of gene sequence. Some other methods like full hashing and heuristic based approaches are used in memory optimization but not efficient than that of Bloom Filter and PDA.

Algorithm 2:

```
Memory Mapping (Θ,Γ, φ,max_length)
1: Initialize the input symbol (anchor) in input tape (Θ)
            $\Theta \leftarrow Anchor$
2: Initialize MMSs in stack tape (Γ)
            $\Gamma \leftarrow MMSs$
3: Initialize count unit (φ)
            $\vartheta \leftarrow 0$
4: $i \leftarrow 1$   and $j \leftarrow \max\_length$
5: while ( $\Gamma[j] == \$$ )
6: If ( $\Theta[i] == \Gamma[j]$ )
7:       $i \leftarrow i+1$   and   $j \leftarrow j-1$
8:          if ( $\Theta[i] == \perp$ )
9:             $\vartheta \leftarrow \vartheta + 1$
10:            $\Theta[i] \leftarrow \Theta[1]$
11:         End if
12:         else go to step 6.
13:         if ( $\Gamma[j] == \$$ )
14:            print φ.
15:         End if
16:         else go to step 6.
17: End if
18: else $j \leftarrow j-1$
19:    If ( $\Gamma[j] == \$$ )
20:       print φ.
21:    End if
22:    else go to step 6
23: End loop
```

IMPLEMENTATION

De Bruijn graph based Bloom filter is implemented under the environment of *de novo* assembly software: Miniaa. A pivotal preliminary measure is to pick the list of distinct *k*-mers that appear in the reads. To remove likely sequencing errors, only the *k*-mers which appear at least *d* times are kept. We experimentally set *d* to 3. Consequently, the Pushdown Automata based memory saving have implemented and experimented under the environments of Java with Integrated Development Environment (IDE) Netbeans. The object oriented implementation helped us to perform the nucleotides (A, C, T, and G) as a distinct object. This object oriented implementation enables faster managements of data set under various class levels for complete data set. We first customize the whole data set into a fixed stack. Then the data set are merged together into fixed memory space. According to the PDA analysis, each and every character set is matched with stored data set in stack. While all the values are matched with the comparisons of desired Anchor seed of the DNA segments, it will counts that one Anchor has found and the process will continue until reached the last nucleotide base pair. In Perl, it is little difficult and time consume to design the PDA and algorithmic analysis due to its more formal and structural based formation. Besides, it is easy to implement the concept under the environment MatLab. But MatLab is not perfect for this environment because desired algorithm functions are not available and difficult to implement the algorithm. Our experiments outperform the result of existing system irrespective of Perl and MatLab environments. That why we have choose Java to implement the system.

IV. Result

The outcomes of these two process, a few interesting changes have had observed. Pushdown Automata is very efficient for any arbitrary predictions in any DNA segments or sequences. It is clearly noticed that Pushdown Automata has potential and strong capabilities to handle the data set whatever the environment. Figure 5 below shows the performance Pushdown Automata.

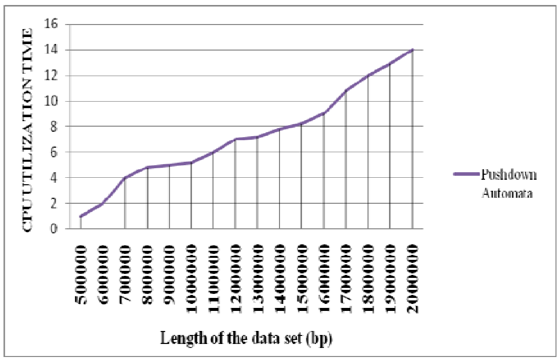


Figure 5: The impact of Pushdown Automata

For de Bruijn graph based alignment the CPU Utilization time for long data set range from 1500000 to 2000000 requires 8.2, 8.5, 10, 11.4,12, and 12.5. On the contrary, Pushdown Automata algorithm takes more time for the same data set and the time values are 8.2, 9, 10.8, 12, 12.9 and 14. But for the previous data set whose lengths are less than 1500000, Pushdown Automata based process takes less time than de Bruijn graph based alignment. Figure 6 below shows the impact for de Bruijn graph based alignment.

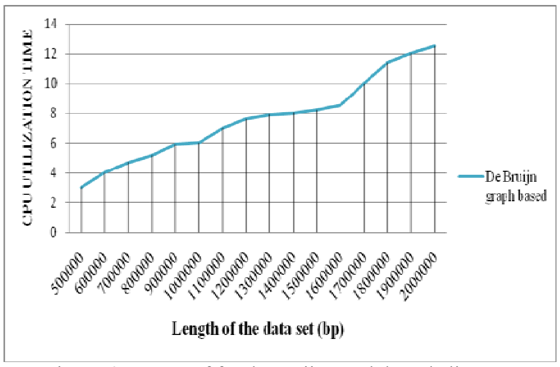


Figure 6: Impact of for de Bruijn graph based alignment.

The reasons behind for de Bruijn graph based alignment requires more time to solve small data set is that it works for arbitrary probabilistic values where Pushdown Automata works deterministic path and values. The comparative results of these two methods are below at figure 7.

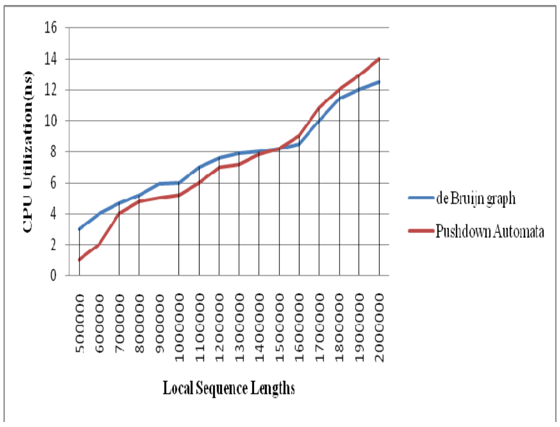


Figure 7: Comparative Illustration of Dynamic Programming and Warshall Graph Algorithm

V. Conclusion

Both de Bruijn graph based alignment and Pushdown Automata perform predictions of DNA base pair according to the process. de Bruijn graph based alignment has better capabilities to handle large data set due to its randomness. On the other side, Pushdown Automata works based on predefine values and path. That why Pushdown Automata has to check the entire path and values weather the path is short or long. That is the reason de Bruijn graph based alignment requires more time. We will find why Randomness causes more time and deterministic process is better for small data set in future work.

References

[1] M.Lavallée-Adam, B.Coulombe, M.Blanchette, “Detection of locally overrepresented GO terms in protein-protein interaction networks”, Research in Computational Molecular Biology Springer;302-320,2009.

[2] Z.P. Li, S.H.Zhang, Y.Wang, X.S.Zhang, L.Chen, “Alignment of molecular networks by integer quadratic programming”. Bioinformatics ,23(13):1631-1639,2007.

[3] R.M.Idury, M.S.Waterman,A new algorithmfor DNA sequence assembly**.** *J Comput Biol* ,**2**(2):291–306,1995.

[4] M.G.Grabherr,Full-length transcriptome assembly from RNA-Seq data without a reference genome. *Nat Biotech*, 29(7):644–652,2011. [http://dx.doi.org/10.1038/nbt.1883]

[5] Y.Peng, H.C.M.Leung, S.M.Yiu, F.Chin, Meta-IDBA: a de Novo assembler for metagenomic data**.** *Bioinformatics*, 27(13):i94–i101,2011.

[6] P. Peterlongo, N.Schnel, N.Pisanti, M.F.Sagot, V.Lacroix, Identifying SNPs without a reference genome by comparing raw reads. In *String* Processing and Information Retrieval. Berlin, Heidelberg: Springer; 147–158,2010.

[7] Z. Iqbal, M.Caccamo, I.Turner, P.Flicek, G.McVean, De novo assembly and genotyping of variants using colored de Bruijn graphs. *Nat Genet* 2012, 44**:**226–232.

[8] G.Sacomoto, J.Kielbassa, R.Chikhi, R.Uricaru, P.Antoniou, M.Sagot, P.Peterlongo, V.Lacroix, KISSPLICE: de-novo calling alternative splicing events from RNA-seq data. *BMC Bioinformatics* 2012, 13(Suppl 6):S5. [http://www.biomedcentral.com/1471-2105/13/S6/S5]

[9] R.Li, H..Zhu, J.Ruan, W.Qian, X.Fang, Z.Shi, Y.Li, S.Li, G.Shan, K.Kristiansen, De novo assembly of human genomes with massively parallel short read sequencing**.** *Genome Res* , 20(2):265,2010.

[10] Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJM, Birol I: ABySS: A parallel assembler for short read sequence data. *Genome Res*19(6):1117–1123 [http://genome.cshlp.org/content/19/6/1117.abstract],2009.

[11] T.C. Conway, A.J.Bromage, Succinct data structures for assembling large genomes**.** *Bioinformatics*, 27(4):479,2011.

[12] R.L. Warren, R.A.Holt, Targeted assembly of short sequence reads. *PloS One*, **6**(5):e19816,2011.

[13] Peterlongo P, Chikhi R: Mapsembler, targeted and micro assembly of large NGS datasets on a desktop computer. BMC Bioinformatics, 13:48,2012.

[14] C.Ye, Z.Ma, C.Cannon, M.Pop, D.Yu, Exploiting sparseness in de novo genome assembly**.** *BMC Bioinformatics* 2012, **13**(Suppl 6):S1. [http://www.biomedcentral.com/1471-2105/13/S6/S1]

[15] J.Pell, A.Hintze, R.Canino-Koning, A.Howe, J.M.Tiedje, C.T.Brown, Scaling metagenome sequence assembly with probabilistic de Bruijn graphs**.** Arxiv preprint arXiv:1112.4193 2011.

[16] M.Zaslavskiy, F.Bach, J.P.Vert,A path following algorithm for the graph matching problem. IEEE Trans Pattern Anal Mach Intell, 31(12):2227-2242,2009.

[17] M.Zaslavskiy, F.Bach, J.P.Vert : Global alignment of protein-protein interaction networks by graph matching methods. Bioinformatics, 25(12):i259-i267,2009.

[18] G.Klau, A new graph-based method for pairwise global network alignment. BMC Bioinformatics, 10(Suppl 1):S59,2009.

[19] M.Bayati, M.Gerritsen, D.F.Gleich, A.Saberi, Y.Wang, Algorithms for large, sparse network alignment problems. 2009 Ninth IEEE International Conference on Data Mining IEEE, 705-710,2009.

[20] A.Kirsch, M.Mitzenmacher , Less hashing, same performance: Building a better Bloom filter**.** *Algorithms–ESA*, 4168**:**456–467,2006.

Md.Sarwar Kamal received the B.Sc (Hons) in computer science and engineering from University of Chittagong Bangladesh in 2009. Since 2009, he has been serving as a faculty member in the Department of Computer Science and Engineering at BGC Trust University Bangladesh Chittagong. Now he is MS (Engineering) student in the Department of Computer Science and Engineering at Chittagong University of Engineering & Technology (CUET), Chittagong, Bangladesh. Currently his research project on Bioinformatics Local sequence alignments algorithms analysis under the guideline of Dr. Mohammad Ibrahim Khan .His research interest includes Bioinformatics and Data mining.

Dr. Mohammad Ibrahim Khan received the B.S. degree in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology (BUET), Bangladesh in 1999. He received M.S. degree in Computer Science and Engineering from the same University in 2002. He received his Ph.D. degree in Computer Science and Engineering from Jahangirnagar University in 2010. Since 1999, he has been serving as a faculty member in the Department of Computer Science and Engineering at Chittagong University of Engineering & Technology (CUET), Chittagong, Bangladesh. Currently his research project on Bioinformatics Local sequence alignments algorithms analysis with his research group. His research interest includes Digital Image Processing, Graph Theory, Cryptography, Digital Watermarking, Multimedia Systems, and Digital Signal Processing.