Indian Journal of Animal Sciences **87** (9): 1155–1162, September 2017/Article

Parallel network motif search using message passing approach for biological complex networks

HIMANSHU[1], A BANDYOPADHYAY[2] and SARIKA JAIN[3]

Amity Institute of Information Technology, Amity University, Noida, Uttar Pradesh 201 313 India

Received: 23 May 2017; Accepted: 7 July 2017

ABSTRACT

Study of complex biological networks is essential for understanding their functional characteristics. Network motifs have functional significance in biological networks as they represent building blocks of these networks. This study evaluates master-worker parallelization approach on sequential PATRICIA trie based fast network motif search algorithm for distributed memory model based High Performance Clusters (HPCs). Proposed algorithm uses PATRICIA trie for data compression during census of subgraphs based upon ESU algorithm. Parallel implementation was done using MPI and C language. We applied proposed parallel algorithm to three real networks viz. networks of metabolic pathway of *E.coli*, electronic and social networks. PATRICIA based parallel approach was able to achieve speedup of 50.75, 49.37, 38.07 as analysed on 101 cores on networks of metabolic pathway of *E.coli*, electronic and social networks respectively for large motifs of size 9 for *E.coli*, social and 10 for electronic networks over the PATRICIA trie based sequential algorithm.

Key words: Complex networks, Data structures, HPC, MPI, Network motif search, Parallelization

Explosion of publicly available high throughput data generated from automation in cell biology has led to development of cell models. As a result, their analysis has also gained significant interest. High throughput techniques help to speed up research in cell biology, drug discovery etc. Gene transcription and signal transduction mechanism are some of the cell processes that play an important role in every process of life, including cell differentiation, metabolism and the cell cycle. Computational methods for development of network models and analysis of their functionality have proven to be valuable research tools (Karlebach and Shamir 2008). Network motif search is one such analytical tool available to researchers. In molecular biology, network motifs are recurrent and unique subgraph patterns occurring in biomolecular networks that are helpful in understanding basic functional components of such complex networks (Shen-orr *et al.* 2002).

Discovering larger motifs imposes a significant challenge as the exponential runtime makes a significant impact on the performance. The most obvious solution to decrease runtime is by implementing parallel algorithms (Tran *et al.* 2014). In the present work, we focused on development and analysis of parallel version of network motif search tool suitable for large network motif discovery and capable of running on distributed-memory high performance clusters. Distributed-memory clusters consist of several computer nodes having multiple processors with 2–16 or more cores each and usually high amount of RAM per node. Distributed memory program expects that each task will have a different virtual address space. So, available memory gets divided among the cores of a node in case of distributed-data parallelism. When in-memory data is large, it is not possible to employ more cores per processor for parallel processing or concurrency, as limitation of memory could make the algorithm infeasible. For higher scalability, it becomes imperative to use a data structure for in-memory representation of data such that the memory cost of algorithm is low for data parallel approach.

Network motif search is a three step process involving identification of all k-size subgraphs in target network, finding subgraph isomorphism, enumeration and frequency counting of k-size subgraphs in random networks and statistical significance calculation. Detecting subgraph isomorphism is computationally hard problem. As the motif size (k) increases or the target network size (s) increases, the problem becomes intractable on sequential systems. Many intractable problems are solved using parallel computing in practice as it helps to expand resources by way of implementing parallel strategies. Network motif search in its parallel form can also speed-up the large motif discovery in target networks. Divide and conquer is one of the many methods for engineering parallel software applications. Based on the methods used to divide tasks and recombine the end result, we get alternative models of parallel programs particularly, granularity. Granularity is the quantity of work done in parallel tasks; and

Present address: [1](himanshu.icar@gmail.com). [2] ITRA, Ministry of Communication and Information Technology, New Delhi 110 030.

communication may be defined as the way in which different units of computation exchange data in order to synchronize their activity. Too fine granularity results in too much communication overhead whereas coarse granularity leads to workload imbalance.

Major constraints in parallelization of network motif search require effective synchronization of parallel census, random network, and statistical significance calculation which could lead to serialization of network-centric algorithms. As we divide the subgraph enumeration into parallel work units, it causes unbalanced computation workload due to non-uniform task size. Dynamic load balancing approach is one of the ways to achieve better and more effective parallelization. Shared memory model systems have advantage that each memory location can be accessed by each core/processor in equal time thereby eliminating the communication time delays. However, such algorithms are not scalable. Distributed memory model has advantage that solutions are scalable, although communication delays are inherent.

Parallelization opportunities exist in network motif search algorithms in census, random network and statistical significance calculation. Approaches to census parallelization include partitioning the target network and each partition be assigned to separate work unit; search tree parallelization wherein recursive search procedure is executed in parallel, with different search tree branches being searched at the same time on separate processor, and query parallelization in single-subgraph algorithms (Rebeiro *et al.* 2012). Random network parallelization is achieved by distributing random networks among the processors and each processor carry out the complete census on its random network. Significance calculation could be parallelized by distributing the calculations at the end of census.

Enumerate Subgraph (ESU) algorithm of FANMOD (Wernicke and Rasche 2006) used for enumeration of k-size subgraphs from network, produces a search tree type structure. ESU is therefore suitable for parallelization. We chose ESU and the tree parallelization approach for subgraph enumeration and census parallelization. It could be ascertained that as we distribute the search tree branches to the processing cores of distributed-memory cluster, each core would require to maintain its own individual copy of data structure to store the subgraph isomorphism details in memory so that number of calls to 'nauty' could be reduced for identification of subgraph isomorphism each time a new subgraph is enumerated. Further, it is observed that as the query subgraph size increases, size of g-trie (Ribeiro and Silva 2009) and quaternary tree (Khakabimamaghani *et al.* 2013) increases much higher than PATRICIA trie (Himanshu *et al.* 2017). So g-trie and quaternary tree based algorithms run the risk of becoming infeasible on distributed-memory cluster if parallel work units are assigned to more number of cores per processor, as the motif size becomes large. Therefore for an algorithm to be effective, it would be desirable to execute parallel work units on larger number of cores per processor for parallel computations. The algorithm which is based upon a compressed data structure to use less memory during census would be more suitable for parallel processing without becoming infeasible as motif size becomes large or in case of bigger target network.

In the present work, we describe parallel network motif search tool which is based upon our previously developed PATRICIA trie based sequential algorithm for large network motif search (Himanshu *et al.* 2007). We have parallelized census on original graph using master-worker strategy with dynamic load balancing of work units. The parallel algorithm is based upon network centric Enumerate Subgraph algorithm to enumerate all possible subgraphs using pattern growth tree structure called ESU-tree. The algorithm uses PATRICIA trie for storing subgraph related substrings in the main memory to reduce the calls to the subgraph isomorphism checking tool 'nauty' for subgraphs similar to those for which isomorphic class has already been determined previously, as 'nauty' is immensely slow. The algorithm attempted to optimize the time and memory cost for fast network motif discovery using PATRICIA trie. The algorithm had clear advantage over QuateXelero for GRN of metabolic pathways *of E.coli*. The algorithm using PATRICIA trie was found to use 7.8 – 38.35% lesser memory as compared to Quaternary tree based algorithm, with satisfactory time performance for networks analysed. In present work, we explore the parallel approach for PATRICIA based network motif search algorithm and analyse its performance on different networks of *E.coli*, social and electronic.

Network motif: Shen-Orr *et al.* (2002) were first to propose network motifs as simple building blocks of complex networks. Network motifs were defined for the first time as pattern of interconnections occurring in complex networks in numbers that are significantly higher than those in the randomized networks. Algorithm was applied to networks from biochemistry, ecology, neurobiology and engineering. Network motif studies in biochemistry included transcription networks which are responsible for regulating the expression of genes in cells. Here, networks are directed networks such that nodes represent genes and edges are directed from a gene that encodes for a transcription factor protein to a gene transcriptionally regulated by that factor. Ever since the network motifs search has gained importance as it is useful in understanding the properties of large-scale, more complex networks such as metabolic pathway of *E. coli*, protein-protein interaction and transcription network of *S. cerevisiae* (Grochow and Kellis 2007).

Network centric approach: Given a number k, problem is to find all k-size overrepresented sub-graphs in the target network G than in the random graph R_g having degree distribution (in and out degrees) as in G.

Speedup: For the analysis of parallel programs, a comparison with the execution time of its sequential implementation is especially important to see the benefit of parallelism (Rauber and Rünger 2009). Such a

comparison is often based on the relative saving in execution time as expressed by the notion of speedup. The speedup S_p (n) of a parallel program with parallel execution time T_p (n) is defined as

$$S_p (n) = T_s (n) / T_p (n)$$

where, p is the number of processors used to solve a problem of size n, T_s (n) is the execution time of sequential implementation to solve the same problem.

Efficiency: It is an alternative measure for the performance of a parallel program. The efficiency captures the fraction of time for which a processor is used fully employed by computations that also have to be performed by a sequential program. The definition of the efficiency is based on the cost of a parallel program and can be expressed as:

$$E=S/p; \text{ where } S=T_{seq}/T_p$$

where T_p is the time a given parallel algorithm needs before producing an answer on p units of execution.

Scalability: It is a measure describing whether a performance improvement can be reached that is proportional to the number of processors employed. Scalability depends on several properties of an algorithm and its parallel execution. Often, for a fixed problem size 'n', a saturation of the speedup can be observed when the number 'p' of processors is increased. But increasing the problem size for a fixed number of processors usually leads to an increase in the attained speedup. In this sense, scalability captures the property of a parallel implementation that the efficiency can be kept constant if both the number 'p' of processors and the problem size 'n' are increased.

Among the earliest attempt on parallelization of network motif search, the algorithm of Wang *et al.* (2005) involved use of Breadth First Search (BFS) along with network parallelization. The algorithm defines and uses the concept of neighbourhood assignments for each node such that partitioning of induced neighbourhoods does not cause overlapping. A set of neighbours is distributed to each worker for parallel computation of subgraph frequencies. The algorithm also uses a load balancing method as the workload is statically computed according to the node degree in each vertex, and the work units are distributed to workers. After collecting subgraphs from each worker, the graph isomorphism is checked and network motifs are identified. The algorithm was analysed with the transcription regulatory network of *E. coli* and *S.cerevisiae* and scales-up up to 32.

Schatz *et al.* (2008) demonstrated two techniques i.e. network partitioning and query set partitioning in their work to parallelize Grochow-Kellis algorithm. The query parallelization, suitable for parallelization of motif centric algorithms, is used to distribute query set among the workers such that each work unit is a subset of the query set. Each worker also gets a full copy of the target network. The algorithm implements two approaches viz. a naive and a first-fit scheduler for workload balancing. Naive approach implements a static load distribution and first-fit distributes the works on demand so has higher overhead. Query parallelization scheme obtained a linear speedup up to 64 processors on a single network of *S.cerevisiae* for small motif size (up to 8). A novel network partitioning method based on hierarchical clustering, has been used for network parallelization. Network parallelization helps to speedup the single query computations and obtains a linear speedup up to 32 processors with small sub-network size. Algorithms were implemented using Java.

Ribeiro *et al.* (2010) presented efficient algorithm for parallel subgraph counting using G-tries, a specialised data structure designed to store and search for subgraphs, implements motif centric approach using receiver initiated work stealing for dynamic load balancing of work. The algorithm was analysed with diverse networks and it achieved almost linear speedup upto 128 processors.

In subsequent work, Ribeiro *et al.* (2011) utilized efficient parallel subgraph counting algorithm to parallelize complete network motif search process i.e. census, random graphs and significance calculation phase. They also defined small work units of census, random network and significance calculation as per their new serial algorithm for parallelization. The algorithm implements dynamic load balancing approach i.e. work-stealing approach of recursively requesting for work by idle processes, from workers having unfinished work. The algorithm also proposed two alternatives; centralized control (with a master–worker strategy) and completely distributed control (with a randomized receiver initiated strategy) for load balancing by work-stealing. Small independent work units that can be redistributed have been used to provide dynamic load balancing both for exhaustive complete computations and approximations using sampling. The work efficiently parallelized the whole network motif discovery problem. Implementation was done using MPI and C++. The algorithm achieved almost linear speedup up to 128 processing cores. G-tries based approach for parallelization uses threshold parameters namely 'splitThreshold' in master-worker approach and 'checkRequestsThreshold' in distributed approach in the algorithm. These parameters are manually calculated based on the number of units processed on the specific HPC; thereby the algorithm is system specific for optimal performance. Adaptability of the algorithm for very different computing environments is not simple. In the paper authors assert that in future they plan to automatically and dynamically compute all threshold parameters, allowing the algorithm better adaptability for very different computing environments. Working tool for discovering motifs is not available.

MATERIALS AND METHODS

Different phases of parallel network motif search method using PATRICIA trie are discribed here. The process is divided into initialization, original graph, random graph and significance calculation phases. Parallelization of original graph and random graph phases was targeted in this method.

Master-worker strategy was used to distribute work units with dynamic load balancing scheme. As per the master-worker strategy, a worker is dedicated exclusively for the load balancing and work distribution (called the master) whereas all other workers process work and communicate only with the master in what regards the work phase.

Initialization phase: As part of initialization, master processing unit reads target graph from a text file residing on hard disk; as on a distributed cluster, only master node may have access to file system on storage servers. It then sends the input graph adjacency matrix to all the worker processing units so that each worker has its own copy of the target network.

Original graph phase: As in network centric approach, we first carry out the enumeration and census of all the k-size subgraphs present in the original network. After all the non-isomorphic subgraph classes in original network have been identified, we proceed for census of only those non-isomorphic subgraphs identified in the original network, in specified number of random graphs.

Enumerate Subgraph (ESU) algorithm of Wernicke and Rasche (2006) is known to produce a pattern growth tree like structure as it enumerates all k-size subgraphs in a network. It uses a function named 'Extendsubgraph' which begins enumeration with lowest numbered vertex. More vertices are then added to the list from the list of its exclusive neighbours. As it uses only those vertices with index number higher than its root vertex index number to extend the partial subgraph ensures that the subtasks are exclusive. It goes on adding vertices to the list iteratively till there are exactly 'k' vertices. The set of k-size lists is the exhaustive set of k-size subgraphs enumerated from the target network. We parallelize enumeration and census on target network by distributing the root vertices among available workers. Each worker then carries out enumeration of k-size subgraphs obtained by starting from that specific vertex and frequency counting of enumerated subgraphs. Mckay's canonical labelling tool 'nauty' is used for subgraph isomorphism checking. It is not possible to know in advance the workload of each worker, so we use dynamic first–fit on demand strategy for load balancing. Work distribution based on root vertices of original network phase creates two different scenarios viz. number of processors could be less than or equal to the number of vertices in target graph ($n \leq s$), or number of processors is more than number of vertices ($n>s$). In first scenario, master process distributes first n-vertices among all the n-workers for enumeration and census of the enumerated subgraphs. Subsequently, whenever any of the n-workers completes enumeration and census of subgraphs, they send a request message to master process to allocate more work. So long as more work units are available, master process keeps sending next available vertex to the workers requesting more work for further processing. This process repeats until all the vertices have been exhausted when master process sends a message with 'FINISH' tag to the worker processes requesting more work to finish the enumeration. All the worker processes during the enumeration and census maintain their individual copies of PATRICIA trie and ZeroOne binary tree. Details of PATRICIA trie and ZeroOne tree data structure and their application have been illustrated in sequential algorithm based network motif search process (Himanshu *et al.* 2017). When a worker receives the message with 'FINISH' tag, it sends three lists viz. a list of subgraph strings obtained by inorder traversal of PATRICIA trie, and other lists containing canonical labels of non-isomorphic subgraph classes and frequencies of the enumerated subgraph obtained from leaves of ZeroOne tree to the master process. The list of canonical labels of subgraph classes is arranged in the same order as their linked PATRICIA trie strings are placed in the list so that it retains the association related details during aggregation in master process. It is worth mentioning that two different leaves in PATRICIA trie may point to same leaf in ZeroOne tree as subgraphs with different types of connectivity may be isomorphic. However, converse is not true. Hence, canonical labels appear repeatedly in the list of ZeroOne tree canonical labels more than once. Master process starts the aggregation process of sets of lists received and waits until all the 'n' workers have finished their allocated work and have sent back these three lists.

Master process aggregates sets of lists of PATRICIA trie strings and k-size subgraph canonical labels received from each of 'n' workers processes and reconstructs a consolidated PATRICIA trie and a ZeroOne tree by recursively inserting the items of the lists in a new PATRICIA trie and ZeroOne tree respectively. The leaves of PATRICIA trie point to one of the leaf of its non-isomorphic class in ZeroOne tree. Each leaf of ZeroOne tree which doesn't have further child leaves stores two

```
Input: original network, motif size, numProcessors, numRandom
1:  initialization()
2:  for all workers do
3:     sendMessage(originalNetwork)
4:     sendMessage(subgraphSize)
5:  Call EnumerateOriginalMaster()
6:  MPI_Barrier()
7:  inorder(patriciaTrie, patStrList, canLabList)
8:  for all workers do
9:     sendMessage(patStrList)
10:    sendMessage(canLabList)
11: MPI_Reduce(freqList)
12: While notFinished(randcount) do
13: receiveMesgwrkr(rfreqList, workerRank) // from initial n
workers
14:    sendMessage(countofRands, rank, WorkTag)
15: for all workers do
16:    receiveMesgwrkr(rfreqList, workerRank) // from all
remaining workers
17: for all workers do
18:    sendMessage(Null, rank, FinishTag)
19: SignificanceCalculation()
20: return
```

Fig. 1. Main program-master process

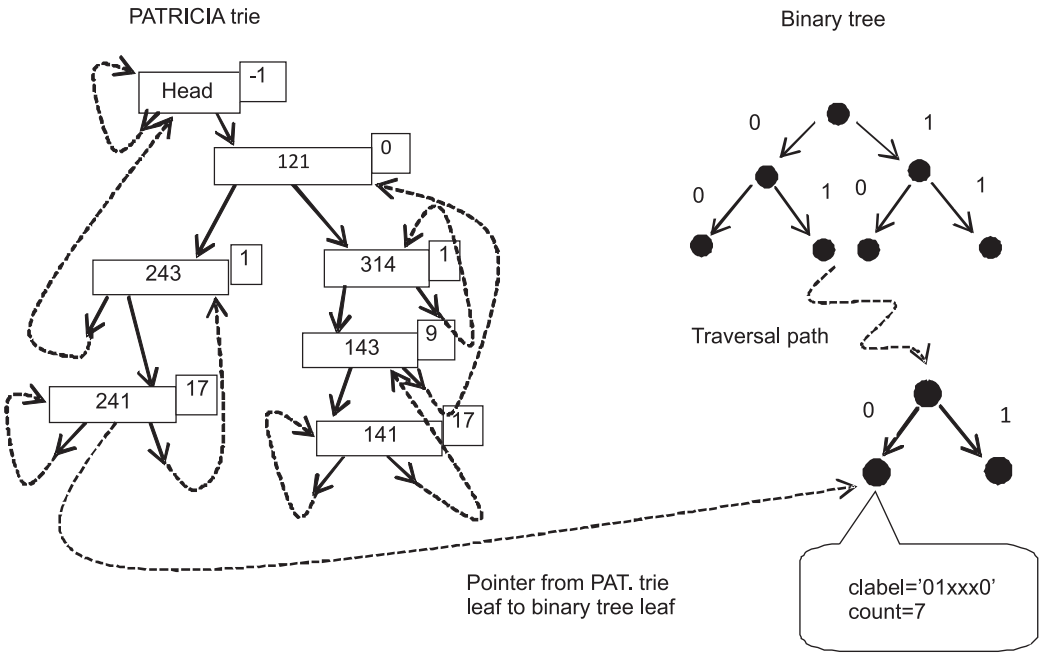

Fig. 2 . PATRICIA trie and binary tree used for enumeration during original and random network phases

variables one each for the frequency count of subgraphs in that particular non-isomorphic class and canonical label of the class (Fig. 2). During aggregation process, if the canonical label corresponding to class of a new string being inserted in PATRICIA trie previously exists in ZeroOne tree, no new leaves get inserted, otherwise a new canonical label is inserted in ZeroOne tree. For aggregation of frequencies of the enumerated non-isomorphic classes, we adopt two-step process. After the aggregation of PATRICIA and ZeroOne tree, an exhaustive list of all the canonical labels of non-isomorphic classes was generated by depth-first traversal of consolidated ZeroOne tree. Inorder traversal of consolidated PATRICIA trie was performed to generate exhaustive list of strings. Both the lists were then sent to all the worker process. All the PATRICIA strings and canonical labels were then inserted by workers in their copies of PATRICIA trie and ZeroOne tree. Frequency of newly inserted canonical label was set to zero. This is essential, as all the workers during original graph phase may not have enumerated all non-isomorphic classes during the enumeration of subgraphs from the root vertices allocated to the respective process during original graph phase. So, after inserting the elements of lists received from master, each worker's copy of ZeroOne tree had all the non-isomorphic classes in similar order and also content of their copy of PATRICIA was same. Each worker then extracts its list of frequencies by depth-first traversal of its copy of ZeroOne tree. The frequencies were then aggregated by the master process by using 'MPI_Reduce' command. After the consolidation process got over, master process owns a PATRICIA trie of all unique strings of PATRICIA tries received from all worker processes and a ZeroOne tree of all the unique identified non-isomorphic subgraph classes in the original graph and their frequencies. The aggregate list of frequencies of k-size non-isomorphic subgraph classes from census of target network was used by master process later for significance calculation.

In case, the number or processors available are more than the number of vertices in target network, the nodes got distributed to just as many processors as number of vertices, and remaining processors remain idle during this phase as random network phase can start only after completion of census of original network in network-centric approach of motif search problem.

The arrangement of strings in these two lists has to be kept in the order of sequence of strings obtained from inorder traversal of PATRICIA trie (as described above). Worker processes reconstruct their copies of PATRICIA trie and ZeroOne tree by simply repeatedly inserting strings of the lists received from master process. We achieved the synchronization of the census on original network and random networks by placing 'MPI_Barrier' command in master and worker processes of main program (Figs 1 and 3).

Random graph phase: Master-worker strategy with dynamic, best-fit on-demand load-balancing was used in a different way from original network phase for parallelizing random network census process. Each worker process owned its individual copy of original network and list of non-isomorphic subgraph classes identified during original graph enumeration in ZeroOne binary tree data structure in memory. Each worker also had a dictionary of subgraph strings in PATRICIA trie- a compressed data structure. We presume that number of available processors for the search would always be less than or equal to number of random graphs since in real world situation, network motif search requires census over thousands of random networks for accuracy purpose.

```
1:  receiveMesgmMaster(originalNetwork)
2:  receiveMesgMstr(subgraphSize)
3:  EnumerateOriginalWorker(g)
4:  MPI_Barrier()
5:  receiveMesgMaster(patStrList)
6:  receiveMesgMaster(canLabList)
7:  consolidatePatZeroOne()
8:  extractfreqZeroOne()
9:  MPI_Reduce(freqList)
10: forever loop
11:     if (FinishTag(Tag)then
12:     return 0
13:   generateRandGraph()
14:   EnumerateandCensusRand(subgraphsize)
15:   ExtractFreq()
16: sendMesgMaster(rfreqList, workerRank)
17: receiveMsgMaster(randCount, Tag)
```

Fig. 3. Main Program – worker process.

During random network phase, all the worker processes start by generating random network using edge-switching method on their copy of original network. Subsequently, each worker process goes on with the enumeration and census on its random network. After completing the census, the worker sends a list of count of frequencies of non-isomorphic classes of the random network to master process. Master process maintains a count of the number of random networks being enumerated by workers. So long as this count is less than desired number of random graphs (r), master process keeps sending message with a 'WORK' tag to the worker which has completed the census of random network. As soon as the count of number of random graphs processed reaches the desired value (r), it sends message with a 'FINISH' tag to all the workers. Master process thus receives the list of count of frequencies from all remaining workers.

In our approach of random graph phase parallelization, enumeration and census of each random graph was a single work unit and was executed in parallel. So, it was not possible to execute the tool on processors/cores higher than number of random graphs (plus one master process). In practical situations, the tool is executed with tens of thousands of random graphs for better accuracy. Therefore, from scalability perspective, our approach is scalable on processors/cores upto the number of random graphs being used in the problem. Since, number of processors/cores would never be higher than random graphs in actual situations, so this may not be a constraint.

Significance calculation phase: Significance calculation

```
Procedure EnumerateOriginalMaster
Require numTasks, numProc
1:  for all workers do
2:     sendMessage(rootNode.targetGraph, workerRank,
WorkTag)
3:  whilenotFinished(targetGraph) do
4:     receiveMessage(rank)
5:     sendMessage(nextrootNode. targetGraph, rank, WorkTag)
6:  for all workers do
7:     receiveMessage(rank)
8:  for all workers do
9:     sendMessage(Null, rank, FinishTag)
10:    receiveMessage(PatriciaStringsList)
11:    receiveMessage(ZeroOneCanLabelsList)
12:    aggregate(PatriciaStringsList, ZeroOneCanLabelsList)
13: return 0
```

Fig 4. Master process - Enumeration of original network

```
Procedure EnumerateOriginalWorker(g)
Require: original graph
foreverloop
    receiveMsgMaster(rootNode, Tag)
    if (FinishTag(Tag))
      inorder(patriciaTrie, patStrList, canLabList)
      SendMsgMaster(patStrList)
      SendMsgMaster(canLabList)
      return 0
    EnumerateandCensus(rootNode, subgraphsize)
    sendMsgMaster(Null)
```

Fig. 5. Worker process - Enumeration of original graph.

was performed by the master process after completion of the original graph and random graph phases. Master process calculates z-scores of all the non-isomorphic subgraph classes to identify the significantly overrepresented subgraphs in original network or network motifs.

RESULTS AND DISCUSSION

We evaluated the performance of parallel implementation of algorithm on a distributed memory High Performance Cluster (HPC). The HPC had 32 blade servers as compute nodes having total 512 computing cores of Intel Xeon (sandy bridge) processor running at 2.6 GHz and 4 GB RAM per core which accumulates to 2 TB RAM. A maximum of 10 processors with 12 cores each were available for the study. The tool was implemented using MPI and C and it is suitable for high performance clusters with distributed memory. We used networks of *E.coli*, electronic and social network for experimental studies (Table 1).

Table 1. Description of networks

Network	Directed/ Undirected/ Both	Vertices	Edges	Description
E.coli	Directed	672	1275	Metabolic pathway of *E.coli* (http: //www.genome.jp/kegg/)
Social	Directed	67	182	Real social network (Kashani *et al.* 2009)
Electronic	Both	252	399	Real electronic network (Milo *et al.* 2002)

Table 2. Processing time for directed graphs

No of processors	Time (Seconds) (r=100)		
	E.coli (s=9)	Elec (s=10)	Social (s=9)
1	24757.62	5885.04	2474.71
4	7429	1981	904.76
8	3255	901.45	427.39
16	1679.71	462.27	246.9
32	958.19	249.13	149.21
64	636.86	168.89	97.89
101	487.82	119.19	65

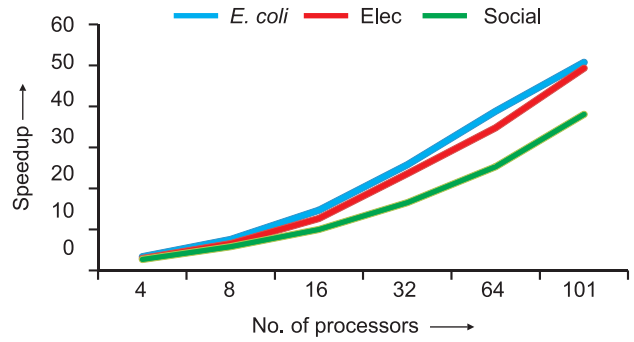

Fig 6. No. of processors vs speedup

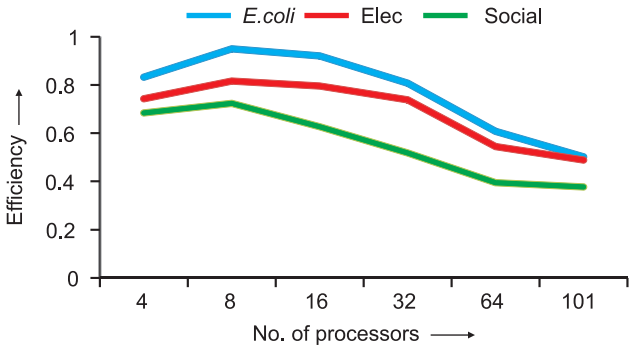

Fig 7. No. of processors vs efficiency

The graphs (Figs 6 and 7) present the speedups achieved in runtime and efficiency from execution of tool scaling from 1 to 101 processors/cores.

The tool was run on 1, 4, 8, 16, 32, 64 and 101 processors/cores for motif size (s) = 9 in case of *E. coli* and social networks and s=10 for electronic network. The wall clock time from start till end was recorded (Table 2). When the tool was executed on single core, the performance was equivalent to sequential search. It was observed that with parallelization of fast network motif search algorithm, maximum speedup upto 50.75, 49.37, 38.07 on *E.coli*, electronic and social networks was obtained with efficiency of 50%, 48.88% and 37.69% respectively when executed on 101 processors. The algorithm showed peak efficiency for all the three graphs when executed on 8 cores. The tool is scalable to higher processors for higher number of random networks. This parallel tool would be useful for large network motif search for studies on complex networks, as it helps to reduce the runtime considerably as compared to sequential algorithm.

In this paper, we implemented parallel algorithm for large network motif search using master-worker parallelization paradigm with dynamic load balancing. We used PATRICIA trie a compressed data structure along with the ESU sequential algorithm. We achieved speedup upto 50.75, 49.37, 38.07 on *E.coli*, electronic and social networks to achieve faster response time as compared to sequential algorithm for fast network motif search implemented using PATRICIA trie. As the memory cost of PATRICIA is less, it is possible to use the tool for searching bigger network motifs without making it infeasible in shared environments where very high memory (RAM) is not available.

Implementation: Parallel tool was implemented in C programming language using MPI (Figs 1 and 3–5). For comparison, sequential version of the algorithm was used which was also implemented in C++.

ACKNOWLEDGEMENT

Authors express their deep sense of gratitude to the Founder President of Amity University, Dr Ashok K. Chauhan, for his keen interest in promoting research in the Amity University, who has always been an inspiration for achieving greater heights. Authors also thank Bioinformatics Resources and Applications Facility (BRAF), C-DAC, Pune for providing the supercomputing facility for carrying out the experimental study.

REFERENCES

Guy Karlebach and Ron Shamir. 2008. Modelling and analysis of gene regulatory networks, *Nature Reviews Molecular Cell Biology* **9**: 770–80.

Grochow J A and Kellis M. 2007. Network motif discovery using sub-graph enumeration and symmetry-breaking. *RECOMB*, pp 92–106.

Himanshu, Chaturvedi K K, Bandyopadhyay A and Jain Sarika. 2017. PATRICIA Trie based time and memory optimization for fast network motif search. *Indian Journal of Animal Sciences* **87**(4): 512–19.

Khakabimamaghani S, Sharafuddin I and Dichter N. 2013. QuateXelero: an accelerated exact network motif detection algorithm. *PLoSOne* **8**(7): e68073.

McKay B D. 1981. Practical graph isomorphism. *Congressus Numerantium* **30**: 45–87.

Milo R, Shen-Orr S and Itzkovitz S. 2002. Network motifs: Simple building blocks of complex networks. *Science* **298**: 824–27.

Rauber T and Rünger G. 2009. Parallel Programming. *Springer Berlin Heidelberg*. p196. doi: 10.1007/978-3-642-04818-0_4.

Ribeiro P, Silva F and Lopes L. 2010. Efficient parallel subgraph counting using G-tries. *Proceedings of IEEE International Conference on Cluster Computing* pp. **2**: 17–26.

Ribeiro P, Silva F and Lopes L. 2011. Parallel discovery of network motifs. *Journal of Parallel Distribution Computers [Internet]. Elsevier Inc.* **72**(2): 144–54.

Schatz M, Cooper-Balis E and Bazinet A. 2008. *Parallel Network Motif Finding.*

Shen-Orr S, Milo R, Mangan S and Alon V. 2002. Network motifs in the transcriptional regulation network of *Escherichia coli.*

Nature Genetics 31(1): 64–68.

Tran, N, Mohan S, Xu Z and Huang C. 2014. Current innovations and future challenges of network motif detection. *Briefings in Bioinformatics* **16**(3): 497–525.

Wang T, Touchman J W, Zhang W, Suh E B and Xue G. 2005. A parallel algorithm for extracting transcription regulatory network motifs. *IEEE International Symposium on Bioinformatics and Bioengineering*, pp. 193–200.

Wernicke S and Rasche F. 2006. FANMOD: A tool for fast network motif detection. *Bioinformatics* **22**(9): 1152–53.

Batagelj M and Mrvar A. 2006. *Pajek Datasets*. Available at http://vlado.fmf.uni-lj.si/ pub/networks/data/

Newman M. 2009. *Network Data*. Available at: http://www-personal.umich.edu/˜mejn/netdata/