Efficient Community Detection Using Power Graph Analysis

George Tsatsaronis
Biotechnology Center
Technische Universität
Dresden, Germany
george.tsatsaronis@biotec.tu-dresden.de

Matthias Reimann
Biotechnology Center
Technische Universität
Dresden, Germany
reimann@biotec.tu-dresden.de

Iraklis Varlamis
Dept. of Informatics and Telematics
Harokopio University
Athens, Greece
varlamis@hua.gr

Orestis Gkorgkas
Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
orestis@idi.ntnu.no

Kjetil Nørvåg
Dept. of Computer and Information Science
Norwegian University of Science and Technology
Trondheim, Norway
noervaag@idi.ntnu.no

ABSTRACT

Understanding the structure of complex networks and uncovering the properties of their constituents has been for many decades at the center of study of several fundamental sciences, such as discrete mathematics and graph theory. Especially during the previous decade, we have witnessed an explosion in complex network data, with two cornerstone paradigms being the biological networks and the social networks. The large scale, but also the complexity, of these types of networks constitutes the need for efficient graph mining algorithms. In both examples, one of the most important tasks is to identify closely connected network components comprising nodes that share similar properties. In the case of biological networks, this could mean the identification of proteins that bind together to carry their biological function, while in the social networks, this can be seen as the identification of communities. Motivated by this analogy, we apply the *Power Graph Analysis* methodology, for the first time to the best of our knowledge, to the field of community mining. The model was introduced in bioinformatics research and in this work is applied to the problem of community detection in complex networks. The advances in the field of community mining allow us to experiment with widely accepted benchmark data sets, and our results show that the suggested methodology performs favorably against state of the art methods for the same task, especially in networks with large numbers of nodes.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph Algorithms*; J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Sociology*

General Terms

Theory, Algorithms, Experimentation.

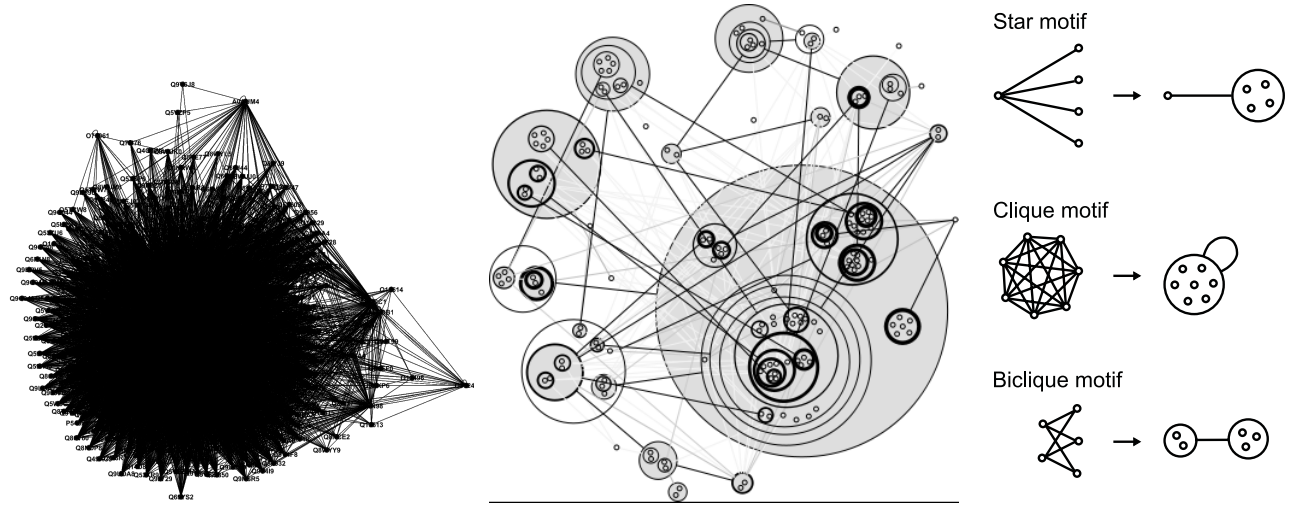
Keywords

Power Graph Analysis, Community Mining, Graph Mining

1. INTRODUCTION

Complex networks are a prosperous field for data mining research. The representation and visualization of complex networks as graphs and the application of data mining techniques can help us uncover interesting knowledge regarding how the constituents of the graphs interact, and, more precisely, which are the components that have a high degree of internal links, yet lower density of links connecting them to the rest of the components. This knowledge is crucial in research areas such as bioinformatics, where for example protein-to-protein interactions are modelled in a huge graph, and the aim is to identify the proteins that bind together to perform a certain biological function.

In analogy, in social networks the respective knowledge is crucial to identify communities, i.e., clusters of nodes that have high density of internal links between their constituents but connect more loosely to the rest of the clusters. In this direction, there have been many advances to the field of community mining and several novel algorithms have been suggested that attempt to identify efficiently the respective communities [2, 5]. Motivated by this analogy, and also by the fact that very recently we applied successfully *Power Graph Analysis* to predict the authors' evolution over time in bibliographical databases [9], in this paper we present for the first time, to the best of our knowledge, the application of a previously successful methodology in bioinformatics, namely



(a) Huge biological "fur ball" network. (b) After the application of *Power Graphs*. (c) Basic motifs recognized by *Power Graph Analysis*.

Figure 1: Figure1(a) shows an example of a huge biological network. Figure 1(b) shows the corresponding *Power Graph*. The three basic motifs recognized by *Power Graphs* are shown in Figure 1(c): *Star*, *Clique* and *Biclique*. *Power Nodes* are sets of nodes and *Power Edges* connect *Power Nodes*. A *Power Edge* between two *Power Nodes* signifies that all nodes of the first set are connected to all nodes of the second set.

Power Graph Analysis [8], to the field of community detection [5]. The benefits of applying *Power Graph Analysis* for community detection are twofold: (1) the methodology allows for fast and large-scale node clustering experiments, since it can compress by even up to 90% the information of the original network in a lossless information manner [8], and, (2) efficient visualization of the original network becomes feasible, through identifying several motifs, such as *cliques* and *bi-cliques*.

In all, the contributions of this work can be summarized into the following: (a) application of the bioinformatics-based *Power Graph Analysis* methodology to the field of community detection, and, (b) thorough experimental evaluation that demonstrates the feasibility and the efficiency of our approach in benchmark datasets. The rest of the paper is organized as follows: Section 2 presents some preliminary concepts and discusses related work. Section 3 discusses how *Power Graph Analysis* can be applied to community detection. Section 4 presents our experimental evaluation, and, finally, Section 5 concludes and provides pointers to future work.

2. PRELIMINARIES AND RELATED WORK

2.1 Power Graph Analysis

In the bioinformatics field, networks play a crucial role, but their efficient visualization is difficult. Biological networks usually result in *"fur balls"*, from which little insight can be gathered. In the direction of providing an efficient methodology for visualizing large and complex networks, such as protein interaction networks, the authors in [8] introduce *Power Graph Analysis*, a methodology for analyzing and representing efficiently complex networks, without losing information from the original networks. The analysis is based on identifying *re-occurring network motifs* using several abstractions. The three basic motifs recognized by *Power Graphs* are shown in Figure 1(c). These are the *Star*, the *Clique* and the *Biclique*, and constitute the basic abstractions when transforming the original graph into a *Power Graph* with *Power Nodes*, i.e., sets of nodes, connected by *Power Edges*. *Power Graphs* offer up to 90% compression of the original network structure [8], allowing for efficient visualization. Figure 1 shows an example of a *"fur ball"* network, and its transformation after the application of *Power Graph Analysis*. Observing Figure 1(b) now constitutes feasible the task of identifying the main protein interactions, compared to examining the original biological network in Figure 1(a). Motivated by the efficiency of *Power Graph Analysis* to uncover the complex structure of such networks, in this work we transfer the same methodology to identify communities.

When transferring *Power Graph Analysis* from biology to community detection, one would expect a discussion whether the basic constructs can be seamlessly and completely transferred to community network data. Do the main motifs also occur in community data, or, vice versa, are there other motifs that could be identified but that are not considered in *Power Graphs* in biology? The *Power Graph* transformation is based on the network motifs *Biclique* and *Clique*. *Power Graphs* have been successfully applied in the biological domain as its networks are rich in such motifs. Community networks are implicitly built on such motifs, as a community comprises of densely linked nodes, identified by the motifs of *Biclique* and *Clique*. Hence, *Power Graph Analysis* is perfectly suited for community detection.

2.2 Community Detection Algorithms

As the aim of the community detection algorithms is to identify clusters of nodes, within which the intra-node connections are dense, and outside which the inter-node connection is less dense, a large focus has been given to the use of *betweenness centrality* measures. In this direction, the

most influential algorithm has been the *Girvan-Newman* algorithm [4], which extends the notion of *vertex betweenness* to the notion of *edge betweenness*. The algorithm uses this notion in order to identify the least central edges, curing this way the pathologies of methods that attempt to identify the most central edges or vertices in a graph. One such pathology is for example the fact that nodes which are connected to the network with only one edge, cannot be classified to a community.

Though influential, the *Girvan-Newman* algorithm has been shown to be less effective for the detection of communities, compared to recent approaches that utilize *greedy strategies* for community detection [5]. Examples of such recent approaches are the *Label propagation* algorithm [6], *heuristic methods for modularity optimization* [1], *multiresolution community detection* algorithms [7], and *greedy size-constrained community detection* approaches [2].

In [6] the authors present a *greedy label propagation* algorithm to detect communities. The algorithm resembles the way $k-$nearest neighbors operate in the data classification paradigm. Initially all the nodes of a network are assigned a unique label, i.e., a unique community identifier, and the algorithm propagates these labels iteratively applying a very simple methodology; each node in each iteration will be assigned the label that most of its neighbors belong to. The process continues until there are no further changes in the label assignment, and the final communities are defined by the nodes' labels, i.e., nodes with the same label belong to the same community. However, a major drawback of this approach is that it does not have a unique solution, as convergence of such a greedy approach is hard to prove. Another problem is that often the solution is reduced to one single community, i.e., all the nodes are assigned with the same label after many iterations. In [1] the authors use again the notion of label propagation, but the approach differs in the greedy step. The nodes are assigned labels based on the *gain of modularity* this assignment would have. In addition, they introduce a second phase, that is executed after each label propagation phase, which consists of contracting partitions into a new network. The whole process finishes when there is no further gain in the modularity. In [7] the authors use again the notion of label propagation, with the difference being again on the assignment criterion of the labels. They introduce the *Absolute Potts Model* which is used as their membership decision function. The method can be used to compute partitions of nodes in different resolutions, and significant structures, i.e., communities, can be identified by measuring strong correlations between the multiple partitions. Finally, in [2], the authors introduce an approach for identifying size-constrained communities. It belongs to the category of greedy approaches that attempt to maximize modularity, where their decision function is based on the notion of *affinity* that measures the strength with which a node is connected to a cluster of nodes, i.e., a community.

2.3 Benchmark Datasets and Evaluation

The creation of benchmark datasets for evaluation has been a long-standing problem in the area of community detection [4, 5]. The problem stems from the fact that there is no common consensus on how exactly a network with communities is defined. However, the past few years there seems to be an acceptance of the *planted l-partition model* [3]. According to that model, partitions that consist of a certain number of nodes are *planted* to a network. For each node, there is a probability p_{in} signifying the chances of the node to get connected with nodes of its group. Respectively, there is a probability p_{out} denoting the chances that the node is connected to nodes of different groups. The assumption is that as long as $p_{in} > p_{out}$ the *planted* groups represent communities, while if $p_{in} \leq p_{out}$ the produced network is simply a random graph. In this work we follow this model for creating *LFR* benchmark synthetic datasets [5] for evaluating our community detection approach[1]. The respective software can be parameterized to produce synthetic graphs of different sizes, different number and sizes of communities, as well as different mixture probability models. In all cases, the software produces the *ground truth*, i.e., which are the communities that should be identified by the tested approaches.

The problem of evaluating different community detection approaches is now reduced to comparing how good the provided partitions by the tested methods are against the *ground truth*. Motivated also by the data mining area (clustering), as well as from the information theory discipline, the respective research community has adopted widely for this purpose the use of the *Normalized Mutual Information Measure* (*NMI*). *NMI* operates directly on the *confusion matrix* created by setting as rows the original communities, i.e., the *ground truth*, and as columns the communities identified by an approach. Let C be the confusion matric, and N_{ij} the element at row i and column j. N_{ij} denotes the number of nodes in the intersection of the original community i and the generated community j. If C_A denotes the number of the communities in the *ground truth*, C_B the number of the generated communities by an approach, N_i the sum of row i, N_j the sum of column j, and N the sum of all elements in C, then the *NMI* score between the *ground truth* partition A, and the generated partition B can be computed as shown in the following equation.

$$NMI(A,B) = \frac{-2\sum_{i=1}^{C_A}\sum_{j=1}^{C_B} N_{ij} log(\frac{N_{ij}N}{N_i N_j})}{\sum_{i=1}^{C_A} N_i log(\frac{N_i}{N}) + \sum_{j=1}^{C_B} N_j log(\frac{N_j}{N})} \quad (1)$$

NMI can also be modified to handle the evaluation of partitions where overlapping communities exist [2, 5], but in this work we only experiment with networks where each node belongs to exactly one community.

3. POWER GRAPH ANALYSIS FOR COMMUNITY DETECTION

In this section we present the details of applying *Power Graph Analysis* for community detection. Primarily, we make the assumption that the input networks are *undirected* and *unweighted* graphs.[2] We proceed by formally describing the problem. Let $G = \{V, E, f\}$ be an input graph, where V is the set of vertices, E is the set of edges, and $f : V \times V \rightarrow E$. The *Power Graph Analysis* transforms this graph into a new graph $PG = \{PV, V', PE, E', g, f'\}$, where PV are the *Power Nodes* of the graph, V' are simple nodes, i.e., $V' \subset V$, PE are the *Power Edges*, E' are simple edges, i.e., $E' \subset E$, $g : PV \times \{PV \cup V'\} \rightarrow PE$, $f' : V' \times V' \rightarrow E'$.

[1]Software is publicly available at: `http://santo.fortunato.googlepages.com/inthepress2`

[2]In future work we plan to address directed and/or weighted graphs.

Input: A *Power Graph* $PG = \{PV, V', PE, E', g, f'\}$
Output: Assignment of all nodes to communities $C = \{C_j\}$

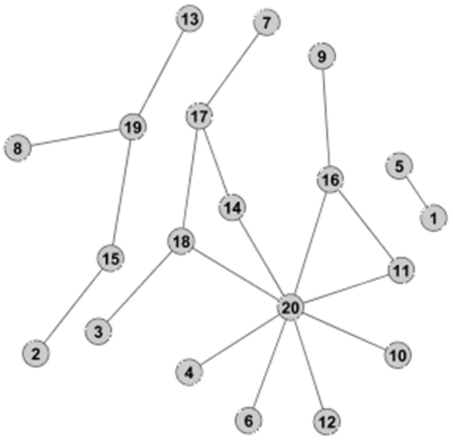
foreach $v_i \in PV_j$ **do**
  $C_j = C_j \cup v_i$
foreach $PV_i \in PV_j$ **do**
  $C_j = C_j \cup C_i$
  $C_i = \emptyset$
foreach $v_i \in V'$: *v_i is connected to a set of* Power Nodes $\{PV_k\}$ *with* $\{PV_k\} \neq \emptyset$ **do**
  $P' = \Cup_{k=1..m} PV_k$
  $j = argmax_j |PV_j|, PV_j \in P'$
  $C_j = C_j \cup v_i$
foreach $v_i \in V'$: *v_i is connected to a set of nodes* $\{v_k\}$ *and* $v_k \in C_k$ **do**
  $C' = \Cup_{k=1..m} C_k$
  $j = argmax_j |C_j|, C_j \in C'$
  $C_j = C_j \cup v_i$
foreach *Node* $v_i \in V'$ *that is not member of a community* **do**
  foreach *Edge* $e \in E'$ *between* $v_i \in V'$ *and* $v_k \in V'$ **do**
    if $v_k \in C_k$ **then**
      $C' = \Cup_{k=1..m} C_k$
      $j = argmax_j |C_k|, C_j \in C'$
      $C_j = C_j \cup v_i$
    else
      $C_{ij} = \emptyset \cup \{v_i\} \cup \{v_j\}$

Algorithm 1: Community detection given the *Power Graph Analysis* output.

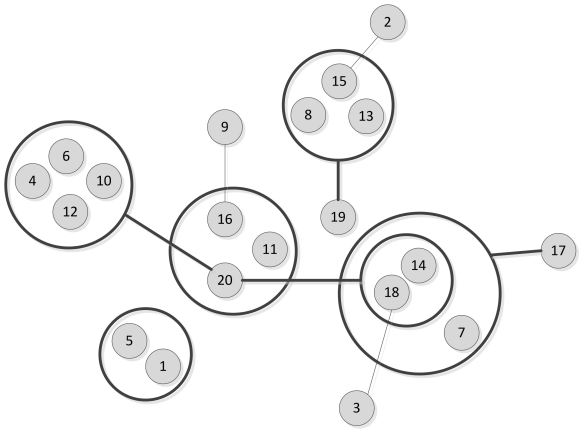
The aforementioned description implies that the resulting *Power Graph* may contain *Power Nodes*, as well as simple nodes. A *Power Node* comprises several nodes from the original graph. An edge between *Power Nodes*, or between a *Power Node* and a simple node, is a *Power Edge*, which means that all the components of one end of the edge are connected to all components of the other end of the edge. However, the resulting *Power Graph* may also contain simple edges, i.e., edges between simple nodes. Thus, not all nodes in the resulting *Power Graph* are necessarily part of a *Power Node*, and not every node is necessarily connected with an edge to a *Power Node*.

Given the output of this transformation during which the motifs described in Figure 1(c) are recognized in the original graph G, in Algorithm 1 we describe how the nodes of PG are assigned to communities.[3] According to Algorithm 1, the output of the *Power Graph Analysis* execution is processed as follows: First, the nodes that belong to the same *Power Node* are assigned to the same community. Second, in the case of *Power Nodes* contained in other *Power Nodes*, all the nodes belonging to the wider *Power Node* are assigned to the same community, i.e., the sub-communities shaped by smaller *Power Nodes* inside the larger *Power Node* are merged. Finally, all the remaining nodes, which have not been assigned into a *Power Node*, i.e. into a community, are

[3]The software to compute the *Power Graph* of any input graph is publicly available from: `http://www.biotec.tu-dresden.de/research/schroeder/powergraphs/`



(a) The original graph.



(b) The resulting *Power Graph*.

Figure 2: An example of detecting communities using *Power Graph Analysis*.

explicitly assigned to a community based on the following cases, which are examined in that particular order: (1) a node $v \in V'$ has a *Power Edge* $pe \in PE$ to a *Power Node* PV. In this case, v will be assigned to the same community with the nodes inside PV. This is in essence the *star* motif. In case v has many *Power Edges* towards different *Power Nodes*, then v is assigned to the community of the nodes belonging to the largest of these *Power Nodes*. Ties are broken using uniform distribution; (2) a node $v \in V'$ has an edge $e \in E'$ towards another node v' that is part of a *Power Node*. In this case, v will be assigned to the same community as v'. In case v has more than one such edges, then it is assigned to the largest of the communities; (3) a node $v \in V'$ has an edge $e \in E'$ towards another node v' that is not part of a *Power Node*. In this case, v and v' are assigned to the same community. In case v' is already member of a community, due to cases (1) or (2), then v joins the same community. In case v has many such edges, then it joins the largest of the communities. Again, ties are broken uniformly. The complexity of Algorithm 1 is $O(PV+V'+V'E')$, and since $V'E'$ is typically larger than PV and V', the complexity is, thus, $O(V'E')$. However, since the vast majority of the nodes in a *Power Graph* are typically members of a *Power Node*, it holds that $V' << V$, and thus $V'E' << VE$, making the

Dataset	Mixing Par.	$\mu_t = 0.1$	$\mu_t = 0.2$	$\mu_t = 0.3$	$\mu_t = 0.4$	$\mu_t = 0.5$	$\mu_t = 0.6$	$\mu_t = 0.7$	$\mu_t = 0.8$	$\mu_t = 0.9$
$1k, S$	#Edges	9,595	9,700	9,869	9,777	9,837	9,755	9,803	9,837	9,758
$1k, S$	#Power Edges	3,686	4,812	5,780	6,522	7,222	7,781	8,204	8,400	8,350
$1k, S$	#Edge Reduction Rate	0.6157	0.503	0.4143	0.3329	0.2657	0.2022	0.163	0.146	0.1443
$1k, L$	#Edges	9,737	9,774	9,667	9,750	9,779	9,890	9,806	9,780	9,764
$1k, L$	#Power Edges	5,788	6,209	6,788	7,371	7,783	8,240	8,342	8,345	8,347
$1k, L$	#Edge Reduction Rate	0.4053	0.3646	0.2978	0.2439	0.204	0.1668	0.1493	0.1466	0.145
$5k, S$	#Edges	49,082	48,620	48,621	49,081	48,834	48,807	48,704	49,183	48,806
$5k, S$	#Power Edges	18,367	24,167	28,566	33,055	36,256	39,633	41,926	43,712	43,562
$5k, S$	#Edge Reduction Rate	0.6256	0.5028	0.4124	0.3248	0.2514	0.1879	0.1391	0.112	0.1074
$5k, L$	#Edges	48,898	48,829	48,996	48,959	49,056	48,841	49,200	48,868	48,713
$5k, L$	#Power Edges	28,983	31,939	35,020	37,761	39,631	41,545	43,346	43,558	43,506
$5k, L$	#Edge Reduction Rate	0.4072	0.3458	0.2852	0.2286	0.1921	0.1493	0.1189	0.1086	0.1068
$100k, L$	#Edges	977,592	977,277	977,112	977,557	977,776	978,051	975,703	978,124	978,975
$100k, L$	#Power Edges	579,012	639,430	695,539	748,338	795,649	839,030	871,719	889,352	888,551
$100k, L$	#Edge Reduction Rate	0.4076	0.3456	0.2881	0.2344	0.1862	0.1421	0.1065	0.093	0.092

Table 1: Number of edges in the original *LFR* graphs and in the constructed *Power Graphs*.

algorithm applicable even for huge *Power Graphs*.

For the execution of Algorithm 1 however, one must also add the computational cost of the *Power Graph* creation, given the original graph. Though the created *Power Graph* is considered as input to Algorithm 1, we explain in the following the theoretical complexity of creating it: the process comprises two-phases. In the first phase the algorithm identifies potential *Power Nodes* using a *Jaccard*-based similarity metric on the neighbors of each node and a similarity based hierarchical clustering algorithm. The second phase of the *Power Graph* algorithm performs a greedy search for *Power Edges*, by examining the problem of minimizing the *Power Graph* structure as an optimization problem. Thus, its complexity is relative to the complexity of the hierarchical algorithm, which has the higher cost ($O(n^2 log(n))$), if the priority-queue *HAC* algorithm is implemented, and to the complexity of the greedy power edge search algorithm, which is linear to the number of *Power Nodes* ($O(pn)$).

In Figure 2 we present an example of the application of Algorithm 1 into the original graph shown in Figure 2(a). The resulting *Power Graph* is shown in Figure 2(b). The original graph was created using the *LFR* benchmark with 20 nodes, an average degree of 4, a mixing parameter of 0.3, and minimum and maximum community sizes set to 5 and 10 respectively. The *ground truth* is that there exist three communities, namely: $C_1 = \{1, 3, 5, 6, 7, 17, 18\}$, $C_2 = \{2, 8, 13, 15, 19\}$, and $C_3 = \{4, 9, 10, 11, 12, 14, 16, 20\}$. Our algorithm finds four communities, namely: $C'_1 = \{1, 5\}$, $C'_2 = \{4, 6, 9, 10, 11, 12, 16, 20\}$, $C'_3 = \{2, 8, 13, 15, 19\}$, and $C'_4 = \{3, 7, 14, 17, 18\}$, producing an *NMI* score of 0.6789.

4. EXPERIMENTAL EVALUATION

In the following, we present the results of our experimental evaluation by detecting communities using synthetic *LFR* benchmark datasets.

4.1 Experimental Setup

We follow the same experimental setup as in [5], in order to produce *LFR* benchmark graphs. More precisely, in order for our results to be comparable with the reported results in the bibliography, we created undirected and unweighted graphs according to the following setup[4]: (i) 900 graphs with $1,000$ nodes each, community sizes between 10 and 50 nodes, and a mixing parameter μ_t ranging from 0.1 to 0.9 (100 graphs for each different μ_t). We will refer to this dataset as $1k, S$, due to the small size of communities; (ii) 900 graphs with the same set up as before, but with community sizes between 20 and 100. We will refer to this dataset as $1k, L$; (iii) and (iv) with the same setup as (i) and (ii) respectively, but with the number of nodes in the graph being $5,000$ nodes. We will refer to these two datasets as $5k, S$ and $5k, L$ respectively. In total, for the experiments of the datasets (i)-(iv) we processed $3,600$ graphs. In addition, to demonstrate the scalability of our approach, we created graphs following setup (iv), but changing the number of nodes to $100,000$. We refer to this dataset as $100k, L$.

4.2 Results

Table 1 shows the efficiency of the suggested approach in compressing the original LFR graphs, without loosing information. The table shows for all the aforementioned datasets (horizontally), and all the different mixing parameters (μ_t) used, the average number of edges in the original graphs (*#Edges*), the average number of *Power Edges* in the constructed *Power Graphs* (*#Power Edges*), and the average edge reduction rate occurred from this transformation (*#Edge Reduction Rate*). Two important conclusions may be drawn: (1) the approach can achieve a compression rate of up to 61.67%, which means that the resulting *Power Graph* holds a lot less than the half of the original edges, enabling the application of our methodology to very large graphs, and (2) as the number of nodes increases from $1,000$ to $100,000$ in the original *LFR* graphs, the edge reduction rate is not affected much, reaching even up to 40.76% for the $100k, L$ dataset. We can also observe how difficult it becomes for the *Power Graph Analysis* to detect cliques and bi-cliques as the mixing parameter increases from 10% to 90%. This also shows that in cases where the mixing parameter increases, we expect a dramatic drop in performance with regards to the successfully detected communities.

[4]If not stated otherwise, in all produced graphs the average degree is 20 and the maximum degree is 50.

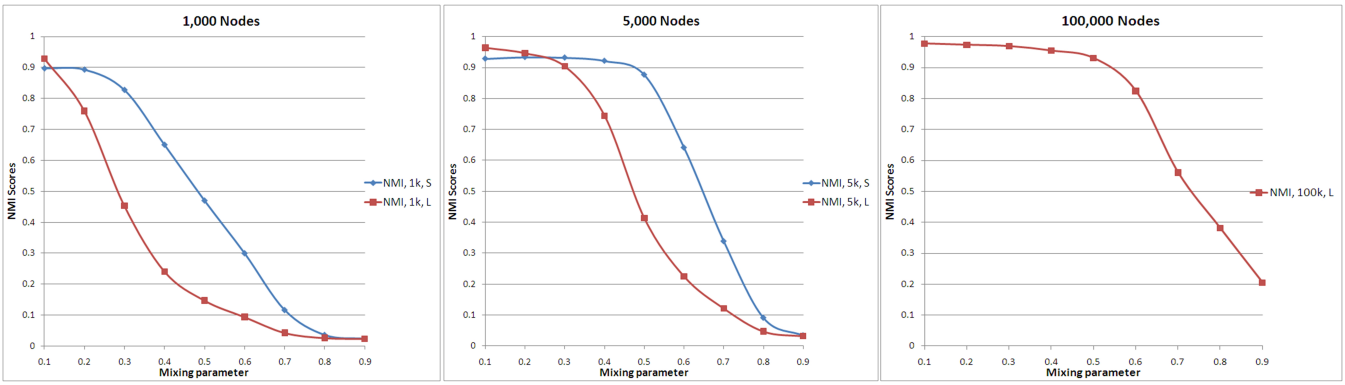


Figure 3: Results on the *LFR* benchmark datasets. 1,000 nodes, 5,000 nodes and 100,000 nodes respectively.

In Figure 3 we present the results of our method in detecting communities, using the previously described *LFR* benchmark datasets. The three graphs show respectively the results when using initial graphs of $1,000$ nodes, $5,000$ nodes, and $100,000$ nodes. In the first two graphs we experimented both in small ($1K, S$, and $5K, S$) and large ($1K, L$, and $5K, L$) planted communities. In the large graphs of $100,000$ nodes we used only the large community setup, i.e., all communities are between 20 and 100 nodes in size. Comparing the results of the first two plots, with the respective reported results of 12 different community detection methods presented in [5], we may see that our method is always amongst the top-5 performing methods. However, comparing our results in the $100,000$ nodes graphs with the rest of methods, we may see that our method is the best performing one among the reported ones in the respective setup [5]. The most interesting findings of our *Power Graph Analysis* approach to detect communities is the fact that if we examine closely the three plots of Figure 3 we may observe that: (1) our approach performs better in identifying smaller communities (blue lines) than large ones (red lines), and, (2) as the number of nodes in the *LFR* graphs increases, our approach constantly becomes better, with its performance in the case of $100,000$ *LFR* graphs being the best reported among all related methods [5]. The respective methods report an *NMI* score between 0.2 and 0.3 for mixing parameters between 0.6 and 0.7, while our method for the same mixing parameters reports an almost double *NMI* score.

5. CONCLUSIONS AND FUTURE WORK

In this paper we introduced a novel methodology for community detection, transferring the paradigm of *Power Graph Analysis* from the bioinformatics domain to the domain of community mining. The advantages of the suggested approach are twofold: (a) the methodology allows for efficient large-scale community detection experiments, as it may compress the original *LFR* benchmark graphs up to, approximately, 60%, and, (b) efficient visualization of the original network and the communities becomes feasible, through the visualization of the *Power Graph*. Our experimental evaluation in more than $4,000$ *LFR* graphs ranging from $1,000$ to $100,000$ nodes showed that the suggested approach has a reported *NMI* score among the top-5 best approaches in the field, and the top performance for the tested networks of the larger size. Our approach has definitely some limitations; primarily it requires the execution of the *Power Graph Analysis* to the input graphs. This is not a trivial computational cost, especially for large graphs, but, as the experiments showed, it is certainly feasible to apply it in graphs that are in the order of magnitude of hundreds of thousands nodes. As a future work, we plan to investigate the role of embedded *Power Nodes* as sub-communities, and we also aim at investigating other types of graphs as well, such as directed and/or weighted. It is also in our next plans to experiment with large real world data sets.

6. REFERENCES

[1] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of community hierarchies in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 10, 2008.

[2] M. Ciglan and K. Nørvåg. Fast detection of size-constrained communities in large networks. In *WISE*, pages 91–104, 2010.

[3] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, 2001.

[4] M. Girvan and M. Newman. Community structure in cosial and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.

[5] A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Phys Rev E Stat Nonlin Soft Matter Phys*, 80(5 pt 2):056117, 2009.

[6] U. Raghavan, R. Albert, and S. Kumara. Near liner time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76(3):036106, 2007.

[7] P. Ronhovde and Z. Nussimov. Multiresolution community detection for megascale netwroks by information-based replica correlations. *Phys. Rev. E*, 80(1):016109, 2009.

[8] L. Royer, M. Reimann, B. Andreopoulos, and M. Schroeder. Unraveling protein networks with power graph analysis. *PLoS Computational Biology*, 4(7):e1000108, 2008.

[9] G. Tsatsaronis, I. Varlamis, S. Torge, M. Reimann, K. Nørvåg, M. Schroeder, and M. Zschunke. How to become a group leader? or modelling author types based on graph mining. In *Proc. of TPDL*, 2011.