Indian Journal of Animal Sciences **87** (4): 512–519, April 2017/Article

PATRICIA trie based time and memory optimization for fast network motif Search

HIMANSHU[1], K K CHATURVEDI[2], A BANDYOPADHYAY[3], SARIKA JAIN[4]

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

Received: 23 August 2016; Accepted: 12 September 2016

ABSTRACT

Network motif search is useful in uncovering the important functional components of complex networks in biological, chemical, social and other domains. PATCOMP - a PARTICIA based novel approach for network motif search is proposed in this paper. The algorithm of PATCOMP takes benefit of memory compression and speed of PATRICIA trie to store the collection of subgraphs in memory and search them for classification and census of network. The structure of trie nodes and how data structure is developed to use it for counting the subgraphs is also described. PATCOMP was compared with QuateXelero and G-Tries.The main benefit of this approach is significant reduction in memory space requirement particularly for large network motifs with acceptable time performance. The experiments with directed networks like *E.coli*, yeast, social and electronic validated the advantage of PATCOMP in terms of reduction in memory usage by 2.7–27.7% as compared to QuateXelero for smaller motif sizes (with exceptions of s=6 for *E. coli* and s=6 for social), and 7.8–38.35% for larger motif sizes. For undirected networks, PATCOMP utilizes less memory by 0.07%-43% (with exception of s=7 for electronic and s=6,8 for dolphin networks).

Key words: Algorithms, Bioinformatics, Complex networks, Data structures, Network motifs, Optimization, PATRICIA

Network motif searching has many applications including its use in bioinformatics in identification of genes responsible for diseases in humans and animals and biotic and abiotic stresses in plants etc. Motif detection can be applied for the identification of TFBS using the nucleotide sequences of livestock species (Rao *et al.* 2014), genetic improvement of livestock species, drug discovery etc. 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 (Milo *et al.* 2002). Some network motifs are depicted in Fig. 1. In graph theoretic terms, networks are described as graphs with nodes representing vertices and interconnections represented as edges. Therefore, network motif is a subgraph in a larger graph, which is statistically over represented than in a set of random networks having similar properties. Network motif is considered as structural building block of several natural networks specifically in biological networks like protein-protein interaction, transcription regulation network, disease gene networks, gene regulation networks etc. They are also believed to have important functional roles besides being structural components of complex networks. Study of network motifs focused not only on biological networks but also on other networks like biochemical, neurobiological, electronic and ecological networks (Milo *et al.* 2002). They observed that network motifs present in different types of networks were unique to them and therefore network motif analysis could be an important approach to uncover basic building blocks of most of the networks.

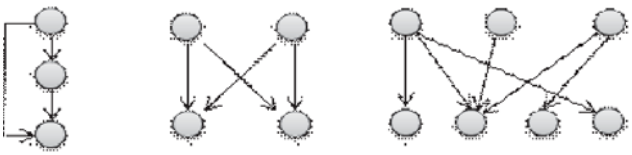

Fig. 1. Examples of network motif (a) Feed forward loop, (b) Bi-Fan, (c) Multi input module.

Network motif searching is np-hard problem. It requires subgraphs to be matched with desired motifs, which involves graph isomorphism checking. Graph isomorphism checking is a problem in np and is not known to be complete, whereas subgraph isomorphism checking is known to be np-complete. Network motif search solutions can be broadly categorized into 'exact motif search' and 'approximate motif search' depending upon the classification and enumeration of subgraphs carried out with or without sampling

Present address: [2]ICAR-Indian Agricultural Statistics Research Institute, New Delhi. [3]ITRA, Ministry of Communication and Information Technology, New Delhi 110 030 India.

performed on main graph. Exact motif search is more computation intensive but results are accurate, whereas approximate approach based solutions are fast with accuracy compromised.

Further the solutions are classified as 'network centric' or 'motif centric' depending upon whether the k-size subgraphs to be searched are enumerated from original network and then used for census on original and random networks as in former class of solution. The alternative approach used for later class of solutions is to first generate all different non-isomorphic classes for the specified motif size and then calculate the frequency of each in the network (i.e. count the number of matches of each class in the network). The drawback of network centric approach is that it requires checking the isomorphism of each enumerated subgraph and as the number of non-isomorphic classes grows exponentially with the increasing size of the subgraph, it results in wastage of time for checking isomorphism of large number of subgraphs, which is time consuming. The drawback of motif-centric solutions is that it may spend unnecessary time for checking subgraphs that may not be present in the target network (Wong *et al.* 2001). FANMOD (Wernicke and Rasche 2006, KavoshKashani *et al*. 2009), G-Tries (Ribeiro and Silva 2010) and QuateXelero (Khakabimamaghani *et al*. 2013) were developed based on exact network motif search approach. The algorithm proposed in this paper is also based upon the exact network motif search. FANMOD can detect motifs of size upto k=8 very fast. It is based upon Rand-ESU algorithm. FANMOD also implements NAUTY, the canonical graph labelling algorithm (McKay 1981) for grouping subgraphs into isomorphic subgraph classes. Kavosh, an algorithm developed for fast k-motif search with less memory usage, has four steps viz. enumeration, classification, random graph generation and motif identification. It could search network motifs upto size 10 for *Escherichia coli*, *Saccharomyces cerevisiae* (yeast) and social networks and up to size 12 for electronic network in satisfactory time. G-Tries enumerates using ESU and G-Tries methods and builds graph trie data structure for the sub graphs. It is used for classification of the original network and random networks. It can search network motifs of size k=3 and more. Its memory usage also increases noticeably with increase in motif size.

QuateXelero makes use of Quaternary tree data structure during enumeration. The quaternary tree is built as the enumeration progresses and it is also used for classification and census on original and random networks. Although QuateXelero is always faster in census on original networks as compared to ESU of G-Tries. It is also generally faster in census on random networks for small motifs. G-tries is fast for census on random network. Overall, QuateXelero has best time performance, however, its main drawback is high memory usage.The quaternary tree based approach trades speed for high memory usage by algorithm. Due to high memory usage, it is possible to use QuateXelero only on high-end computing facilities with huge memory for large motif (k=11, 12) discovery. This paper explores the impact of memory space optimization using PATRICIA (Practical Algorithm to Retrieve Information Coded in Alphanumeric) trie for network motif search so that the tool could be used for large motif detection using less memory within satisfactory time performance. The tool would benefit researchers who do not have access to high-end computing resources and it could also be useful for motif search for size unachievable before with the same computing resources. Owing to reduced memory space requirement, the algorithm would be more suitable for parallelization.

MATERIALS AND METHODS

PATRICIA, a radix trie, has exactly 'n' number of nodes for 'n' elements, so it stores a string exactly once. PATRICIA search is a binary algorithm and while searching for a search key, branching happens based upon the bit of its binary representation. Search path is followed by examining the bits of the search key present at the bit index position of each node along the path. If the bit is '1', right branch is followed and in case of '0', left branch is followed. Bit index of a node is a bit position in the binary representation of key stored in that node. Structure of each node of the trie used in PATCOMP is depicted in Fig. 2. The algorithm uses a binary tree to store the classified non-isomorphic subgraphs in the original and random graphs, as also in QuateXelero.

Enumeration: For enumerating all subgraphs of size 'k' in a given network, PATCOMP algorithm uses procedure similar to the one in FANMOD, algorithm. In FANMOD, the subgraph is extended by one vertex in each step in the enumeration phase (Fig. 3). As the subgraphs are enumerated one by one using ESU, a binary key string is formed using a numeral for each type of four possible types of connectivity between the vertices. Character code '1' indicates one-way connection from the existing vertex to added vertex, '3' stands for a one-way connection in the reverse direction, '4' indicates no connection between them, and '2' represents a two-way connection. Motif size, number

```
Key : string;
bitindex : integer;
leafPtr : pointer to corresponding leaf of binary tree
holding classified non-isomorphic subgraphs
```

Fig. 2. Description of a node of PATCOMP algorithm's Trie data structure.

Table 1. Motif size and corresponding trie node string size

Motif size	3	4	5	6	7	8	9	10	11	12	13
Node string size	3	6	10	15	21	28	36	45	55	66	78

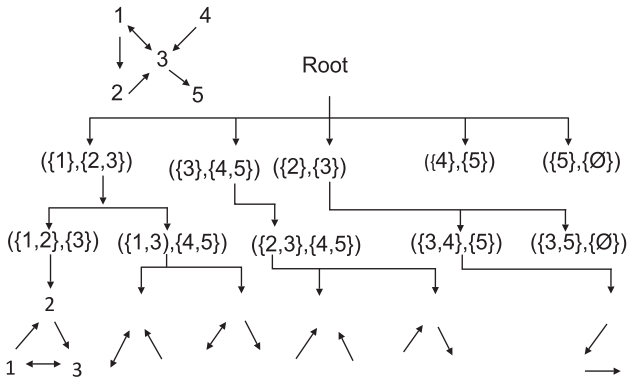

Fig. 3. Recursive search tree as generated by implementation of ESU algorithm for the given 5-node graph. (Adapted from Ribeiro 2009).

of bits in the key and memory required to store the key string (Table 1).

Classification: During the enumeration, as the k-size subgraph is enumerated, its key string formed as described in enumeration above is first searched in the trie. If the key is found, it returns a pointer to the leaf containing counter of the corresponding subgraph in the binary tree, which is increased by one and updated. In case, the key string is not found in the trie, a call to 'Nauty' is made to classify the subgraph. With the help of canonical label of the subgraph obtained from 'Nauty', search is performed on binary tree and the corresponding leaf in the binary tree is identified. Subgraph counter of the identified leaf, which signifies the number of subgraphs found in that class in the network, is increased by one. The key string is then inserted in the trie along with a pointer to the leaf in trie, set to the identified leaf of the binary tree. This process is repeated until the complete graph is enumerated. For the census on the original network, the binary tree would be modified when a new non-isomorphic class is found. However, for the random networks census, the structure of the binary tree is not changed. The binary tree is searched until there is a node, which is null or it is a leaf. Null node means that the recently enumerated subgraph is of a new isomorphic class, non-existent in the original network, hence the subgraph is ignored. In the case of a leaf, the counter in the binary tree corresponding to the leaf is increased by one.

Motif detection: Census on original network results in count of all non-isomorphic subgraph class in the leaves of binary tree. The frequencies of subgraphs are also computed for a set of specified number of 'm' random graphs which are mostly generated by using Markov-chain Monte Carlo edge switching method. For determining statistical significance of the identified subgraphs, Z-score is calculated as follows:

$$Z\text{–score}_i = \frac{f_{Gi} - \text{avg}\,(f_{Ri})}{\text{std dev}\,(f_{Ri})}$$

where f_{Gi}, avg (f_{Ri}) and std dev (f_{Ri}) are the count of i^{th} subgraph in original network, average number of occurrences of i^{th} subgraph in random graphs and standard deviation of occurrences of i^{th} subgraph in random networks, respectively.

Datasets: We have used standard networks viz. the metabolic pathway of *E. coli* bacteria, transcription network of yeast (*S. cerevisiae*), a real social network, a real electronic network, protein-protein interaction of budding yeast and frequent assoiations between a group of dolphins (Table 2) for validating our algorithm as used in Khakabimamaghani *et al*. (2013) for comparison of QuateXelero with other algorithms. As in QuateXelero, we eliminated the self-loops from all the networks. Test results were grouped into directed networks (small motifs and large motifs) and undirected networks. In present paper, PATCOMP was compared with G-Tries and QuateXelero.

RESULTS AND DISCUSSION

G-Tries and QuateXelero are among fastest network motif searching tools available for the standard networks and small and large motif sizes. Whereas, G-Tries takes very long time for census on original networks, QuateXelero trades speed for very high memory requirement.The PATRICIA based approach was compared with QuateXelero and G-Tries. For undirected networks and smaller motif search indirected networks, all three tools were run on the same computer, having Quad-Core Intel Xeon Processor ES2650, 2.6 GHz, 64 GB main memory and MS Windows 2008 R2 installed with Cygwyn (64-bit). For experiments with larger motif search in directed networks, a symmetric multiprocessing system (SMP) having 64-core Intel Xeon ® CPU E-7 2830, 2.13 GHz processor, 1.5 TB RAM and RHEL6 64-bit OS, available at ICAR-Indian Agricultural

Table 2. Description of networks

Network	Directed/ Undirected/ Both	Vertices	Edges	Description
Yeast	Directed	688	1079	Transcription network of yeast (Alon U 2002)
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)
YeastPPI	Undirected	2361	6646	Protein-protein interaction of budding yeast (Batagelj and Mrvar 2006, Bu *et al.* 2003)
Dolphin	Undirected	62	159	Frequent associations between a group of dolphins (Lusseau *et al.* 2003, Newman 2009)

Algorithm 1: PATCOMP (Original Network)

Input: Graph G a positive integer k
Output: k subgraphs census of graph G

1: For all vεV(G)
2: $V_{Ext} \leftarrow \{u\varepsilon N(V): u>v\}$
3: EXTENDSUBGRAPH(V_{Subg}, V_{Ext}, v, subgStr)
4: **procedure**EXTENDSUBGRAPH(V_{Subg}, V_{Ext}, v, subgStr)
5: if $|V_{Subg}|$=k then
6: Leaf1←Lookup(patTrie, subgStr) //Search the Patricia trie for subgraph string subgStr and return pointer to the Bleaf to which the subgraph string node pointer points
7: if Leaf1=NULL then //Only in this case it is required to call NAUTY
8: Leaf1 ←BLeaf(CANONICALLABELING(V_{Subg})) // BLeaf returns a pointer to corresponding leaf in the binary tree
9: INSERTSUBGRSTRINPATTRIE(patTrie, subgStr, Leaf1)// Insert new node in Patricia Trie with new string and a pointer to corresponding leaf in the binary tree
10: INCREMENTCOUNT(Leaf1.pointer) //Increases the counter of BLeaf to which the subgraph string node pointer points
11: else
12: while $V_{Ext}\neq\varnothing$ do
13: remove random chosen w$\in V_{Ext}$
14: subgStr←MAKESTR(V_{Subg}, w, subgStr) //Make subgraph string
15: $V'_{New}\leftarrow\{u\in N_{Exclusive}(w, V_{Subg}): u>v\}$
16: $V'_{Ext}\leftarrow V_{Ext}\cup V'_{New}$
17: EXTENDSUBGRAPH($V_{Subg}\cup\{w\}$, V'_{Ext}, v, subgStr')
18: procedure MAKESTR(V_{Subg}, w, subgStr) returns ResultStr
19: for all u$\in V_{Subg}$ do
20: if (u, w)$\in$E(G) and (w, u)$\in$E(G) then
21: Append '2' to the subgStr
22: else if (w, u)$\in$E(G) then
23: Append '1' to the subgStr
24: else if (u, w)$\in$E(G) then
25: Append '3' to the subgStr
26: else
27: Append '4' to the subgStr
28: return subgStr

Statistics Research Institute, New Delhi, was used.

The results of experiments for smaller motif size (s=5,6,7), of directed networks are listed in Table 3. The number of random networks was set to 100 in experiments for all networks except yeast. The number of random networks for yeast was set to 10 as it takes very long time to complete with 100 networks. It was seen that, while PATCOMP was superior to G-Tries in terms of speed, memory space requirement of PATCOMP was less than QuateXelero in general and less than G-Tries in case of *E.coli* and social network. The savings in memory increase with increasing motif size. It is worth noting that

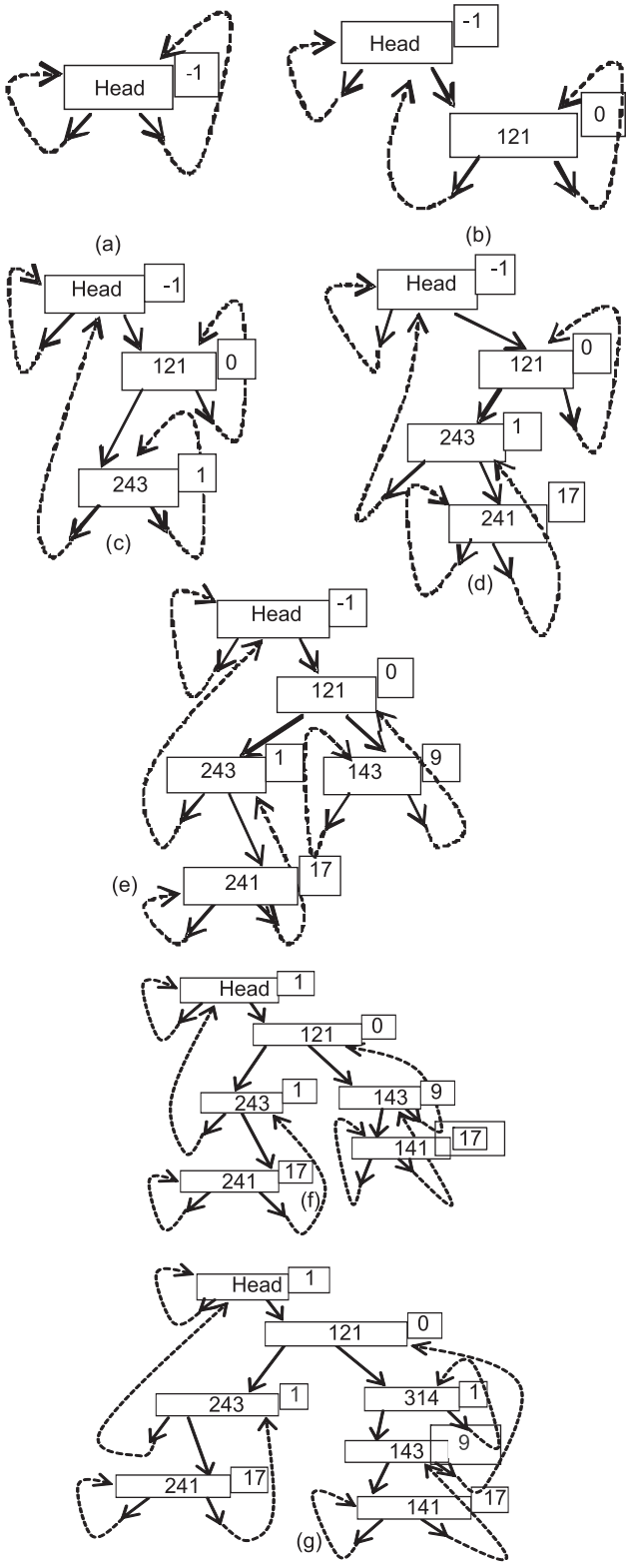

Fig. 4. (a) to (g) Step-by-step construction of PATRICIA trie during enumeration and classification of a sample graph in Fig 3 for 3-node motifs in PATCOMP. Keystring is composed of digits representing each type of connectivity present among the nodes in the subgraph being enumerated. Number in the square at the top corner of keystring indicates bit-index.

Table 3. Performance of G-Tries, QuateXelero and PATCOMP for smaller motifs (s = 5, 6, 7)

Network	s	Time (Original Network) (sec)			Avg. Time (Random Network) (sec)			*Total Time (sec)			Memory (MB)		
		G-Tries	QuateXelero	PATCOMP	G-Tries	QuateXelero	PATCOMP	G-Tries	QuateXelero	PATCOMP	G-Tries	Quate Xelero	PATCOMP
E. coli	5	0.09	0	0	0.027	0.00376	0.0095	3	0.531	1.125	3.6	11.9	8.6
	6	0.757	0.062	0.109	0.298	0.028	0.084	31	3.28	9	20.5	21.4	22.7
	7	6.95	0.484	0.84	3.67	0.242	0.78	380	26.04	81.15	209.1	137.9	116.0
Yeast	5	4.12	0.078	0.171	0.054	0.0828	0.2	5	0.906	2.203	1.4	1.7	1.6
	6	71.81	0.95	2.42	0.98	1.512	4.12	82	16.125	43.75	2.4	3.5	3.1
	7	1689.41	14.03	40.5	16.66	30.65	90.48	1856	320.719	945.59	8.5	16.2	14.6
Social	5	0.012	0	0	0.0035	0.00125	0.00265	0.37	0.156	0.296	3.9	3.7	3.6
	6	0.083	0.016	0.031	0.023	0.00889	0.022	4	1.187	2.375	46.3	14.3	21.8
	7	0.156	0.265	0.25	0.554	0.081	0.159	24	9.984	18.25	273.8	256.8	235
Electronic	5	0.029	0	0	0.00157	0.001	0.00187	0.24	0.14	0.218	1.2	4.6	4.4
	6	0.157	0	0	0.0074	0.0051	0.106	1.3	0.562	1.109	2.4	3.9	3.5
	7	1.106	0.046	0.063	0.036	0.032	0.064	5	3.14	6.64	9.6	13.2	12.2

*No. of random graphs (r) is 100 for *E. coli*, social and electronic networks; for yeast, number of random networks (r) is 10

Table 4. Performance of G-Tries, QuateXelero and PATCOMP for larger motifs (s = 9, 10, 11 and more)

Network	s	Time (Original Network) (sec)			Avg. Time (Random Network) (sec)			*Total Time (sec)			Memory (MB)		
		G-Tries	QuateXelero	PATCOMP	G-Tries	QuateXelero	PATCOMP	G-Tries	Quate Xelero	PATCOMP	G-Tries	Quate Xelero	PATCOMP
E. coli	9	7.39	46.67	76.86	1121.41	36.67	89.94	6426	244.26	549.56	1.17 GB	2.03 GB	1.87 GB
	10	6530.98	353.39	599.84	12109	320.41	793.01	67626	2081.75	4701.74	7.22 GB	16.23 GB	14.27 GB
	11	56223.06	2577	4713	156247.08	3539	6368.85	841509	21102	37540.44	47.42 GB	122.71 GB	87.61 GB
Avg. growth ratio		446.18	7.43	7.83	11.85	9.89	8.42						
Yeast	9	909844.41	5198	16109	7444.17	20618.02	68050.77	946934	108291.29	356269.5	230.18 MB	942.35 MB	860.44 MB
Social	9	25.05	9.81	11	9.23	12.08	16.18	170	84.47	108	1.29 GB	2.76 GB	2.5 GB
	10	134.36	55.94	67.99	50.81	85.24	111.1	998	567.26	746.53	7.34 GB	18 GB	15.02 GB
	11	814.39	290	366.09	214.65	543.01	629.98	5259	3633.63	4158.73	39.39 GB	107.8 GB	75.54 GB
Avg. growth ratio		5.71	5.44	5.78	4.86	6.71	6.26						
Electronic	9	67.1	2.41	3.66	1.94	2.8	4.39	79	16.8	26.13	34.64 MB	102.05 MB	61.89 MB
	10	493.11	14.26	22.02	10.54	17.83	27.63	558	105.19	163.01	110.25 MB	702.25 MB	548.87 MB
	11	4067	167.86	129.42	67.41	217.51	184.28	4473	1267.72	1069.41	933.16 MB	4.6 GB	3.22 GB
	12	36648.93	1052.72	789.33	335.32	1798.5	1258.33	38720	10134.42	7148.3	4.9 GB	24.35 GB	18.40 GB
	13	388123.41	6370.54	4992.95	2136.26	12280.98	8870.55	401161	68189.79	49798.73	30.35 GB	143.18 GB	104.71 GB
Avg. growth ratio		8.80	7.50	6.08	5.79	8.41	6.71						

Table 5. Performance of G-Tries, QuateXelero and PATCOMP with 100 random networks

Network	Algorithm	Time (T)	T/T_{max}	Memory (m) (GB)	m/m_{max}
E. coli (s=9)	G-Tries	380	1	209.1	1
	QX	26.04	0.068	137.9	0.659
	PATCOMP	81.15	0.213	116	0.554
Yeast (s=7)	G-Tries	1856*	1	8.5	0.524
	QX	320.719*	0.172	16.2	1
	PATCOMP	945.59*	0.509	14.6	0.901
Social (s=9)	G-Tries	24	1	273.8	1
	QX	9.984	0.416	256.8	0.937
	PATCOMP	18.25	0.760	235	0.858
Electronic (s=9)	G-Tries	5	0.753	9.6	0.727
	QX	3.14	0.472	13.2	1
	PATCOMP	6.64	1	12.2	0.924

*Value in sec.

QuateXelero was fastest in all cases, which is achieved at the cost of high memory.

For, larger motif size (s=9,10,11 and higher) of directed networks, the number of random networks was set to 5 for all the experiments to save the time. Although it was not sufficient for motif detection, it was possible to get the results for comparison purpose. The results revealed that PATRICIA based approach was faster than G-Tries for census on original networks in case of all directed networks, also the time for census on random networks was less than G-Tries in case of *E.coli* (Tables 3, 4). In case of social, yeast and electronic networks, performance of PATCOMP was better vis-à-vis memory utilization as compared to QuateXeloro.

To analyse the space, time trade-off, all the three tools were run for motif size (s=9) with 100 random networks for all four networks. Results of total time and memory space used are listed in Table 5. Space vs Time graph in effectively demonstrates better optimization in case of PATCOMP as compared to QuateXelero for all the networks (Fig. 4). Similar results are expected for larger motif sizes also.

In case of undirected networks, the number of random networks were fixed at 100 for all experiments. The results are listed in Table 6. The results revealed that memory space utilization of PATRICIA was less than Quaternary tree in general.

CONCLUSION AND FUTURE WORK

In this paper, we introduced a novel network motif searching algorithm, PATCOMP with the objective of optimizing the space, time trade-off. The PATCOMP algorithm is based upon ESU of FANMOD. The results are promising. Specific outcomes experiments are discussed in the succeeding paragraph.

In general, the PATCOMP utilizes less memory by 2.7–27.7% as compared to QuateXelero for directed networks for small motif sizes (with exceptions of s=6 for *E. coli* and s = 6 for social), and memory utilized by PATCOMP is less by 7.8 – 38.35% for directed networks for large motif sizes.

For undirected networks, PATCOMP utilizes less memory by 0.07%-43% (with exception of s =7 for electronic and s=6,8 for dolphin networks).

For directed networks, time taken by PATCOMP is more by 1.55x–3.11x that of QuateXelero for small motif sizes and 0.7x–3.28x for larger motifs. Processing time for census on original network is in general less than G-Tries for Network motif finding for directed networks (as also is the

Table 6. Performance of QuateXelero and PATCOMP

Network	s	QuateXelero				PATCOMP			
		Original Network (sec)	Avg. Time Random Networks (sec)	*Total Time (sec)	Memory	Original Network (sec)	Avg. Time Random Networks (sec)	*Total Time (sec)	Memory
YeastPPI	4	0.046	0.055	6.87	168.66 MB	0.078	0.085	10.1	167.89 MB
	5	1.26	1.74	177.31	168.67 MB	2.29	3.08	311.82	168.55 MB
	6	37.72	63.83	6422.65	168.80 MB	70.09	118.74	11946.15	168.80 MB
Electronic	6	0	0.008	1.06	1.82 MB	0.031	0.012	1.35	1.03 MB
	7	0.031	0.025	2.62	6.33 MB	0.046	0.047	4.83	8.55 MB
	8	0.14	0.15	15.319	11.94 MB	0.23	0.31	31.62	11.12 MB
	9	0.73	0.93	94.25	31.77 MB	1.4	2.07	209.21	24.67 MB
	10	4.22	5.35	539.89	131.57 MB	8.33	12.04	1213.04	101MB
Dolphins	6	0.015	0.01	2.949	1.12 MB	0.015	0.018	1.87	3.02 MB
	7	0.062	0.078	7.9	7.74 MB	0.07	0.14	14.38	7.55 MB
	8	0.4	0.57	58.56	60.79 MB	0.54	1.09	110.93	66.03 MB
	9	2.37	5.57	364.57	765.82 MB	3.33	7.43	752.86	747.07 MB
	10	14.11	22.97	2352.97	9.16 GB	18.81	46.29	4704	8.23 GB

* No of random networks = 100

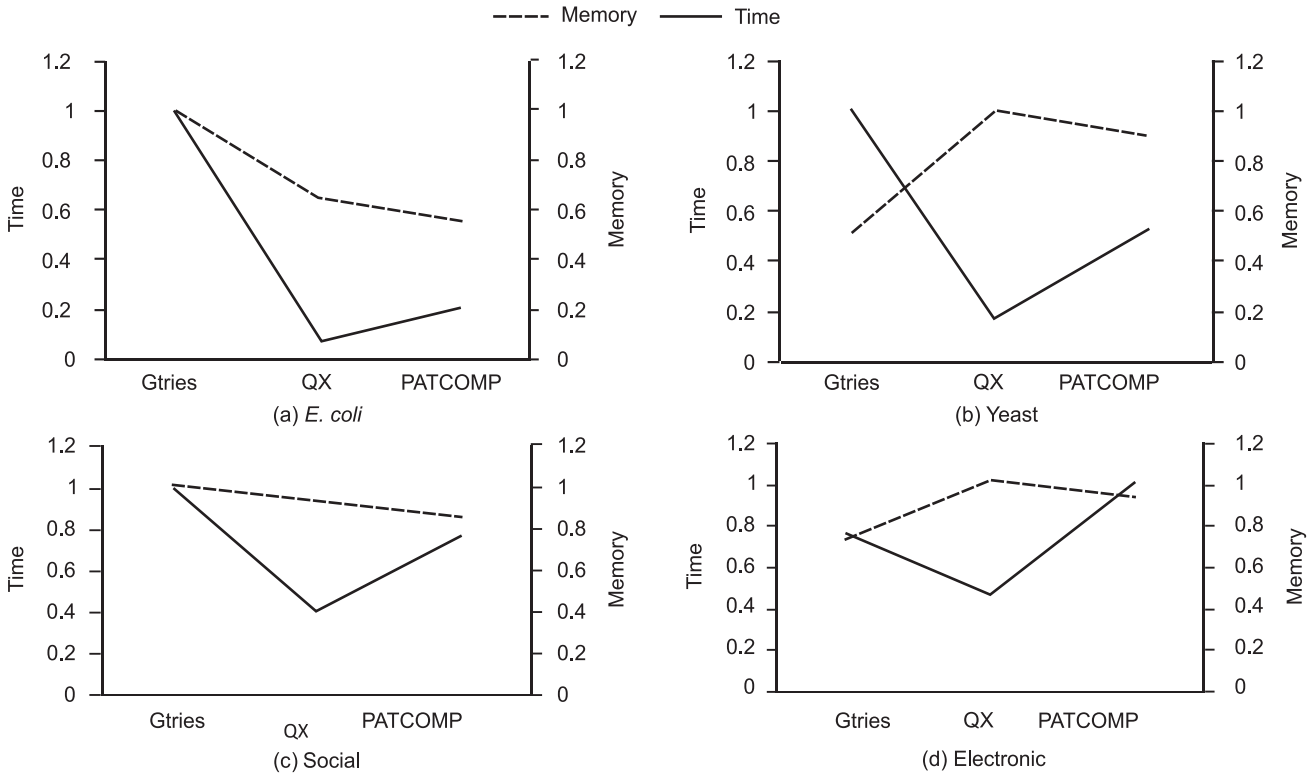

Fig. 5. Time Vs Memory plot of G-Tries, QuateXelero and PATCOMP.

case with QuateXelero).

For undirected networks, time taken by PATCOMP is more by 0.6x – 2.24x that of QuateXelero.

In PATCOMP algorithm, it optimizes the space time trade-off in view of the very high memory requirement of QuateXelero and slow speed of G-Tries for *E. coli*, yeast and social networks (directed).

PATCOMP is better in general as compared to QuateXelero for all directed networks as regards memory cost.

For electronics network, PATCOMP is both time efficient as well as memory efficient for large motifs i.e. k = 11, 12, 13. Since, average growth ratio of PATCOMP for random networks is lesser in comparison to G-Tries and QuateXelero, it is expected to be faster for larger motifs.

In view of the memory vs time optimization achieved by using the PATCOMP for network motif search, it is expected that parallel version of the algorithm will perform better as compared to QuateXelero. Therefore, we plan to implment parallel version of the algorithm in future.

IMPLEMENTATION

PATCOMP was implemented in C++ programming language as described in algorithm-1. PATRICIA is implemented as per the details in Robert (1983). For comparison, QuateXelero source code was downloaded from http://lbb.ut.ac.ir/Download/LBBsoft/QuateXelero and G-Tries source code from http: //www.dcc.fc.up.pt/gtries/.

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 and has always been an inspiration for achieving greater heights. Authors also thank ICAR for providing the supercomputing facility (ASHOKA) for carrying out this experimental study.

REFERENCES

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

Himanshu and Jain S. 2016. Impact of memory space optimization technique on fast network motif search algorithm. ICCCCS 2016, Ajmer, India (accepted). Springer.

Kashani Z R, Ahrabian H, Elahi E *et al*. 2009. Kavosh: a new algorithm for finding network motifs. *BMC Bioinformatics* **10**: 318.

Khakabimamaghani S, Sharafuddin I, Dichter N *et al*. 2013. QuateXelero: an accelerated exact network motif detection algorithm. *PLoS One* **8**(7).

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

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

Robert S. 1983. Algorithms. Addison Wesley.

Ribeiro P, Silva F and Kaiser M. 2009. Strategies for network motifs discovery. Proceedings of the 5th IEEE International

Conference on E-Science, IEEE CSPress, Oxford, UK.

Ribeiro P and Silva F. 2010. Efûcient subgraph frequency estimation with G-tries. International Workshop on Algorithms in Bioinformatics (WABI), *LNCS* **6293**: 238–49. Springer.

Rao A R, Dash M and Sahu T K *et al*. 2014. Statistical and bio-computational applications in animal sciences. *Indian Journal of Animal Sciences* **84**(5): 475–89.

Schreiber H, and Schwobbermeyer. 2004. Towards motif detection in networks: frequency concepts and flexible search. Proceedings of the International Workshop on Network Tools and Applications in Biology 91–102. Camerino, Italy.

Warnicke S. 2006. Efficient detection of network motifs. *IEEE/ ACM Transactions on Computational Biology and Bioinformatics* **3**(4): 347–59.

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

Wong E, Baur B, Quader S *et al*. 2001. Biological network motif detection: principles and practice. *Briefings in Bioinformatics* **13**(2): 202–15.

The *E. coli* Database. Available: http: //www.genome.jp/kegg/

Alon U. 2002. The *S. cerevisiae* Database. Available: http: // www.weizmann.ac.il/mcb/ UriAlon.

Bu D, Zhao Y, Cai L *et al*. 2003. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Research* **31**: 2443–50.

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

Lusseau D, Schneider K, Boisseau O J *et al*. 2003. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. can geographic isolation explain this unique trait? *Behavioral Ecology and Sociobiology* **54**: 396–405.

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