

Detecting Hubs and Quasi Cliques in Scale-free Networks

Sriganesh Srihari
School of Computing
National University of Singapore
srigsri@comp.nus.edu.sg

Hoong Kee Ng
School of Computing,
National University of Singapore
nghoongk@comp.nus.edu.sg

Kang Ning
Department of Pathology,
University of Michigan
kning@umich.edu

Hon Wai Leong
School of Computing,
National University of Singapore
leonghw@comp.nus.edu.sg

Abstract

Scale-free networks are believed to closely model most real-world networks. An interesting property of such networks is the existence of so-called hub and community structures. In this paper, we model hubs as high-degree nodes and communities as quasi cliques. We propose a new problem formulation called the λ -LIST DOMINATING SET and show how this single problem is suited to model both the structures in real-world networks better than traditional problems like VERTEX COVER and CLIQUE. Additionally, we provide a fixed-parameter tractable algorithm to this detect these structures and show experimental results on Protein-Protein Interaction networks.

1. Introduction

Many studies have been carried out to understand and model the behavior of real-world networks. In 1999, Barabási et al. studied the Web and to their surprise found that the Web does not follow the classical Erdős-Rényi random graph model, but rather has a power law distribution of connectivity. This results in the formation of (relatively) small number of ‘hubs’ or centers having large interactions and vast majority of nodes having small interactions. Subsequently, Barabási et al. proposed a generative mechanism called the ‘preferential attachment’ model leading to the so-called ‘scale-free’ class of networks. Hubs in a scale-free network ensure that the network does not lose its connectivity, but when removed, can break the network. Most nodes also have a tendency to take part in ‘com-

munities’. Every node within a community interacts with ‘most’ of other nodes in its community and with some outside.

In this paper, we study hubs and communities in scale-free networks. In order to emphasize on the motivational aspect of this study, we take the example of Protein-Protein Interaction networks. While the research community is still debating on the scale-free nature of biological networks, PPI networks tend to show this behavior [1]. PPI networks represent interaction between proteins. These interactions are large and complex and believed to direct most biological processes and pathways. Hubs control most of these interactions and hence represent essential or lethal proteins, while communities represent protein complexes involved in certain biological tasks. Most lethal proteins are strategically located and if disrupted, can lead to biological lethality. Hence, their study is relevant to understanding the causes of diseases, while detection of protein complexes is relevant to understand protein affinity and biological processes.

2. Modeling hubs and communities

In this paper, we model all networks as undirected simple graphs, $G = (V, E)$ with $|V| = n$ and $|E| = m$, where $d(v_i)$ denotes the degree of $v_i \in V$ and $N(v_i)$ the neighborhood of v_i . We begin by stating the following problem,

LIST DOMINATING SET problem [2]: For a given graph $G = (V, E)$, an integral list $L = \{l(v_1), \dots, l(v_n)\}$, $0 < l(v_i) \leq d(v_i)$, $1 \leq i \leq n$, and a positive integer k , a subset $D \subseteq V$ is called L -

dominating if for every $v_i \in V$, either $v_i \in D$ or $|N(v_i) \cap D| \geq l(v_i)$. The problem asks if we can find a D such that $|D| \leq k$.

The interesting contribution in this paper is to model both hubs and communities through the LIST DOMINATING SET problem. This is an interesting problem which generalizes the classical VERTEX COVER problem when $l(v_i) = d(v_i)$, and the DOMINATING SET problem when $l(v_i) = 1$. It is proved to be NP-complete in the general complexity domain and also W[2]-complete in the parameterized domain [2].

In order to arrive at a practical algorithm for this problem, we propose a new formulation, which we call the (k, λ) -LDS problem,

(k, λ) -LDS: For a given graph G , an integral list $S = \{s(v_1), \dots, s(v_n)\}$, $0 \leq s(v_i) < d(v_i)$, $1 \leq i \leq n$ and positive parameters k and λ , a subset $D \subseteq V$ is called (k, λ) -dominating if for every $v_i \in V$, either $v_i \in D$ or $|N(v_i) \cap D| \geq d(v_i) - s(v_i)$. The problem asks whether we can find a D , such that $|D| \leq k$ and $\sum_{v_i \in V \setminus D} s(v_i) \leq \lambda$.

Here, we refer to S as ‘slack’ list, given by $S = \{s(v_1), \dots, s(v_n)\} = \{d(v_1) - l(v_1), \dots, d(v_n) - l(v_n)\}$. For a vertex v_i outside D , $d(v_i) - s(v_i)$ gives the number of neighbors of v_i in D and $s(v_i)$ gives the number of neighbors outside D , while λ is an additional parameter introduced that bounds the number of edges allowable among the vertices outside D . In the following sections we show that this formulation is better suited for modeling real-world networks.

Modeling Hubs: Intuitively, hubs are *high-degree nodes* with which most of other nodes interact. Hubs have been previously modeled as vertex covers of the graph model $G(V, E)$, or by just blindly picking all nodes with degree above a certain threshold. Modelling hubs by vertex covers *forces* all linkages to be *covered* while threshold-based approach may miss some hub vertices that are below the degree threshold. We model hubs by (k, λ) -LDS for which the $s(v_i)$ values are *small*. This results in set D becoming the hub-set for the graph G covering all but $s(v_i)$ edges for each $v_i \in V \setminus D$. In the special case of $s(v_i) = 0$, D becomes the the vertex cover of G . However, the advantage (k, λ) -LDS offers is the flexibility by means of ‘tunability’ of S . We can control the number of neighbors inside and outside D by tuning S . This suits real-world networks like PPI networks very well. For instance, any non-hub protein can have few direct interactions with other non-hub proteins to achieve certain biological functionalities, but at the same time be densely linked to hub proteins.

Modeling Communities: Communities can be modeled as ‘Quasi Cliques’. Quasi Cliques are subgraphs

of a given graph with ‘small’ number of edges missing in contrast to cliques that are completely connected. We usually associate a quasi clique Q with two parameters, γ , the edge density and $|Q|$, the size of the quasi clique. Q is called γ -quasi clique if the ratio of the number of edges in Q to the total number of edges in a clique of the same size, is γ . Quasi cliques model real-world communities better than cliques. For instance, proteins within a complex interact with most of other proteins, but not necessarily all. It is interesting to note that when we consider the complement of a given graph as an instance of the (k, λ) -LDS problem to find the set D , the remaining vertices (in $V \setminus D$) form a quasi clique of the original graph. $s(v_i)$ gives the maximum number of edges missing from every vertex v_i in the quasi clique. Since $\sum_{v_i} s(v_i) \leq \lambda$, at most $\frac{\lambda}{2}$ edges can be missing from the quasi clique, which gives a lower bound on γ . Hence, our formulation not only bounds the number of missing edges but also specifies the edges corresponding to which vertices can be missing.

Example: Let $n = 100$, $s(v) = 2$, $k = 80$ and $\lambda = 20$. Hence, D covers all but 0,1 or 2 edges incident on $v \in V \setminus D$. Also, at most $\lambda = 20$ edges can be missing in the quasi clique of size $n - k = 20$ resulting in $\gamma \geq \{\binom{20}{2} - 20\} / \binom{20}{2} = 0.894$.

3. Solution to the (k, λ) -LDS problem

In this section, we propose a *fixed parameter tractable* (FPT) algorithm for (k, λ) -LDS. Problems solvable in $O(f(k)n^{O(1)})$, where k is a positive input parameter and $f(k)$ is a function of k alone, fall under FPT. Such algorithms are practical compared to the trivial $O(n^k)$ algorithms for NP-hard problems. The FPT algorithm for VERTEX COVER has recently been applied to detect cliques in gene networks [3], which proves the immense scope of such algorithms in applications. Through the following results we propose an FPT algorithm for our problem,

Lemma 1 *If for $v_i \in V$, $d(v_i) > k + \frac{\lambda}{2}$, then v_i is part of every (k, λ) -list dominating set D .*

Proof: Observe that if $v_i \in V \setminus D$, then D will be able to accommodate at most k neighbors of v forcing the remaining neighbors to be in $V \setminus D$. This means, $\sum_{v_i \in V \setminus D} s(v_i) > \lambda$. \square

Theorem 1 (k, λ) -LDS is FPT.¹

¹We would like to thank Venkatesh Raman and Saket Saurabh for some key observations during this work.

Proof: Let $U = V$, $D = \emptyset$, $k > 0$ and $\lambda > 0$. Lemma 1 leads to the following *Kernelization Rule*: If there is a vertex v_i such that $d(v_i) > k + \frac{\lambda}{2}$, then do $D := D \cup \{v_i\}$ and $U := U \setminus \{v_i\}$. We apply the rule exhaustively. If $|D| > k$, return NO.

After application of the rule, all vertices $v_i \in U$ have $d(v_i) \leq k + \frac{\lambda}{2}$. The number of edges in the graph $G[V \setminus D]$ is at most $(k - |D|)(k + \frac{\lambda}{2}) + \frac{\lambda}{2}$, if it is to have a solution. (Reason: D can accommodate $k - |D|$ more vertices, each of which has degree less than $k + \frac{\lambda}{2}$. Hence, moving any vertex from $V \setminus D$ to D can cover at most $k + \frac{\lambda}{2}$ edges. Plus, allowable $\frac{\lambda}{2}$ edges in $G[V \setminus D]$.) If the number of edges is more than this, return NO. This ends our kernelization step.

Next, we then perform a *Depth-bound Search* on the remaining graph. Our parameters are $k' = k - |D|$ and λ . At every step of the search we maintain two partitions of U :

- X : for every $v_i \in X$, $|N(v_i) \cap D| \geq d(v_i) - s(v_i)$,
- Y : for every $v_i \in Y$, $|N(v_i) \cap D| < d(v_i) - s(v_i)$.

We refer the vertices in X as *saturated* and those in Y as *unsaturated*. We pick an edge (u, v) from the remaining graph and branch upon the following conditions: either (u, v) is in $G[V \setminus D]$ or it is not. If not, then either u or v is in D . So, we recursively solve the problem by performing a three-way branching at each step. More specifically, if we move a vertex u (or v) into D , we set $D := D \cup \{u\}$ (or $D := D \cup \{v\}$) and $U := U \setminus \{u\}$ (or $U := U \setminus \{v\}$). By doing this, if we can find vertices $w \in Y$ such that $|N(w) \cap D| \geq d(w) - s(w)$, then we move w from Y to X , that is, $X := X \cup \{w\}$ and $Y := Y - \{w\}$ (that is, w is now saturated). We reduce the parameter k' by 1. However, if the edge (u, v) is retained in $G[V \setminus D]$, we reduce the parameter $\frac{\lambda}{2}$ by 1. At any particular step, if $k = 0$ and $Y \neq \emptyset$, we return NO. Else, we return YES and the set D .

The correctness of the algorithm is clear from the description. For the time complexity, observe that at each recursive step, we perform a three-way branching and the depth of the recursion tree is at most $d = k + \frac{\lambda}{2}$. Hence, the total number of nodes in the tree is bounded by $3^{k + \frac{\lambda}{2}}$. Since we spend polynomial time for kernelization and at each step of the search tree, the complexity of our algorithm is bounded in the worst case by $O(3^{(k + \frac{\lambda}{2})n})$, which is FPT. \square

Pruning the search tree: This is based on the observation that when we pick an edge (u, v) on any recursive call, if $s(u) = 0$ or $s(v) = 0$ or $\lambda = 0$ then we can avoid the third branch altogether. Hence, the complexity can be reduced to $O(c^{(k + \frac{\lambda}{2})n})$, $2 \leq c < 3$.

Algorithm 1 findLDS (G, k, λ)

```

Var: bool r;
if  $k = 0$  and  $Y \neq \emptyset$  then
    return NO;
end if
if  $Y = \emptyset$  and  $\sum s(v_i \in V \setminus D) \leq \lambda$  then
    return YES,  $D$ ;
end if
pick an edge  $(u, v)$ ; /* Mark as covered.*/
 $D \leftarrow D \cup \{u\}$ ;  $U \leftarrow U \setminus \{u\}$ ;
find and move saturated vertices  $w$  from  $Y$  to  $X$ ;
 $r \leftarrow \text{findLDS}(G, k - 1, \lambda)$ ;
if  $r = \text{YES}$  then
    return YES;
end if
 $D \leftarrow D \cup \{u\}$ ;  $U \leftarrow U \setminus \{u\}$ ;
find and move saturated vertices  $w$  from  $Y$  to  $X$ ;
 $r \leftarrow \text{findLDS}(G, k - 1, \lambda)$ ;
if  $r = \text{YES}$  then
    return YES;
end if
if  $s(u) > 0$  and  $s(v) > 0$  and  $\lambda > 1$  then
     $r \leftarrow \text{findLDS}(G, k, \lambda - 1)$ ;
    if  $r = \text{YES}$  then
        return YES;
    end if
else
    return NO; /*Unmark the edge.*/
end if
end findLDS routine;

```

Lemma 2² For a graph $G = (V, E)$, let $S = \{s(v_1), \dots, s(v_n)\}$ be an integral list, k and λ be positive parameters. If G has a (k, λ) -list dominating set D of size at most k such that $\sum_{v_i \in V} s(v_i) \leq \lambda$, then the subgraph $\bar{G}[V \setminus D]$ of the complement graph \bar{G} of G , is a γ -quasi clique of size at least $(n - k)$ with $\left\{ \binom{n-k}{2} - \frac{\lambda}{2} \right\} / \binom{n-k}{2} \leq \gamma \leq 1$.

Corollary 1 Applying (k, λ) -LDS on the complement graph \bar{G} gives a γ -quasi clique in the original graph G with γ same as above.

4. Experiments and results

We implemented our algorithm using C++ on a dual core 3.0GHz P4 Linux machine with 2GB RAM.

Input data: We first generated scale-free graphs using a preferential attachment model from [4] by setting the initial cycle length to be 20 and minimum degree to be 3. Next, we considered two samples of PPI networks from *saccharomyces cerevisiae* (budding yeast) consisting of 300 and 1000 high-confidence interactions involving 298 and 543 proteins respectively (Source: AP-MS experimental data from [5]).

Setting the parameters: k is the input parameter that determines the size of the desired solution set. For scale-free graphs, the algorithm tends to find a solution for (relatively) small values of k . This is due to the presence of small number of high-degree nodes which when

²The proof will appear in the long version of the paper.

Table 1. Scale-free graph model

n	m	$d_{max}(v)$	k	λ	$s(v)$	k'	T
300	1505	141	30	12	2	20	0.06
500	2767	234	70	16	2	56	0.07
900	4986	276	90	36	2	76	0.12

Table 2. Hubs in PPI networks

n	m	$d_{max}(v)$	k	λ	$s(v)$	k'	T
298	300	13	154	0	0	154	0.5
298	300	13	147	15	1	147	0.36
298	300	13	141	17	2	142	0.02
543	1000	33	295	0	0	295	0.37
543	1000	33	271	37	1	271	0.29
543	1000	33	261	59	2	261	0.08

picked, cover most of the interactions. However, if the graph is very dense (complement of scale-free graphs), a small k will typically not give a solution. k also determines the extent of kernelization. If k is too small, too many nodes might get included into the solution overshooting its size. If k is too large, the problem might not kernelize well. λ determines the number of slack edges allowable and so it should be much smaller than the sum of slacks of vertices to ensure most interactions are covered by the hubs. Finally, we use S as a ‘tuner’ to vary the slack of each vertex from 0 to $\log(n)$.

4.1. Results

Table 1 summarizes the results on scale-free graphs, where $d_{max}(v)$ is the maximum degree of the graph, k is the size of the desired hub set, $s(v)$ is the slack for every vertex, k' is the reduced parameter after kernelization and T is the time in seconds. The algorithm performs well for scale-free graphs because these are sparse and have small number of high-degree nodes.

Table 2 summarizes the results on PPI graphs. It’s interesting to note the reduction in the size of hub set as the slack is increased. This is one way to arrive at a selective set of hub proteins. One can use this to arrive at party (those linked to many) or date (those linked to only one) hub proteins. Finally, Table 3 summarizes the results of finding quasi cliques on PPI graphs. The γ -values are 1, 0.7423 and 0.7054 respectively.

Biological significance: From the Gene Ontology database [6] we can verify the shared properties of proteins. p -values are used to calculate the statistical and biological significance of the protein groups identified. Smaller p -values indicate more inter-related proteins. For the sample of size 300, our algorithm

Table 3. Quasi cliques in PPI networks

n	m	k	λ	$s(v)$	$n - k$	T
298	43953	295	0	0	3	1.58
298	43953	272	49	2	20	1.4
543	146153	512	137	2	31	2.29

found a hub protein YDR496C linked to YPL093W, YHR052W, YOL041C, YLL008W, YPL043W, YKL014C, YPL012W, YHR197W, YFR001W, YER126C, YDR087C, YLR009W, YDR101C. From the GO database, 13 out of 14 genes are known to be part of *ribosome biogenesis and assembly* process ontology with p -value=2.95e-14, while 12 out of 14 genes are known to be part of *nuclear lumen* component ontology with p -value=2.95e-10.

5. Conclusion and Future Work

We have proposed a FPT algorithm for the List Dominating Set, problem which models both hubs as well as communities in scale-free networks better than the traditional problems. A future direction for this work would be to partition graphs based on density-based methods and applying our algorithm on each partition to detect multiple quasi cliques in the graph. As a final word, FPT algorithms are usually analysed in the worst case, but tend to perform well in practice. These have immense scope in applications to data mining and pattern recognition. *Acknowledgment:* This work is supported in part by the National University of Singapore under Grant R252-000-289-112.

References

- [1] S Wutchy, *Scale-free behavior in protein domain networks*, Mol.Bio.Evolution 18(2001).
- [2] V Raman, S Saket, S Sriganesh, *Parameterized algorithms for generalized domination*, To appear in Proceedings of 2nd COCOA, LNCS(2008).
- [3] NE Baldwin, EJ Chesler, S Kirov, MA Langston, JR Snoddy, RW Williams, B Zhang, *Computational, Integrative, and Comparative Methods for the Elucidation of Genetic Coexpression Networks* Journal of Biomed. Biotech., pp172-180, Hindawi(2005).
- [4] J Diaz, J Petit, D Thilikos, *Kernels for the vertex cover problem on the preferential attachment model*, WEA, LNCS 4007, pp 231-240(2006).
- [5] The Biogrid, *PPI from Affinity-purification/mass-spectrometry(AP-MS) experiments*, <http://www.thebiogrid.org/>
- [6] Gene Ontology Database, <http://db.yeastgenome.org/cgi-bin/GO/goTermFinder.pl>