Network Clustering via Clique Relaxations: A Community Based Approach

Anurag Verma, Sergiy Butenko
Texas A&M University

February 2012

TEXAS A&M
ENGINEERING
INDUSTRIAL
and SYSTEMS
ENGINEERING

Table of contents

1 Introduction

2 Clique Relaxations

3 Network Clustering using Clique Relaxations

WordNet dictionary

An exclusive circle of people with a common purpose.

Luce and Perry (1949)

Social clique – a group of people that know (are friends of) all other people in the group.

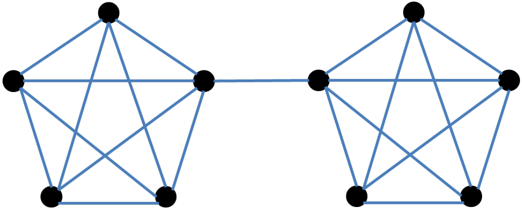
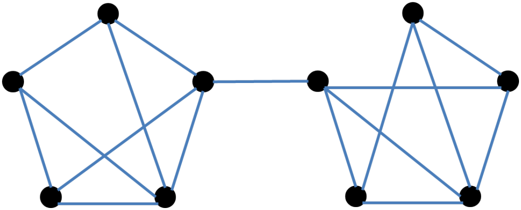


Figure: The "perfect cluster"

Perfect may mean impractical. Some examples:



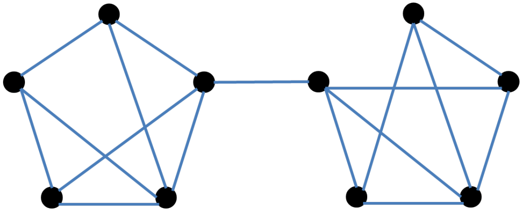
Furthermore, finding large cliques is often computationally expensive.

Instead of cliques, using clique relaxations could provide better results:

- Restricting a violation of a clique defining property:
  - **s-plex:** Each vertex is connected to all but s nodes in the induced subgraph.
  - **s-club:** The diameter of the induced subgraph is at most s.
  - γ**-quasi-clique:** The density of the subgraph is at least γ.
- Ensuring the presence of a clique defining property:
  - **k-Core:** Each vertex has k neighbors in the induced subgraph.
  - **k-Community:** Every edge has at least k neighboring nodes (A node is a neighbor of an edge if it is a neighbors with both its end points).

A clique of size ω is a 1-plex, 1-club, 1-quasi-clique, $(\omega - 1)$-core and $(\omega - 2)$-community.

Examples of clique relaxations



Both sets of 5 nodes form a
2-plex,
2-club,
3-core,
1-community,
0.8-quasi-clique.

Some background on k-community

- k-community was introduced to develop scale reduction techniques for the maximum clique problem.
- If we know a lower bound l_ω on the clique size, then we can be sure that the $(l_\omega - 2)$-community contains the maximum clique in the graph.
- This is much tighter than finding the $(l_\omega - 1)$-core as done by Abello et al (1999).
- Iteratively remove edges that have less than k neighboring nodes ($O(mk\Delta)$).
- Maximum cliques were obtained on all graphs in the SNAP database (graphs with up to 4 million nodes).

Why k-community?

- All the clique relaxations defined earlier help us in identifying large clusters.
- For participating in this challenge, we chose k-communities for
  - Tightness as a cluster when compared to k-cores.
  - Computational ease when compared to the remaining relaxations (can be found in polynomial time).

Participating in the challenge

- Found out about the challenge two weeks before the extended paper submission deadline!
- Since k-communities identify large cohesive clusters, why not use them for developing a clustering algorithm?

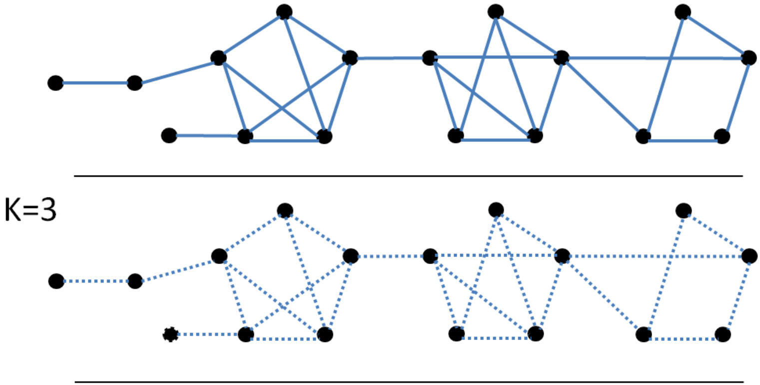
k-Community clustering

Consider the following algorithm:

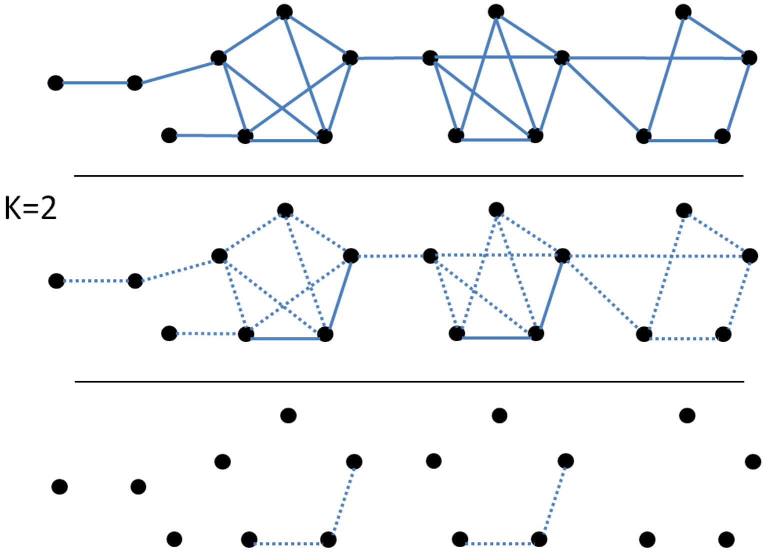
1. Find the largest k' such that the k'-community of G is non-empty.
2. Place all the k'-communities of G in distinct clusters.
3. Remove from G all the nodes that have been placed in a cluster.
4. Repeat steps 1-3 until $k' = 0$ or all nodes have been clustered.

Computational complexity: $O(m\Delta^3)$.

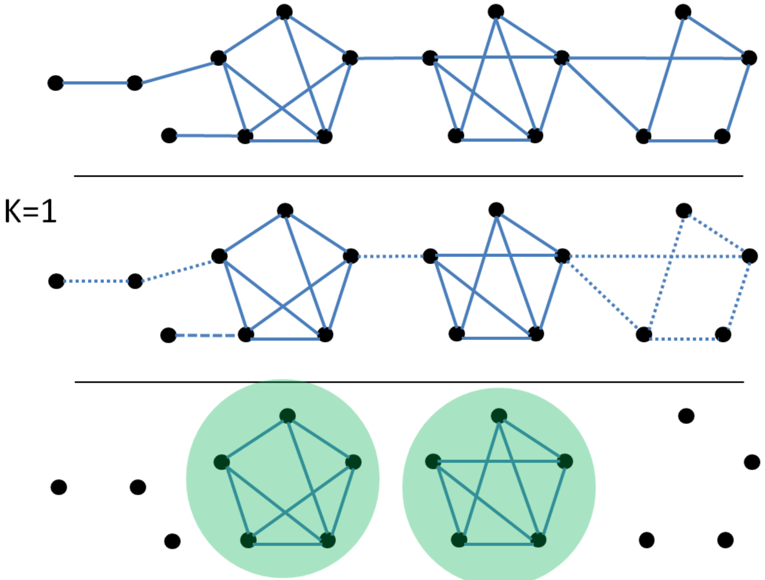
k-Community clustering



k-Community clustering

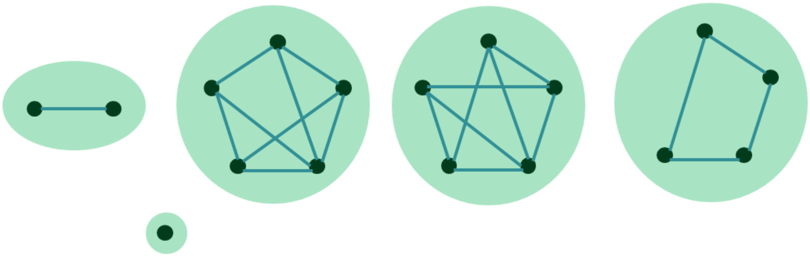


k-Community clustering

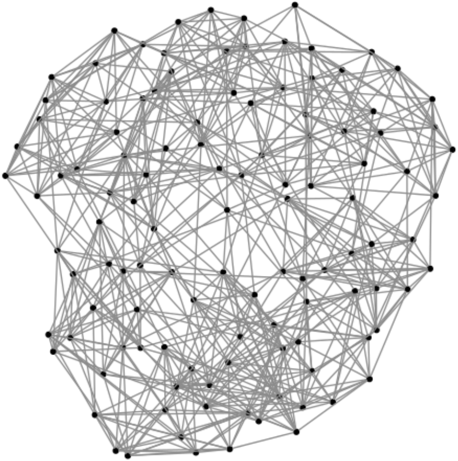


K=0

Final Clustering



Football Networks: node - NCAA football teams, edge - played a game in 2001.[1]



115 nodes, 613 edges.

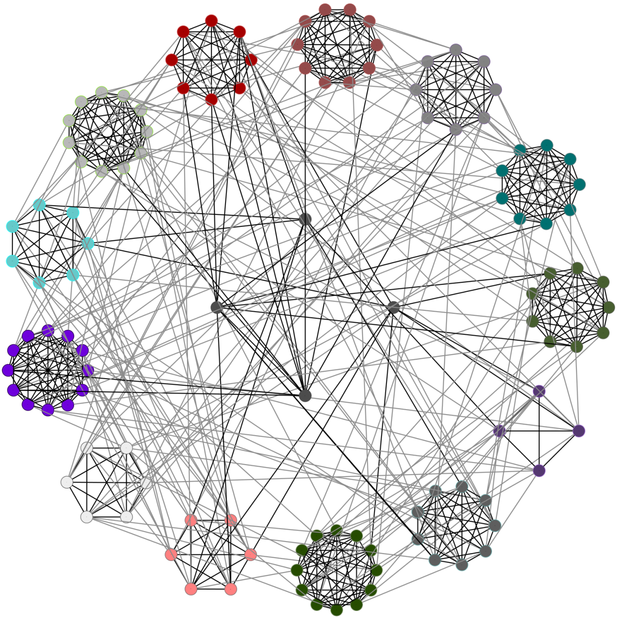
[1] Girvan & Newman, Proc Natl. Acad. Sci., 2002.

The football example

Actually:
11 Conferences,
5 independents.

Clustering:
13 Clusters,
4 independents.

Diagram generated by GraphViz.



k-Community clustering

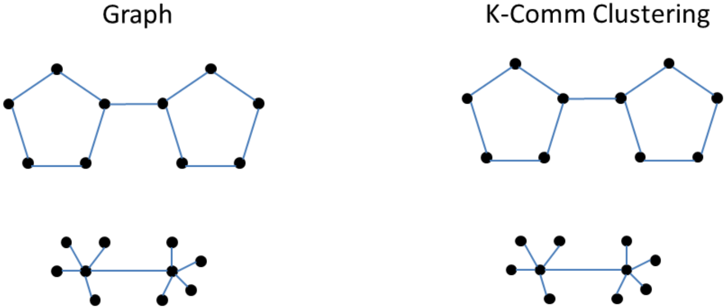


Figure: Definition of a cluster matters! In both the cases, 2-clubs would cluster the graph.

k-Community clustering

1. Measures to evaluate a clustering: Modularity, Performance, Minimum intra-cluster density, Average inter-cluster expansion.
2. The proposed clustering algorithm does not aim to optimize any of the quantitative measures of clustering quality.
3. The results of numerical experiments show that it performs quite well with respect to many such measures.
4. Algorithm coded in C++.
5. Computations on a desktop computer.
    - Intel Core i7 860 @ 2.80GHz 8-core, 64 bit, 8GB RAM.

Table: Modularity of clusters found by using the k-community clustering. Single core desktop PC.

Name	n	m	Mod	Cov	M-Cov	Perf	Aixc	Aixe	Mid	Time
celegans_metabolic	453	2025	0.31	0.57	0.82	0.82	0.50	3.34	0.04	0.03
email	1133	5451	0.39	0.44	0.95	0.95	0.60	5.65	0.02	0.08
polblogs	1490	16715	0.21	0.40	0.89	0.88	0.09	2.22	0.03	0.50
power	4941	6594	0.85	0.87	0.99	0.99	0.17	0.49	0.01	0.13
PGPgiantcompo	10680	24316	0.73	0.74	1.00	1.00	0.21	0.99	0.01	0.72
astro-ph	16706	121251	0.54	0.54	1.00	1.00	0.40	2.88	0.04	9.12
memplus	17758	54196	0.54	0.64	0.97	0.97	0.25	1.42	0.00	0.89
as-22july06	22963	48436	0.36	0.76	0.59	0.59	0.45	2.20	0.00	1.31
cond-mat-2005	40421	175691	0.51	0.51	1.00	1.00	0.45	2.41	0.01	11.76
kron_g500-simple-logn16	65536	2456071	-0.02	0.34	0.67	0.67	0.00	0.39	0.00	708.82
preferentialAttachment	100000	499985	0.00	0.92	0.02	0.02	0.91	26.71	0.00	31.84
G_n_pin_pout	100000	501198	0.18	0.45	0.72	0.72	0.80	8.83	0.00	44.13
smallworld	100000	499998	0.57	0.57	1.00	1.00	0.50	4.93	0.13	10.25
luxembourg.osm	114599	119666	0.01	1.00	0.01	0.01	0.28	0.84	0.00	7.94
rgg_n_2_17_s0	131072	728753	0.61	0.61	1.00	1.00	0.45	4.72	0.20	19.16
caidaRouterLevel	192244	609066	0.60	0.62	0.99	0.99	0.40	2.03	0.00	116.65
coAuthorsCiteseer	227320	814134	0.69	0.69	1.00	1.00	0.32	1.88	0.01	170.59
citationCiteseer	268495	1156647	0.43	0.45	0.98	0.98	0.50	4.25	0.00	287.20
coPapersDBLP	540486	15245729	0.67	0.67	1.00	1.00	0.44	9.69	0.18	14383.00
eu-2005	862664	16138468	0.44	0.45	0.99	0.99	0.67	21.58	0.00	121419.00
in-2004	1382908	13591473	0.63	0.63	1.00	1.00	0.40	12.89	0.00	33522.00
belgium.osm	1441295	1549970	0.02	0.99	0.02	0.02	0.32	0.99	0.00	1212.34
333SP	3712815	11108633	0.00	1.00	0.00	0.00	0.49	2.89	0.01	9968.71

Table: Comparison Modularity found by Newman's Fast Algorithm on some select graphs

Graph	Newman[2]	k-Core	k-Community
Jazz	0.44	**0.33**	0.28
Celegans_Metabolic	0.40	0.20	**0.31**
Email	0.48	0.31	**0.39**
PGP	0.73	0.67	0.73

[2] Duch & Arenas, Phy. Rev. E, 2005.

Competing for modularity...

1. Should we stop k' from going all the way to zero to make sure only tight clusters are formed?
2. Should assign unclustered nodes to one of the clusters based on some criteria?
3. Should we join two clusters formed during the course of the algorithm? Should we split clusters?
4. Should we move individual nodes from one cluster to another if that improves modularity?

Competing for modularity...

1. Should we stop k' from going all the way to zero to make sure only tight clusters are formed? YES!!
2. Should assign unclustered nodes to one of the clusters based on some criteria? YES!
3. Should we join two clusters formed during the course of the algorithm? Should we split clusters?umm..
4. Should we move individual nodes from one cluster to another if that improves modularity?maybe.. (highest increase 0.04)

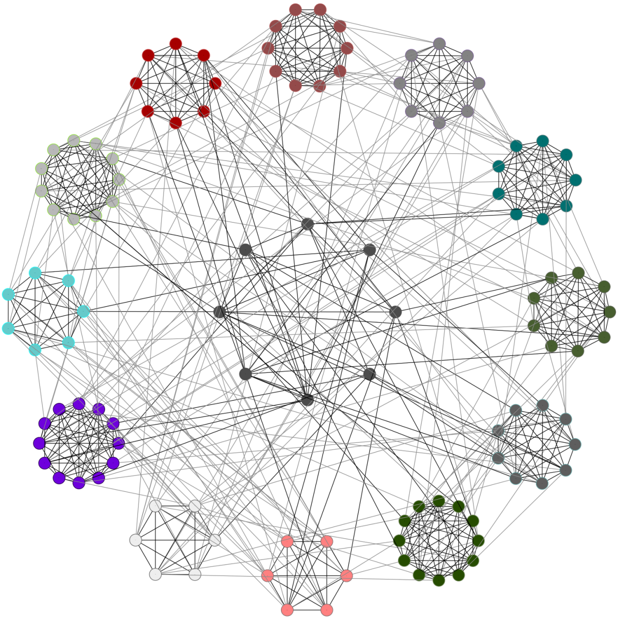
The football example

Actually:
11 Conferences,
5 independents.

No Enhancement:
13 Conferences,
4 independents.

Clustering:
12 Clusters,
8 independents.

Diagram generated by GraphViz.



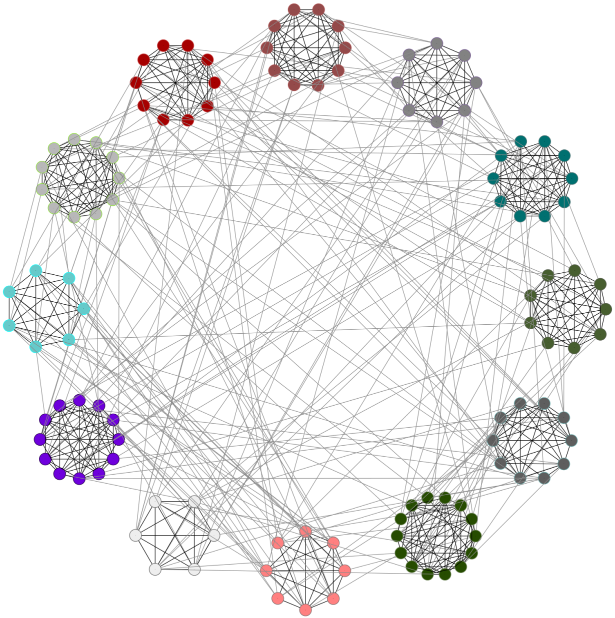
The football example

Actually:
11 Conferences,
5 independents.

No Enhancement:
13 Conferences,
4 independents.

Clustering:
12 Clusters,
0 independents.

Diagram generated by GraphViz.

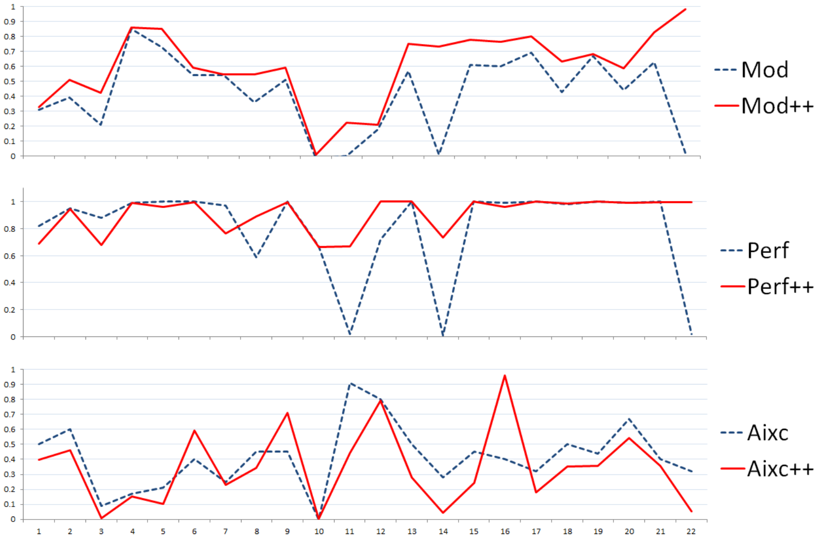


Computational Results with the Enhancements

TEXAS A&M ENGINEERING INDUSTRIAL and SYSTEMS ENGINEERING

Table: Modularity of clusters found by using the k-community clustering with the enhancements. Single core desktop PC.

Name	n	m	Mod	Cov	M-Cov	Perf	Aixc	Aixe	Mid	Time
celegans_metabolic	453	2025	0.33	0.70	0.69	0.69	0.40	2.98	0.03	0.11
email	1133	5451	0.51	0.57	0.95	0.95	0.46	4.11	0.07	0.36
polblogs	1490	16715	0.43	0.93	0.67	0.68	0.01	0.02	0.04	0.45
power	4941	6594	0.86	0.87	0.99	0.99	0.15	0.43	0.01	0.83
PGPgiantcompo	10680	24316	0.85	0.89	0.96	0.96	0.10	0.48	0.00	1.61
astro-ph	16706	121251	0.59	0.61	1.00	1.00	0.59	2.10	0.00	39.10
memplus	17758	54196	0.54	0.77	0.76	0.76	0.23	1.53	0.00	9.02
as-22july06	22963	48436	0.55	0.66	0.89	0.89	0.34	1.26	0.00	12.56
cond-mat-2005	40421	175691	0.59	0.60	1.00	1.00	0.71	1.96	0.02	48.19
kron_g500-best-logn16	65536	2456071	0.01	0.52	0.66	0.66	0.00	0.07	0.00	869.34
preferentialAttachment	100000	499985	0.23	0.56	0.67	0.67	0.44	4.41	0.00	216.45
G_n_pin_pout	100000	501198	0.21	0.21	1.00	1.00	0.79	7.97	0.02	317.19
smallworld	100000	499998	0.75	0.75	1.00	1.00	0.28	2.83	0.02	109.09
luxembourg.osm	114599	119666	0.73	0.99	0.73	0.73	0.04	0.11	0.00	193.98
rgg_n_2_17_s0	131072	728753	0.78	0.78	1.00	1.00	0.24	2.79	0.08	245.13
caidaRouterLevel	192244	609066	0.77	0.81	0.96	0.96	0.96	2.15	0.00	386.70
coAuthorsCiteseer	227320	814134	0.80	0.80	1.00	1.00	0.18	1.58	0.00	674.91
citationCiteseer	268495	1156647	0.63	0.65	0.99	0.99	0.35	3.11	0.00	790.05
coPapersDBLP	540486	15245729	0.68	0.68	1.00	1.00	0.35	11.78	0.09	12297.60
eu-2005	862664	16138468	0.59	0.59	0.99	0.99	0.54	21.72	0.00	120100.00
in-2004	1382908	13591473	0.83	0.84	1.00	1.00	0.36	9.51	0.00	49121.80
belgium.osm	1441295	1549970	0.98	0.98	1.00	1.00	0.05	0.13	0.00	23532.90


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Mod
Mod++
1
0.8
0.6
0.4
0.2
0
Perf
Perf++
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
Aixc
Aixc++

Comparison with Newman's Fast Algorithm

Table: Comparison of Modularity found by Newman's Fast Algorithm on some select graphs

Graph	**Newman**	**k-Comm**	**k-Comm++**
Jazz	0.44	0.28	0.43
Celegans_Metabolic	0.40	0.31	0.33
Email	0.48	0.39	0.51
PGP	0.73	0.73	0.85

Conclusion

- Introduced a general purpose clustering algorithm (not modularity maximization) based on clique relaxations.
- User can customize the algorithm by defining what a cluster should look like.
- Algorithm does not aim to optimize any performance measure used in the challenge.
- Using k-community as a structure does well for a number of clustering quality measures.
- Enhancements to the basic algorithm can be designed according to requirements.

Thank You!