

A Parallel Noise-Reducing Sampling Algorithm for Large-Scale Network

Kanimathi Duraisamy, Department of Computer Science, University of Nebraska at Omaha

1. Problem and Motivation

Large-scale networks are extensively used to model the complex interactions between various entities in data-intensive domains such as social sciences [1], bioinformatics [2] and software engineering[3]. Networks (or graphs) are formed of vertices and edges, where vertices represent the entities and the edges represent the relation between them. Analysis of large-scale networks is a crucial tool in exploring and understanding the behavior of complex systems.

There exist two important challenges in developing efficient and accurate network analysis methods. First, most networks are extremely large and therefore analyzing these systems is both computation and memory intensive. This issue can be resolved by (i) using high performance computing to divide the data and algorithm over multiple processing units [4,5,6] or (ii) by sampling [7,8,9], that is, extracting a representative subgraph that exhibits similar characteristics to the original larger network.

A more insidious problem concerns noise in networks. Real-world networks are built using experimental (such as gene correlation networks) or subjective (census reports, epidemic distribution) techniques. The fluctuations and bias inherent in these methods would also be present in the form of small errors or noise within networks generated using this process. However, eliminating of noise from empirical results is an often overlooked during network analysis.

Our research involves developing a parallel network sampling technique that can filter out the noise, while preserving the important characteristics of the network. Thus, our algorithm can address both the issues; that of creating noise free networks as well as that of handling large data. Our sampling technique is based on extracting the maximal chordal subgraphs (MCS) of the original network. A chordal subgraph consists of portions of the graph where the length of a cycle cannot exceed three [10].

2. Background and Related Work

Relevant Combinatorial Terminology: We introduce some graph terminology (based on the definitions provided in [10]) that will be help in the subsequent explanation of the algorithms. A graph is a set of vertices and edges (V, E) . A vertex u is said to be a neighbor of vertex v , if they are joined by an edge. The *degree* of a vertex u is given by the number of its neighbors. A path is an alternating sequence of vertices and edges, where subsequent vertices are connected by an edge. A cycle is a path where the identical and final vertices are the same. A clique is a set of vertices that are all connected to each other.

Related Work: Graph sampling has been used for many applications, such as in obtaining spanning trees for preconditioners [11] or compressing the network to improve visualization [12]. Most sampling methods for large scale-free networks are based on random sampling, such random node selection [7] or random walks on the network [8,9], and focus primarily of preserving the combinatorial properties rather than removing noise. A parallel version of random walks is based on starting multiple walks simultaneously on different processors [13]. Parallel algorithms for obtaining spanning trees such as breadth first trees, connected components and minimum spanning trees on large-scale networks are also being investigated [4,5,6]. However the spanning tree methods focus more on graph traversal than sampling important regions. To the best of our knowledge, our method is one of the first tightly-coupled parallel network sampling algorithms [14, 15].

Filtering noise for large networks is still a largely unaddressed problem. Some recent work has focused [16, 17] on using machine learning techniques to detect noise in biological networks and uses supervised learning to predict noise based on prior information.

3. Approach and Uniqueness

We have developed a parallel graph sampling algorithm based on finding the maximal chordal subgraphs. A chordal graph is a graph where the length of a cycle is be no more than three [10]. Therefore a chordal subgraph preserves most of the cliques and highly connected regions of the network and increases the path lengths between loosely connected regions.

Finding the maximum chordal subgraph is an NP-hard problem. A sequential algorithm for obtaining the maximum chordal subgraph is provided by Dearing et. al. [18]. This method is based on growing the graph from a starting vertex and adding edges so long as they maintain the chordal characteristics. Initially the chordal subgraph consists of the starting vertex and its associated edges. In the subsequent steps, the vertex with the maximum number of visited neighbors is selected. An edge from the current selected vertex a , (a,b) is added to the chordal subgraph if the number of visited neighbors of b is a subset of the number of visited neighbors of a .

The sequential algorithm for finding MCS and indeed most of the sampling methods assume that the original network is connected. However, many real-world networks, such as our test suite of gene correlation networks have disconnected components. Based on our initial tests, we discovered that a completely chordal subgraph is a very strict restriction, and can disintegrate some clusters, that are almost, but not exact, cliques. To counteract this effect we modified the algorithm to extract quasi-chordal subgraphs, which can include few cycles of length greater than three. In order to accommodate large networks, we implemented a parallel algorithm to identify spanning quasi-chordal subgraphs.

Our primary contribution is in developing a parallel sampling technique for large-scale networks that not only extracts important combinatorial properties, but also eliminates some of the inherent noise in the networks. As seen in Section 4, reduction of noise provides additional insight to the functional properties of the underlying application. Figure 1 demonstrates how QCS based sampling can effectively select good representative samples of a large graph.

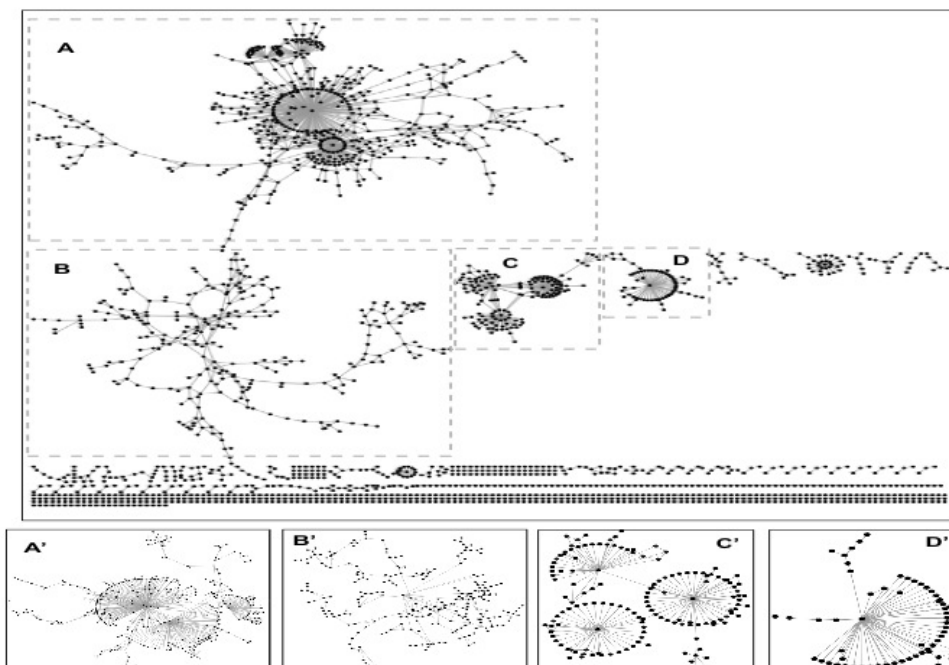


Figure 1: Gene expression network of the hypothalamus of a mouse brain with larger components highlighted in broken line boxes (A-D) and the respective chordal graph representations shown below (A'-D'). The chordal graphs preserve the structure but have significantly lower number of edges.

Parallel Algorithm for QCS: Our parallel implementation on a distributed memory system was follows: We divided the network across P processors, and identified the local maximal chordal subgraph formed only of the edges whose endpoints lie completely within a processor. For each pair of processors we identified a receiver where the synchronization would take place. We sent the border nodes to designated receiver processors. The edges that lie across processors were included only if two border edges with a common vertex combined with a previously marked chordal edge to form a triangle. This implementation generated quasi-chordal subgraphs (QCS), since the inclusion of border edges can sometimes increase the length of cycles by more than three. A detailed description can be found in [14]. Figure 2 provides the pseudocode and illustrates with a simple example of how the method works.

Pseudocode For Extracting Quasi-Chordal Subgraphs

Input: The original network G

Output: Reduced quasi-chordal network \tilde{G}

1. Partition the network across P processors
2. For subgraphs in each processor P_i
 - (a) Find maximal chordal subgraph in processor P_i
 - (b) Mark the border edges
 - (c) Send border edges to receiver processor, P_j
3. After receiving border vertices from sender processor, P_k
 - (a) For any two border edges (n_a, n_b) and (n_a, n_c) , if (n_b, n_c) is a chordal edge then, include (n_a, n_b) and (n_a, n_c) in the subgraph
4. End

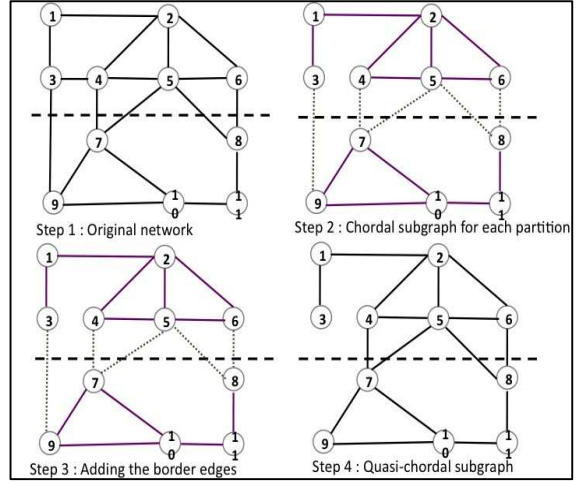


Figure 2. Left Figure: Pseudocode for extracting quasi-chordal subgraphs. Right Figure: Steps in applying the parallel graph sampling algorithm to a small graph. Step 1 shows the original network partitioned into two processors. Step 2 shows how the maximal chordal subgraph is obtained in each individual partition. The solid lines show the chordal edges and the dotted lines the border edges. In Step 3, the border edges that form a triangle with one chordal edge are selected. In this example, they are the pair (4,7) and (5, 7) and the pair (5,8) and (6,8). Step 4 shows the final quasi-chordal subgraph. There is now a cycle of length 5, (5,7,10,11,8), in the subgraph. (from [14]).

4.Results and Contributions:

We evaluated the effectiveness of our sampling algorithm by comparing both the combinatorial and functional properties of the original and subnetworks. The data sets for our experiments were obtained from NCBI's Gene Expression Omnibus (GEO) website (<http://www.ncbi.nlm.nih.gov/geo/>). Analysis of different test suites is provided in our earlier publications [14,15]. Due to the scarcity of space we will describe illustrate the important results based on one representative network (5,349 vertices and 7,277 edges) derived from hippocampal samples of young mice, aged 2 months (GSE5078 series by Verbitsky et al. [19]).

Analysis of combinatorial properties: We computed the clustering coefficient, degree distribution and the core numbers for the network and the subgraphs using MatlabBGL library [20]. The clustering coefficient of a vertex is computed as the ratio of the edges between the neighbors of a vertex to the total possible connections between the neighbors. A large clustering coefficient indicates presence of dense modules. The degree distribution shows how many vertices fall under the same range of degree, and in the case of gene correlation networks follows the power law. The core number of a vertex is defined as the largest integer c such that the vertex exists in a graph where all degrees are greater than equal to c . This metric relates to how closely high degree vertices are connected.

Table 1 compares the combinatorial properties including reduction in edges, clustering coefficients, number of vertices with high degrees (hubs) and high core numbers. The numbers in the parenthesis denote the reduction percentages. As expected, the subgraphs have lower number of edges,

the higher the number of processors, the more the reduction. Though there is significant reduction in the subgraphs, we still retain a high percentage of the hubs and vertices with high core numbers. The mean clustering coefficient remains the nearly equal indicating that the important clusters are preserved. The degree and clustering coefficient distribution is maintained in the subgraph samples (Figure 3).

Combinatorial Properties	Original Network	Quasi-Chordal Subgraph of Young Mice(GSE 5078) with				
		1 Partition	2 Partitions	4 Partitions	8 Partitions	16 Partitions
Number of Edges	7,277	4,949(31.9)	4,269(41.3)	4,029(44.6)	3,657(49.7)	3,753(48.4)
Mean Clust. Coeff.	.48	.39	.47	.40	.41	.41
High degree vertices	146	73(50)	106(72)	96(65)	86(58)	95(65)
Core Numbers	46	26(56)	25(54)	39(84)	36(78)	26(56)

Table 1. Comparison of combinatorial properties between the original networks and subgraphs (from [14]).

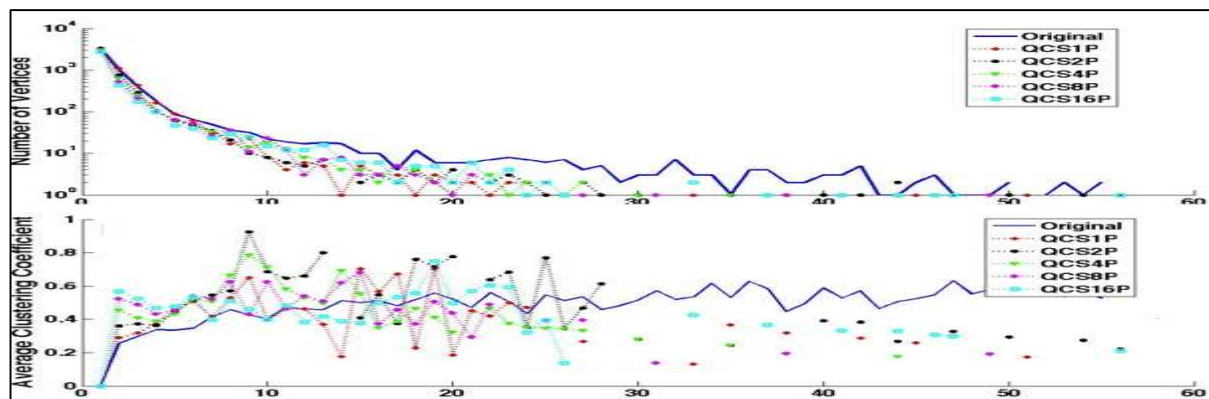


Figure 3. Degree distribution (top panel) and average clustering coefficient (bottom panel) of the original network (solid line) and the sampled networks (dotted lines). Y-axis gives the number of vertices (top figure) and average clustering coefficient per degree (bottom figure). X-axis gives the degrees in the network (from [14]).

Analysis of functional properties: We used the Cytoscape plug-in MCODE [21] on the network to identify clusters. Table 2 lists the most represented Gene Ontology (GO) molecular function terms per cluster as found in the original network and sampled networks on 1,2,4,8 and 16 processors respectively. Our results show that most of the important functional units are preserved in the samples. In addition, due to the elimination of noise, sampling sometimes reveals new clusters in the reduced networks, like enzyme regulator activity in QCS 4 and 8 and transmembrane transport activity in QCS 8 and 16.

Analysis of Parallel Implementation: The scalability of our parallel algorithm is computed as follows; Let the total number of edges be E , the maximum degree be d and the number of processors involved be p . The complexity of the sequential algorithm is given by $T_{seq}(E) = O(Ed)$ [18]. The parallel overhead $T_{over}(E,p)$ consists of communicating the border nodes from each processor, denoted by b , and checking if they form a triangle with a chordal edge. Assuming equal distribution of border nodes, the total communication volume is $O(bp)$ and the computation volume over all processors is $O(b^2p)$. Therefore, $T_{over}(E,p) = O(b^2p)$. In order maintain isoefficiency, $T_{seq}(E) \geq T_{over}(E,p)$, therefore, $E \geq Cb^2p/d$, where C is a constant. The memory required to store the network is approximately $O(E)$. Therefore, the scalability function is $Cb^2pd/p = O(b^2d)$, thus the overhead increases with the number of border edges.

This analysis is also borne out by empirical results. Figure 4 shows the breakdown of the execution times of the different sections of the algorithm over 1,2,4,8 and 16 processors. As can be seen from the figure the time to compute the local QCS (blue blocks) gradually decreases over the number of processors. However, the communication costs (border edges) keep increasing with the number of processors, which finally leads to significant increase in the execution time.

Cluster Id	Original Network	QCS on 1 Partition	QCS on 2 Partitions	QCS on 4 Partitions	QCS on 8 Partitions	QCS on 16 Partitions
1	protein binding	protein binding	protein binding	protein binding	protein binding	protein binding
	catalytic activity	catalytic activity	catalytic activity			catalytic activity
	binding	binding	binding	binding	binding	binding
				enzyme regulator activity	enzyme regulator activity	
2	receptor binding	receptor binding	receptor binding	receptor binding		
	protein binding	protein binding	protein binding	protein binding	protein binding	protein binding
	binding	binding	binding	binding	binding	binding
					transmembrane transport. activity	transmembrane transport. activity

Table 2. Comparison of functional units of the young mice network. The leftmost column of functionalities is from the original network, proceeding to the functionalities obtained for QCS with 1,2,4,8 and 16 partitions respectively. Same color within each cluster represents similar functionality (from [14]).

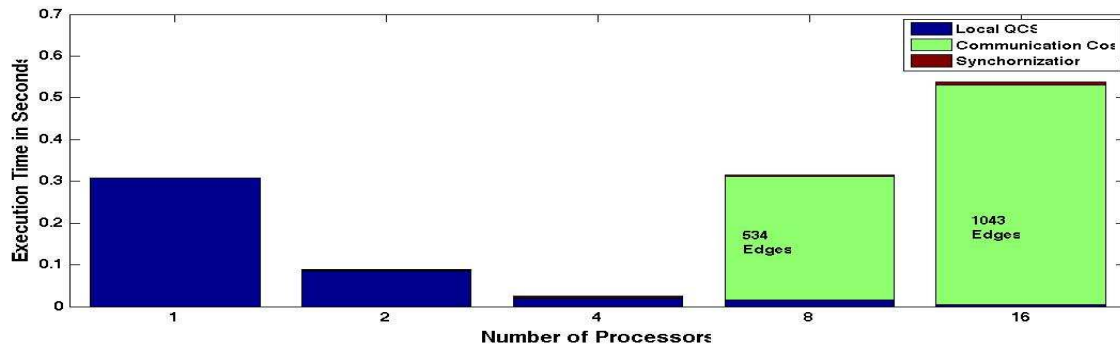


Figure 4. Breakdown of execution time for obtaining QCS over different number of processors. As the number of processors increase, the communication overhead for the border edges outweighs the gains due to parallelization.

5. Discussion

Our results show that QCS based sampling can be very effective in reducing the memory requirements, while preserving important combinatorial and functional properties. We observe that the reduction of noise can unveil new functional units within the networks. However, we did not find any simple relation between graph properties and retention of functional units. We would like to note, therefore, that only comparing combinatorial properties may not be sufficient to validate sampling of real-world networks.

Our parallel algorithm loses efficiency as more processors are added. We believe that this is not necessarily due to bad partition, but because border edges increase with more processors. We are currently investigating a master-worker approach that might reduce some of the communication costs. Other future directions include sampling on weighted networks and on dynamic evolving networks.

Acknowledgments : I would like to thank the Dr.Sanjukta Bhowmick and Dr. Hesham Ali of the University of Nebraska at Omaha(UNO), for their help and Kate Dempsey (UNO) for her help in obtaining the biological functional units. The research was funded by the Nebraska EPSCoR First Award and the College of IS&T, University of Nebraska at Omaha.

References:

- [1] J. C. Miller and J. M. Hyman. Effective vaccination strategies for realistic social networks. *Physica A*.386,780-785(2007).

- [2] K. Voevodski, S. H. Teng, and Y. Xia. Finding local communities in protein networks *BMC Bioinformatics*,10, 297(2009).
- [3] R. Yokomori, H. Siy, M. Noro, and K. Inoue. *Assessing the Impact of Framework Changes Using Component Ranking. International Conference on Software Maintenance. ICSM '09.*2009.
- [4] D. A. Bader, G. Cong. A Fast, Parallel Spanning Tree Algorithm for Symmetric Multiprocessors.*18th International Parallel and Distributed Processing Symposium. IPDPS'04.*
- [5] G. Cong, G. Almasi, V. Saraswat. A Fast PGAS Implementation of Distributed Graph Algorithms. *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC'10.* 2010.
- [6] V. Agarwal, F. Petrini, D. Pasetto and D. A. Bader. Scalable Graph Exploration on Multicore Processors. *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis. SC'10.* 2010.
- [7] J. Leskovec and C. Faloutsos. Sampling from large graphs. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD'06.* 2006.
- [8] V. Krishnamurthy, M. Faloutsos, J.-H. Chrobak, M. Cui, L. Lao, and A. G. Percus. Sampling large internet topologies for simulation purposes. *Computer Networks* 51, 2007, 4284–4302.
- [9] J. Leskovec, J. Kleinberg, C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, KDD'05.* 2005.
- [10] J. L. Gross, J. Yellen. *Handbook of Graph Theory and Applications*, CRC Press, 2004.
- [11] B. Marshall, J. R. Gilbert, B. Hendrickson, N. Nguyen, S. Toledo. Support-graph preconditioners. *SIAM Journal on Matrix Analysis and Applications* 27, 4, 2006, 930–951.
- [12] D. Rafiei, S. Curial, Effectively visualizing large networks through sampling, *Visualization Conference, IEEE* 2005.
- [13] A. Rasti, M. Torkjazi, R. Rejaie, N. Duffield, W. Willinger, D. Stutzbach, Respondent-driven sampling for characterizing unstructured overlays, *INFOCOM* 2009. pp. 2701–2705.
- [14] K. Dempsey, K. Duraisamy, H. Ali, S. Bhowmick, A parallel graph sampling algorithm for analyzing gene correlation networks. *International Conference in Computational Science 2011. ICCS'11.* 2011.
- [15] K. Duraisamy, K. Dempsey, H. Ali, S. Bhowmick, A noise reducing sampling approach for uncovering critical properties in large scale biological networks. *The 2011 International Conference on High Performance Computing & Simulation. HPCS'11.* 2011 (submitted).
- [16] A. Miranda, L. Garcia, A. Carvalho, and A. Lorena, Use of classification algorithms in noise detection and elimination. Hybrid Artificial Intelligence Systems, *Lecture Notes in Computer Science*, 5572. 2009, pp. 417–424.
- [17] G. L. Libralon, A. Carvalho, and A. C. Lorena, Preprocessing for noise detection in gene expression classification data. *Journal of the Brazilian Computer Society* 15.2009, 3 –11.
- [18] P. M. Dearing, D. R. Shier and D. D. Warner, Maximal Chordal Subgraphs, *Discrete Applied Mathematics*. 20, 3,1988. 181-190.
- [19] M. Verbitsky, A. L. Yonan, G. Malleret, E. R. Kandel, T. C. Gilliam, P. Pavlidis, Altered hippocampal transcript profile accompanies an age-related spatial memory deficit in mice. *Learning and Memory* (Cold Spring Harbor, N.Y.) 11,3, 2004, 253–260.
- [20] MatlabBGL Library(http://www.stanford.edu/dgleich/programs/matlab_bgl/).
- [21] G. D. Bader, C.W. Hogue, An automated method for finding molecular complexes in large protein interaction networks, *BMC Bioinformatics* 4. 2. 2003.