Network Clustering via Clique Relaxations: A Community Based Approach

Anurag Verma, Sergiy Butenko

Abstract In this paper, we present a general purpose network clustering algorithm based on a novel clique relaxation concept of *k-community*, which is defined as a connected subgraph such that endpoints of every edge have at least k common neighbors within the subgraph. A salient feature of this approach is that it does not use any prior information about the structure of the network. By defining a cluster as a k-community, the proposed algorithm aims to provide a clustering of a network into k-communities with varying values of k. Even though the algorithm is not designed to optimize any particular performance measure, the computational results suggest that it performs well on a number of criteria that are used in literature to evaluate the quality of a clustering.

1 Introduction

Network (graph) based data mining is an emerging field that studies network representations of data sets generated by an underlying complex system in order to draw meaningful conclusions regarding the system's properties. In a network representation of a complex system, the network's nodes typically denote the system's entities, while the edges between nodes represent a certain kind of similarity or relationship between the entities. Network clustering, aiming to partition a network into clusters of similar elements, is an important task frequently arising within this context. The form of each cluster in the partitioning is commonly specified through a predefined graph structure. Since a cluster is typically understood as a "tightly knit" group of elements, the graph theoretic concept of a *clique*, which is a subset of nodes inducing a complete subgraph, is a natural formalization of a cluster that has been used in

Anurag Verma, Sergiy Butenko
Texas A&M University, College Station, TX, USA. e-mail: anuragverma@tamu.edu, butenko@tamu.edu

many applications. This results in partitioning into "ideal" clusters, with the highest possible level of cohesiveness one can hope for.

The flawlessness of the clique structure as a theoretical formalization of a cohesive cluster turns into a "curse of perfection" when it comes to practical applications. Since each node in a clique is required to be connected to all other nodes in the clique, a highly cohesive structure might not get identified as a cluster by the mere absence of a few edges. In real life data sets, this is of critical importance because some edges could be missing either naturally or due to erroneous data collection. Moreover, given that networks arising in many important applications tend to be very large with respect to the number of nodes and very sparse in terms of the relative number of edges, the clique clustering usually results in meaninglessly large number of clusters in such situations. In addition, computing large cliques and good clique partitions are computationally challenging problems, as finding a maximum clique and a minimum clique partition in a graph are classical NP-hard problems [8].

To circumvent these drawbacks of cliques, researchers in several applied fields, such as social network analysis and computational biology, have defined and studied structures that relax some of the properties of cliques, and hence are aptly called clique relaxations. Some of the popular clique relaxations include s-plexes, that require each vertex to be connected to all but s other vertices [11]; s-clubs, that require the diameter of the induced subgraph to be at most s [2]; and γ-quasi-cliques, which require the density of the induced subgraph to be at least γ [1]. It should be noted that each of 1-plex, 1-club and 1-clique represents a clique. By relaxing the properties of a clique, namely the degree, diameter, and density, these clique relaxations capture clusters that are strongly but not completely connected. However, like the clique model, these clique relaxations still suffer from the drawback of being computationally expensive.

In 1983, Seidman [10] introduced the concept of a *k-core* that restricts the minimum number k of direct links a node must have with the rest of the cluster. Using k-cores to model clusters in a graph has considerable computational advantages over the other clique clique relaxation models mentioned above. Indeed, the problem of finding the largest k-core can be easily solved in polynomial time by recursively removing vertices of degree less than k. As a result, the k-core model has gained significant popularity as a network clustering tool in a wide range of applications. In particular, k-core clustering has been used as a tool to visualize very large scale networks [4], to identify highly interconnected subsystems of the stock market [9], and to detect molecular complexes and predict protein functions[5, 3]. On the downside, the size of a k-core may be much larger than k, creating a possibility of a low level of cohesion within the resulting cluster. Because of this, a k-core itself may not be a good model of a cluster, however, it has been observed that k-cores tend to contain other, more cohesive, clique relaxation structures, such as s-plexes, and hence computing a k-core can be used as a scale-reduction step while detecting other structures [6].

Most recently, the authors of the current paper proposed yet another clique relaxation model of a cluster, referred to as *k-community*, that aims to benefit from the positive properties of k-cores while ensuring a higher level of cohesion [12]. More

specifically, a k-community is a connected subgraph such that endpoints of every edge have at least k common neighbors within the subgraph. The k-community structure has proven to be extremely effective in reducing the scale of very large, sparse instances of the maximum clique problem [12]. This paper explores the potential of using the k-community structure as a network clustering tool. Even though the proposed clustering algorithm does not aim to optimize any of the quantitative measures of clustering quality, the results of numerical experiments show that it performs quite well with respect to most of such measures available in the literature.

The remainder of this paper is organized as follows. Section 2 provides the necessary background information. Section 3 outlines the proposed network clustering algorithm. Section 4 reports the results of numerical experiments on several benchmark instances, and Section 5 concludes the paper.

2 Background

In this paper, a network is described by a simple undirected graph $G = (V, E)$ with the set $V = \{1, 2, \ldots, n\}$ of nodes and the set E of edges. We call a pair of nodes u and v such that $(u, v) \in E$ *adjacent* or *neighbors*. For a node u, let $N_G(u) = \{v : (u, v) \in E\}$ denotes the neighborhood of u in G. Then the degree $deg_G(u)$ of u in G is given by the number of elements in $N_G(u)$. Let $\delta(G)$ denote the minimum degree of a node in G. For a subset C of nodes, $G[C] = (C, E \cap (C \times C))$ denotes the subgraph induced by C. Next we define two clique relaxation concepts that play a key role in this paper.

Definition 1 (k-core). A subset of nodes C is called a k-core if $G[C]$ is a connected graph and $\delta(G[C]) \geq k$.

Definition 2 (k-community). A connected subgraph $G' = (V', E')$ of G is a k-community if any two vertices $u, v \in V'$ are connected if and only if $(u, v) \in E$ and $|N_{G'}(u) \cap N_{G'}(v)| \geq k$.

Given a positive integer k, both of these structures are in essence trying to find a cluster of vertices that satisfies some minimum node degree requirements. In the case of k-core, the presence of each node has to be supported by the presence of at least k neighbors, while in the case of k-community, the presence of each edge has to be supported by the presence of at least k alternative edge-disjoint paths of length two. It is instructive to note that every k-community is also a $(k+1)$-core, but the converse is not true. Given a positive integer k, all k-communities of a graph G can be easily computed as outlined in Algorithm 1.

Algorithm 1 $k-$`Community`(G): Algorithm to find k-communities of G

1: $E' \leftarrow E$
2: **repeat**
3: $E \leftarrow E'$
4: $E' = \phi$
5: **for** every $(i, j) \in E$ **do**
6: **if** i, j have $\geq k$ common neighbors in G=(V,E) **then**
7: $E' \leftarrow E' \cup \{(i, j)\}$
8: **end if**
9: **end for**
10: **until** $E = E'$
11: **return** All the connected components of $G'(V, E')$ as the k-communities.

3 Clustering Algorithm

The algorithm described in this section is based on the idea of finding k-communities for large k and placing them in different clusters. To this end, we identify the largest k' such that the k'-community of G is non-empty, and place all k'-communities formed in distinct clusters. Once this has been done, all the nodes that have been placed in clusters are removed from G and the whole procedure is repeated till either k becomes small or no vertices are left to cluster. If any vertex is left to cluster, we attach it to the cluster that contains the most neighbors of that vertex. The procedure is described in Algorithm 2.

In this algorithm, we stop when k becomes small enough so that a k-community becomes meaningless. For example, any set of vertices that induce a tree will form a 0-community. While in some cases this might be the best possible option (the original graph is a forest), for most clustering instances we would like the vertices in a cluster to share more than just one edge with the remaining nodes. To get a good sense of what this small enough value should be, we evaluate how good the clustering is when the least k is in the range $[l, u]$, and pick the best one out of those. For the results in this paper, the values of u and l used were 8 and 0 respectively.

In Algorithm 2, we can replace k-community in step 5 with k-core, with the remaining steps of the algorithm as they are, to obtain a core-based clustering.

It should be noted that the clustering can further be improved by using a local search on any criteria provided such as modularity, performance, aixe and aixc as described in the DIMACS 2011 challenge [7]. However, in this algorithm we only provide results obtained from a general approach that is not designed to optimize any of these measures. Furthermore, another advantage of this method is that it does not use any prior information about the graph such as the number of clusters, degree distribution, etc. This makes it a very general approach that is applicable even when no information about the structure of the graph is available.

Some illustrations of clusterings found by the k-core and k-community approach described in this section are provided in Figure 1. It should be noted that, although k-communities are strictly stronger relaxations, the clustering formed by the core-

Algorithm 2 k-`Community Clustering`(G): Algorithm to find clusters in G

$G' \leftarrow G$
$\mathscr{C} \leftarrow \emptyset$
repeat
  $k \leftarrow$ highest integer such that k-community(G') is non-empty.
  Find all the k-Communities in G' and add them to $\mathscr{C}$.
  Find the set of vertices L that are not yet clustered.
  $G' \leftarrow G[L]$.
  if $k \leq u$ **then**
    $\mathscr{C}^k \leftarrow \mathscr{C}$
    for every $v \in L$ **do**
      Add v to the cluster $C^k \in \mathscr{C}^k$ which maximizes $|N(v) \cap C^k|$.
    end for
  end if
until k=1 or G' is empty
for every $v \in L$ **do**
  Add v to the cluster $C \in \mathscr{C}$ which maximizes $|N(v) \cap C|$.
end for
$\mathscr{C} \leftarrow \arg\max_{\mathscr{C}^k}$ `modularity`$(\mathscr{C}^k)$.
return $\mathscr{C}$

based approach can in some cases be better than that obtained using the community-based approach.

4 Computational Results

In this section we provide some computational results obtained by using the k-community and k-core clustering on the graph sets provided in the DIMACS 2011 challenge [7]. We have targeted the test sets *clustering* and *walshaw* for our experiments.

Table 1 presents the modularity and number of clusters of the clustering formed by the k-core and k-community clustering for 19 *clustering* and 18 *walshaw* graphs. For each graph, the higher of the two modularities as found be the two methods is highlighted in bold. It should be noted that the k-community clustering is better on 25 of the 37 graphs tested. A specially noteworthy graph is the *football* graph, in which each college plays more or less the same number of games, with games between different conferences also present. As a result, the whole graph is identified as a 7-core and placed in one single cluster. On the other hand, the k-community clustering is able to identify 13 conferences and sub-conferences as clusters. The second graph illustrated in Figure 1 is similar in structure to the football graph.

In addition, Table 1 also reports the time taken by the two approaches on each of the graphs. It can be seen that our approach is effective even for large graphs with more than $200,000$ vertices.

Table 2 presents the coverage, mirror coverage and performance for the same clusterings that were presented in table 1. For each graph, the table highlights the

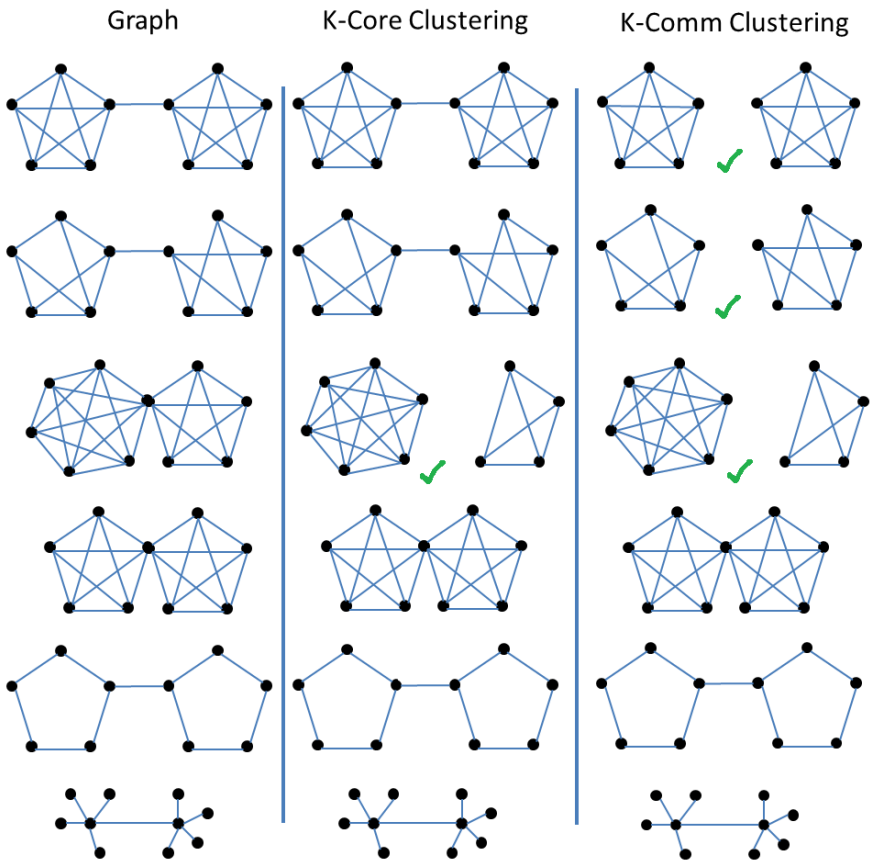

Fig. 1: Clustering found by the k-core and k-community approaches on some illustrative graphs. The checkmark indicates that the expected clustering was found.

higher performance entry amongst k-core and k-community clustering, subject to the condition that the coverage and mirror coverage were more than 0.5. Again, it should be noted that k-community clustering performs better than k-core clustering for 23 out of the 37 graphs tested.

5 Conclusion

This paper introduces k-community partitioning, which can be thought of as something between k-core clustering and clique partitioning. The use of polynomially computable k-community not only provides a faster approach, but also provides a

Table 1: Modularity of clustering found by using the community based and core based approaches. The modularity that is higher between the two methods is highlighted in bold.

Graphs			**core-based**			**community-based**		
	n	**m**	**Mod**	**Clusters**	**Time(s)**	**Mod**	**Clusters**	**Time**
Clustering								
karate	34	78	0.25	3	0.05	**0.40**	4	0.03
dolphins	62	159	0.11	5	0.05	**0.49**	6	0.03
polbooks	105	441	**0.49**	3	0.03	0.39	8	0.05
adjnoun	112	425	**0.12**	4	0.03	0.04	2	0.03
football	115	613	0.00	1	0.09	**0.57**	13	0.06
celegans_metabolic	453	2025	0.23	19	0.08	**0.28**	30	0.06
jazz	198	2742	**0.33**	6	0.08	0.28	8	0.09
netscience	1589	2742	**0.86**	576	0.20	0.85	590	0.19
email	1133	5451	0.30	7	0.14	**0.34**	72	0.19
power	4941	6594	0.79	13	0.88	**0.85**	189	0.88
polblogs	1490	16715	**0.20**	274	0.36	0.16	279	0.56
hep-th	8361	15751	0.56	2509	1.16	**0.67**	2359	2.06
PGPgiantcompo	10680	24316	**0.77**	104	2.67	0.75	84	2.23
cond-mat	16726	47594	0.62	2852	5.33	**0.65**	3113	7.81
as-22july06	22963	48436	0.43	33	3.70	**0.50**	162	3.95
cond-mat-2003	31163	120029	0.53	3568	23.55	**0.54**	4339	24.42
G_n_pin_pout	100000	501198				**0.20**	4484	157.55
coAuthorsCiteseer	227320	814134	**0.74**	2705	421.96	**0.74**	3696	449.62
coAuthorsDBLP	299067	977676	0.64	1911	651.23	**0.65**	5904	879.48
Walshaw								
add20.graph	2395	7462	**0.57**	11	0.23	**0.57**	11	0.28
data.graph	2851	15093	0.33	3	0.34	**0.49**	5	0.27
add32.graph	4960	9462	0.47	113	0.45	**0.87**	70	0.25
bcsstk33.graph	8738	291583	**0.21**	35	4.94	0.19	77	12.05
whitaker3.graph	9800	28989	**0.43**	2	1.53	0.00	1	1.61
wing_nodal.graph	10937	75488	0.12	19	4.80	**0.55**	26	2.31
vibrobox.graph	12328	165250	**0.50**	105	3.53	0.42	269	4.26
bcsstk29.graph	13992	302748	0.16	376	5.91	**0.54**	517	14.67
cti.graph	16840	48232	**0.47**	2	5.50	0.00	1	4.63
memplus.graph	17758	54196	**0.52**	14	6.89	**0.52**	14	6.77
bcsstk30.graph	28924	1007284	**0.60**	420	60.82	0.49	791	133.61
bcsstk31.graph	35588	572914	**0.74**	266	25.24	0.69	414	27.34
bcsstk32.graph	44609	985046	**0.75**	402	55.06	0.66	629	131.68
fe_body.graph	45087	163734	**0.64**	581	39.33	0.59	10291	52.38
finan512.graph	74752	261120	0.40	513	77.64	**0.78**	96	27.78
598a.graph	110971	741934	0.48	24	199.58	**0.59**	170	251.60
fe_ocean.graph	143437	409593	**0.09**	20	460.94	0.00	1	311.90
m14b.graph	214765	1679018	0.01	3	1074.65	**0.69**	134	394.47

Table 2: Performance value of clustering found by using the community-based and core-based approaches. The performance that is higher between the two methods given that the coverage and mirror coverage are higher than 0.5 is highlighted in bold. The clustering used is the same as the one used in Table 1.

Graphs			**core-based**			**community-based**		
	n	**m**	**Cov**	**M-Cov**	**Perf**	**Cov**	**M-Cov**	**Perf**
Clustering								
karate	34	78	0.79	0.61	0.64	0.73	0.80	**0.79**
dolphins	62	159	0.87	0.41	0.45	0.68	0.88	**0.86**
polbooks	105	441	0.85	0.71	0.73	0.56	0.89	**0.87**
adjnoun	112	425	0.70	0.55	**0.56**	0.73	0.17	0.21
football	115	613	1.00	0.00	0.09	0.66	0.99	**0.96**
celegans_metabolic	453	2025	0.48	0.87	0.86	0.44	0.92	0.91
jazz	198	2742	0.58	0.86	**0.82**	0.43	0.90	0.84
netscience	1589	2742	0.88	1.00	**1.00**	0.86	1.00	**1.00**
email	1133	5451	0.68	0.71	**0.71**	0.39	0.94	0.94
power	4941	6594	0.97	0.81	0.81	0.86	0.99	**0.99**
polblogs	1490	16715	0.40	0.86	0.85	0.33	0.88	0.87
hep-th	8361	15751	0.68	0.92	0.92	0.67	1.00	**1.00**
PGPgiantcompo	10680	24316	0.80	0.96	**0.96**	0.77	0.96	**0.96**
cond-mat	16726	47594	0.67	0.95	0.95	0.65	1.00	**1.00**
cond-mat-2003	31163	120029	0.59	0.94	0.94	0.55	1.00	**1.00**
G_n_pin_pout	100000	501198				0.38	0.81	0.81
as-22july06	22963	48436	0.72	0.73	0.73	0.65	0.90	**0.90**
coAuthorsCiteseer	227320	814134	0.76	0.98	0.98	0.75	0.99	**0.99**
coAuthorsDBLP	299067	977676	0.68	0.95	0.95	0.65	0.99	**0.99**
Walshaw								
add20	2395	7462	0.70	0.89	0.89	0.70	0.89	**0.89**
data	2851	15093	0.98	0.44	0.44	0.98	0.58	**0.58**
add32	4960	9462	0.96	0.63	0.63	0.95	0.94	**0.94**
bcsstk33	8738	291583	0.90	0.39	0.39	0.76	0.44	0.44
whitaker3	9800	28989	0.98	0.45	0.45	1.00	0.00	0.00
wing_nodal	10937	75488	0.98	0.17	0.17	0.64	0.91	**0.91**
vibrobox	12328	165250	0.63	0.88	**0.88**	0.45	0.99	0.99
bcsstk29	13992	302748	0.94	0.41	0.42	0.55	0.99	**0.99**
cti	16840	48232	0.99	0.47	0.47	1.00	0.00	0.00
memplus	17758	54196	0.82	0.76	**0.76**	0.82	0.76	**0.76**
bcsstk30	28924	1007284	0.69	0.94	**0.94**	0.49	1.00	1.00
bcsstk31	35588	572914	0.78	0.95	**0.95**	0.76	0.91	0.91
bcsstk32	44609	985046	0.81	0.96	**0.96**	0.74	0.94	0.94
fe_body	45087	163734	0.96	0.72	0.72	0.74	0.85	**0.85**
finan512	74752	261120	0.95	0.63	0.63	0.81	0.98	**0.98**
598a	110971	741934	0.95	0.54	0.54	0.73	0.85	**0.85**
fe_ocean	143437	409593	0.99	0.10	0.10	1.00	0.00	0.00
m14b	214765	1679018	1.00	0.01	0.01	0.77	0.92	**0.92**

more effective clustering method by being able to identify cohesive structures that might not be cliques. k-Community clustering also provides advantages over k-core clustering due to the more cohesive nature of a k-community. As our computational results show, both the k-core and k-communities perform well for certain graphs, but k-community approach outperforms the k-core approach in general.

References

[1] J. Abello, M.G.C. Resende, and S. Sudarsky. Massive quasi-clique detection. In S. Rajsbaum, editor, *LATIN 2002: Theoretical Informatics*, pages 598–612, London, 2002. Springer-Verlag.

[2] R.D. Alba. A graph-theoretic definition of a sociometric clique. *Journal of Mathematical Sociology*, 3:113–126, 1973.

[3] M. Altaf-Ul-Amin, K. Nishikata, T. Koma, T. Miyasato, Y. Shinbo, M. Arifuzzaman, C. Wada, and M. Maeda et al. Prediction of protein functions based on k-cores of protein-protein interaction networks and amino acid sequences. *Genome Informatics*, 14:498–499, 2003.

[4] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani. k-core decomposition: a tool for the visualization of large scale networks. *CoRR*, abs/cs/0504107, 2005.

[5] G. D. Bader and C. W. V. Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*, 4(2), 2003.

[6] B. Balasundaram, S. Butenko, and I.V. Hicks. Clique relaxations in social network analysis: The maximum k-plex problem. *Operations Research*, 59:133–142, 2011.

[7] DIMACS. 10th DIMACS Implementation Challenge: Graph Partitioning and Graph Clustering. http://www.cc.gatech.edu/dimacs10/, 2011.

[8] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[9] J. Idicula. Highly interconnected subsystems of the stock market. NET Institute Working Paper No. 04-17, 2004. Available at SSRN: http://ssrn.com/abstract=634681.

[10] S. B. Seidman. Network structure and minimum degree. *Social Networks*, 5:269–287, 1983.

[11] S. B. Seidman and B. L. Foster. A graph theoretic generalization of the clique concept. *Journal of Mathematical Sociology*, 6:139–154, 1978.

[12] A. Verma and S. Butenko. Maximum clique problem on very large scale sparse networks.