Using Stable Communities for Maximizing Modularity [⋆]

S. Srinivasan and S. Bhowmick

Department of Computer Science, University of Nebraska at Omaha

Abstract. Modularity maximization is an important problem in network analysis and is used for detecting communities in networks. However, like many other NP-hard combinatorial optimization problems, the results–the value of the modularity as well as the community membership–are affected by the order in which the vertices in the network are processed. We define stable communities as groups of vertices that always remain in the same community, independent of the vertex perturbations to the input graph and develop an algorithm that can identify pockets of such stable communities. We show through empirical results that identifying the stable communities as a preprocessing step before executing an agglomeration based community detection algorithm can increase the modularity of the overall network.

1 Introduction

A complex network has community structure if it divides naturally into groups of vertices with denser connections within a community and sparser connections across the communities. Community detection is an important analysis tool in many applications including biological networks [20], collaboration patterns [1] and epidemiology [4]. A popular metric for measuring the quality of communities is based on computing the modularity of the network [14]. Generally, a high modularity, indicates a better partitioning of the network into communities.

Maximizing modularity is an NP-hard problem [5]. Consequently there exist many classes of heuristics to maximize modularity including agglomerative, divisive and spectral methods [17]. Recent publications [9] have also shown that modularity maximization suffers from the problem of resolution limit, that is communities smaller than a certain size cannot be detected, and in certain networks even the highest modularity value does not always reveal the optimal division. Moreover, like all NP-hard combinatorial optimization problems, the value of modularity and the division of the vertices into communities is dependent on the order in which vertices are processed.

We believe that these instabilities may occur in cases where the entire network is not modular enough to be classified into communities. Rather, some

[⋆] This work has been supported by the College of Information Science and Technology, University of Nebraska at Omaha (UNO) and the FIRE grant from the UNO Office of Research and Creative Activity.

portions of networks create naturally forming communities, while the remaining vertices are allocated to communities based on parameters of the underlying algorithms and permutations to the input. We define a *stable community* as a group of vertices that are always allocated to the same community independent of the perturbations to the input. The number of stable communities in a network can provide an estimate of its inherent modularity. In this paper, we develop an algorithm for identifying such stable communities. We show that by combining the vertices in stable communities as a preprocessing step to agglomerative community detection we can increase the value of the modularity.

The rest of the paper is arranged as follows. In Section 2, we provide definitions for the network terminology used in this paper and a brief overview of some of the related research in this area. In Section 3, present our algorithm to identify stable communities in networks. In Section 4 we demonstrate using empirical results, on a test suite of networks, how combining the identified stable communities in the preprocessing step can improve the modularity value. We conclude in Section 5 with discussions and plans for future research.

2 Terminology and Related Research

We define some terms associated with network analysis that are used in this paper. A network (or graph) $G = (V, E)$ is defined as a set of vertices V and a set of edges E. An edge $e \in E$ is associated with two vertices u, v which are called its *endpoints*. A vertex u is a *neighbor* of v if they are joined by an edge. The *degree* of a vertex u is the number of its neighbors. A *path*, of length l, in a graph G is an alternating sequence of $v_0, e_1, v_1, e_2, \ldots, e_l, v_l$ vertices and edges, such that for $j = 1, \ldots, l$; v_{j-1} and v_j are the endpoints of edge e_j, with no edges or internal vertices repeated. A *cycle* is a path whose initial and final vertices are identical.

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 *betweenness centrality* of a vertex (or edge) is the ratio of the number of shortest paths traversing through it to the total number of shortest paths in the graph. Detailed descriptions of these terms can be found in [15].

The metric of modularity of a network was proposed by Newman and Girvan [14] and is based on the concept that random networks do not form strong communities. Given a partition of a network into M groups, let C_{ij} represent the fraction of total links starting at a node in group i and ending at a node in group j. Let $a_i = \sum_j C_{ij}$ correspond to the fraction of links connected to subgroup i. Under random connections, the probability of links that begin at a node in i, is a_i, and the probability of links that end at a node in j, a_j. Thus, the expected number of within-community links, between nodes with group i, is a_i^2. The actual fraction of links within each group is C_{ii}. So, a comparison of the actual and expected values, summed over all groups of the partition gives us the modularity, which is the deviation of the partition from random connections: $Q = \sum(C_{ii} - a_i^2)$.

Maximizing modularity is a popular method for finding good communities in networks. However finding the optimal modularity is an NP-hard problem [5]. There exist many heuristics for maximizing modularity including spectral partitioning, divisive and agglomerative methods [17]. Clauset *et. al.* [6], developed a greedy agglomerative algorithm (to be referred as the CNM algorithm) which initially considers every node in the network as an individual community, then computes the increase in modularity for each pair of connected communities. The pair of communities with the highest increase in modularity are then merged. The process is iteratively repeated until there is no combination of vertices that increase modularity. One of the disadvantages of the CNM method is that it cannot backtrack to correct any earlier mistake made due to a greedy choice. This is addressed in the Louvian method proposed in [3]. In this method at each iteration vertices are moved from one community to another to increase the overall modularity of the network.

Agglomerative methods, and modularity maximization in general, are restricted by the resolution limit problem, that is they are unable to detect communities smaller than a certain size. The Louvian method provides a hierarchy of how the communities merged, and the lower levels can indicate the communities of smaller size. Other solutions to obtaining good communities include approaches such as finding small seeded communities [19] or pruning some of the outlier vertices to obtain better modularity.

The effect of perturbations of the input to the community detection results is still a major issue. Karrer *et. al.* [12] conducted a study of robust or statistically significant communities by perturbing the connectivity of the network and then comparing change in community structures. It should be noted that just comparing the value of the modularity does not give a clear estimate of the difference in results since two different partition schemes on the same network can give identical modularity values. A stronger comparison is based on comparing the allocation of the vertices within the partitions. One method of comparing across two partitions (obtained from different algorithms) is by using the Rand index [18]. Given two different partitions of a network, the Rand index is computed as follows; Let a be the pair of vertices that are present in the same community over both the partitions, let b be the pair of nodes that were in different communities for both the partitions, then the Rand index is computed as the ratio of the sum of a and b over all possible pairs of vertices. A high Rand index (maximum value 1) indicates that the two partitions are equal and a low Rand index indicates that they are very dissimilar.

3 Stable Communities in Networks

Our contribution is to characterize stable communities, develop an algorithm to identify some of them and then combine the identified stable communities as a preprocessing step for community detection. While this approach can be applied to any agglomeration based method, in this paper we use CNM as the base modularity maximization algorithm.

3.1 Effect of Vertex Ordering on Community Detection

In agglomeration based techniques, the order in which the vertices, and later the communities, are processed, significantly affects the final partition–and thereby the modularity value. As a first step to understanding the effect of vertex ordering on modularity, we permuted a test-suite of networks as follows;

1. *Random (RN)*: This is a random ordering of the vertices of the network.
2. *High Degree (HD)*: The vertices are ordered according to the descending order of degree.
3. *Low Degree (LD)*: The vertices are ordered according to the ascending order of degree.
4. *Betweenness Centrality (BC):* The betweenness centrality of the vertices are computed. We form clusters containing one vertex with low betweenness centrality and its neighbors. Vertices in the cluster are ordered consecutively. Clusters with vertices of low betweenness centrality are given lower numbers, indicating that they are to be processed earlier.
5. *Clustering Coefficient (CC):* The ordering is similar to BC. We form clusters containing one vertex with high clustering coefficient and its neighbors. The clusters of vertices with high clustering coefficient are ordered to be processed earlier.

Our test-suite consists of nine unweighted and undirected networks obtained from the clustering instances in the DIMACs website [7]. These are; **(i)** Jazz (network of jazz musicians) [10], **(ii))** PolBooks (network of books about USA politics) [16], **(iii)** Chesapeake (ecosystem from Chesapeake bay) [2],**(iv)** Power(topology of power grid in the western states of USA) [21] **(v)** Celegans(N)(neural network of C. elegans) [21], **(vi)**Celegans(M) (metabolic network of C. elegans) [8], **(vii)** Dolphin (social network of dolphins) [13], **(viii)** Football (network of American football games) [11] and **(ix)** Delaunay (triangulations at random points in an unit square). The properties of the networks and their modularity values under different orderings are given in Table 1.

The orderings based on degree generally favor the earlier processing of high (HD) degree vertices. We see that HD produces higher (or equal) values of modularity as compared to LD. However, HD also has fewer components, indicating the possibility of preferential attachment. The BC and CC orderings try to influence well connected clusters of vertices to be processed consecutively. The results do not show any clear indication of which of these two orderings are better. When comparing the four orderings (HD,LD,BC and CC), LD, often produces the lowest modularity. There is no particular pattern to the behavior of RN, the random ordering. It sometimes gives the highest modularity and sometimes the lowest.

We note that the modularity of the Jazz and PolBooks networks have remained same across all the orderings, while the deviation of the modularity

Name	**Vertices**	**Edges**	**RN**	**HD**	**LD**	**BC**	**CC**
Jazz	198	2742	.4387 (4)	.4387 (4)	.4387 (4)	.4387 (4)	.4387 (4)
PolBooks	105	441	.5019 (4)	.5019 (4)	.5019 (4)	.5019 (4)	.5019 (4)
Chesapeake	39	170	.2489 (3)	**.2570** (3)	.2489 (3)	.2342 (3)	.2375 (3)
Power	4941	6594	**.9157(63)**	.9112(66)	.9140(65)	.9157(63)	9084(62)
Celegans(N)	297	2148	.4024 (11)	.4016 (8)	.4040 (9)	**.4124** (8)	.4019 (10)
Celegans(M)	453	2025	.3965 (11)	.3999 (10)	**.4040 (10)**	.3999 (10)	.3992 (10)
Dolphin	62	159	.4630 (5)	**.5028** (5)	.4685 (5)	.4766 (5)	.5006 (5)
Football	115	613	**.5682** (6)	.5510 (6)	.5497 (6)	**.5682** (6)	.5400 (6)
Delaunay	1024	3056	.7405 (5)	**.7466** (6)	.7168 (7)	.7160 (7)	.7238 (5)

Table 1. The maximum modularity value of a test-suite of networks under different orderings. Jazz and Polbooks have the same modularity across all orderings. The highest obtained value of modularity of the other networks are highlighted in bold. The number of communities from each ordering is given in parenthesis.

values of the Football, Dolphin and Delaunay networks are higher (shown in Table 2). This leads us to conjecture that Jazz and PolBooks have more stable groups of vertices, that are always combined together.

Since identical values of modularity do not necessarily mean identical partitions, we compared the community membership obtained by the different orderings with that obtained by RN (the Rand Index). Based on the results in Table 2 we see that Jazz and PolBooks indeed retain perfect correspondence across orderings, followed by Chesapeake, Celegans(N) and Power, then Dolphin, Football and Celegans(M), while Delaunay has the worst correspondence of all. It is also interesting to see that the relative values of the deviation in modularity correspond with the Rand Indices. Networks with higher rand indices have lower standard deviation of the modularity values. These results indicate that Jazz and Polbooks are likely to have many stable communities, followed by Chesapeake, Celegans(N) and Power. Dolphin, Football and Celegans(M) will likely have fewer stable communities and Delaunay the fewest stable communities among all networks.

While the empirical results are not conclusive evidence of stable communities, the values seem reasonable based on the application domain. For example; Jazz is a network of collaborations between East coast and West coast musicians, and it is likely that musicians are more likely to work together if they are in nearby geographic locations and PolBooks is a network of books on politics which can be quite polarizing. The fact that the Delaunay network had almost no stable structure also points to the accuracy of our hypothesis. Delaunay networks are formed of triangles used for partitioning a surface. The triangles will have no particular preference of attachment and almost every ordering can produce a new community structure.

Name	**Rand In dex**				Deviation of
	RN vs HD	**RN vs LD**	**RN vs BC**	**RN vs CC**	Modularity
Jazz	1	1	1	1	0
PolBooks	1	1	1	1	0
Chesapeake	.967	1	.938	.968	.0047
Power	.9740	.9751	.9712	.9685	.0032
Celegans (N)	.9546	.933	.924	.899	.0045
Celegans (M)	.7937	.8039	.7876	.7648	.0027
Dolphin	.894	.959	.917	.909	.0184
Football	.822	.92	1	.913	.0124
Delaunay	.344	.3454	.3451	.3451	.014

Table 2. Comparison of the partitions obtained through different orderings. Columns 2-5 give the Rand Index of different partitions as compared with the random ordering(RN). Column 6 gives the deviation in modularity values over five different orderings.

3.2 Approaches to Identifying Stable Communities

We now consider the problem of identifying stable communities in a network. A naive method would be to apply a variety of community detection algorithms on different permutations of the network, and analyze this collection of partitions to find stable groups of vertices. Apart from being computationally expensive, the effectiveness of this method is also dependent on the number and quality of the perturbations and the community detection methods.

We posit that stable communities are an inherent property of the network and our goal is to develop algorithms that can identify them without executing all the different community detection algorithms. In order to design algorithms for identifying stable communities, we focus on the fact the a good partition of communities will provide a high (if not maximum) modularity. Therefore, we consider the following characteristic of stable communities—*these are unique groups of vertices such that the modularity of the network will decrease if any vertex from a group is moved to another.* Although this property seems to hold for any network partition that gives the maximum modularity, the key idea is that these groups have to be unique. That is once the groups of vertices are formed, there will exist no other partition of the same set of vertices that will provide the highest modularity. In most cases, only a subset of vertices of the entire network will satisfy this property and our goal is to identify this subset and the groups that they form.

First Approach. We define *stable communities to have more connections within the group (internal) than outside (external).* Such communities are easy to find. We order the vertices in descending order of their clustering coefficients. For each vertex with a high clustering coefficient, we form a cluster of the vertex and its neighbors. If the cluster satisfies the given property, then it is a stable community.

However, this method is not sufficient to identify stable communities. This is because even though cumulatively, within the community, the total number of external connections may be less than the internal ones, some individual vertices in the community might have more outer connections. In this case there is a possibility that they might migrate to a different community from the original one to provide higher modularity.

Second Approach. We modify our definition as follows; *any subset in a stable community should have more internal connections than external.* However this definition leaves only certain cliques, with extremely low external connections to be identified as stable. This approach is therefore very restrictive and indeed, we did not find any such stable community in all our test networks.

Third Approach. In our third approach, we relax the definition of stable communities as follows; *any subset of vertices in a stable community should have more internal connections than connections to any one external community.* This definition takes into account the relative "pull" of the external communities. If a subset from the group is connected strongly to only one external community, then there is a possibility that that subset can migrate out of its original group. However if the connections are divided across many external communities, then it is likely that the subset will be more attracted to its original group.

One of the main challenges in designing an algorithm based on this definition is that without any preliminary division of the network, we cannot know which are the internal and the external connections. We therefore define a stable community Φ^k of strength k, such that the internal connections in Φ^k are greater than the external connections to neighbors that are at a maximum distance k from each other. The value of k represents the tightness or the maximum diameter of the external communities. Our heuristic for identifying Φ^k is as follows;

```
Pseudocode for Identifying Stable Communities
Input: Network $N$, Maximum distance between communities $k$
Output: Set of stable communities of strength $\Phi^k$
Order vertices in descending order of clustering coefficients
For all vertices $v_i$ in network $N$ that are not in any stable community
    Create a subset $S$ of $v_i$ and its neighbors.
    For each neighbor $n_j$
        Compute, $x_j$, the number of connections of $n_j$ within $S$
        Identify external neighbors of $n_j$ which are not in $S$
        Identify, $y_j$, the subset of external neighbors which are all within distance $k$ of each other.
    If $x_j > y_j$ for all $n_j$ then mark $S$ as a stable community
Merge stable communities if they have common vertices
```

If the average degree of a vertex is d, then a vertex v_i will have on average $O(d)$ neighbors. Each neighbor n_j will also have $O(d)$ neighbors. The complexity for identifying internal and external vertices is $O(d)$, the complexity for computing x_j is $O(d)$ and the complexity for computing y_j for all possible values of k is $O(d^3)$. Therefore the total complexity for identifying individual quasi-chordal vertices is $O(d) * (O(d) + O(d^3)) \approx O(d^4)$. A vertex can be present in at most

d quasi-stable sets. If there are V vertices then the total number of elements to be compared and merged will be at most $O(dV)$.

Additionally, computing the connections for each subset in the community is computationally prohibitive. If there are n vertices in the cluster, then we have to check whether the conditions satisfy for 2^n subsets. We resolve this issue by only considering sets of a vertex and its neighbors at a time, and merging stable communities if they have common vertices. In practice, the execution time can be significantly reduced by creating subsets for vertices with only high clustering coefficients and not including neighbors with very high degrees. Subsets of vertices that do not share links to any external community are not considered, many clusters are disqualified after a few initial tests and generally the number of vertices per cluster is quite small (average 4-6). Additionally, the values of the shortest paths between external vertices can be reused for many neighbors and setting k to small values, such as 2-4 is sufficient for identifying most of the stable communities.

4 Modularity Maximization Using Stable Communities

We propose an initial preprocessing step, where stable communities are identified and merged, and then a community detection method is applied to this preprocessed network. Our corrective modification is based on the structure of a group of vertices, rather than one node at a time, and also takes into account the propensity of the network to form communities. The number of stable communities obtained can be an indicator to the inherent modularity of the network. If the number of stable communities is low or zero, maybe the network is not as modular and community detection on it would only be of academic rather than practical value.

We implemented this method on the test-suite of networks. The results are given in Table 3. Though several stable communities were found and combined for Jazz, the modularity did not change, indicating that the Jazz network is indeed very stable. Combining stable communities improved the modularity values for Power, Dolphin, Celegans(N) and Celegans(M). Only one stable community was obtained for Chesapeake and its modularity improved slightly (the modularity for BC ordering decreased a little). Similarly only one stable community was identified for Football, but including the preprocessing stage did not change the modularity values. No stable community was found for Delaunay. This result points to the effectiveness of our method. As discussed earlier, Delaunay being a network of triangles should not have any inherent community structure. The number of communities in the networks were slightly smaller than those obtained from the original algorithm. Generally lower number of communities give higher modularity values. The most surprising results were from PolBooks. We expected that like Jazz, PolBooks is very stable and the modularity values would not be affected by the preprocessing. However, the preprocessing step significantly improved the modularity of this network and the number of communities also increased.

The values of the Rand index (as compared to the random ordering) are given in Table 4. The values of Jazz and Football did not change, but for other networks they were equal or degraded slightly. Rand index is measures not so much the quality of the partitioning as the stability of the communities. The standard deviation of the modularity values were also worse (higher) for all the other networks, except Power for which it slightly improved. We conjecture that this change in values was due to movement of other non stable communities, and is an effect of the latent reordering caused by combining the stable communities earlier. Nevertheless, we plan to investigate this phenomena further.

Table 5 gives the execution time of the original algorithm and and the new one based on combining stable communities. Due to the extra time required for identifying stable communities, the times of the modified method is generally slightly higher. However, in most cases, this increase is only a small percentage of the total execution time.

Name	**RN**	**HD**	**LD**	**BC**	**CC**
Jazz (no change)	.4387(4)	.4387(4)	.4387(4)	.4387 (4)	.4387(4)
PolBooks	.5029 (5)	.5207(5)	.5019(4)	**.5229**(5)	.5019(5)
Chesapeake	.2489(3)	**.2570**(4)	.2489(3)	.2341(3)	.2570(3)
Power	.9125(63)	.9131(65)	.9140(65)	**.9159** (60)	.9128 (63)
Celegans (N)	.4236 (9)	.4215 (10)	.4128(11)	.4242(11)	**.4266**(8)
Celegans (M)	.4227(9)	.4227(9)	.4150 (11)	.4351 (10)	.4131 (9)
Dolphin	.4823(5)	.5086(5)	.5042(5)	.4973(5)	**.5187**(5)
Football (no change)	**.5682** (6)	.5510 (6)	.5497 (6)	**.5682** (6)	.5400 (6)

Table 3. The maximum modularity value of test-suite of networks under different orderings, obtained by combining stable communities in the preprocessing stage. All the values (except for Cheaspeake-BC and Power-RN) are either equal to or higher than the values obtained by the base method. The highest obtained values of modularity are highlighted in bold. The numbers in the parenthesis give the number of communities.

Name	**Rand In dex**				Deviation of
	RN vs HD	**RN vs LD**	**RN vs BC**	**RN vs CC**	Modularity
Jazz (no change)	1	1	1	1	0
PolBooks	.9454	.843	.843	.897	.0108
Power	.973	.974	.974	.973	.0094
Chesapeake	.967	1	.938	.968	.0014
Celegans (N)	. 855	.869	.84	.83	.0053
Celegans (M)	.78	.76	. 80	.71	.0087
Dolphin	.827	.858	.851	.868	.0136
Football (no change)	.822	.92	1	.913	.0124

Table 4. Rand Index of different partitions, obtained by combining stable communities in the preprocessing stage, as compared with the random ordering.

Name	RN	HD	LD	BC	CC
Jazz	13.15(15.3)	13(18.06)	13.37(14.79)	13.33(14.86)	13.07(15.78)
PolBooks	.693 (.864)	.657 (.837)	.726 (.835)	.701(.838)	.708(.855)
Power	109.49 (148.05)	108.78 (168.16)	114.72 (148.75)	105.10(152.32)	113.02 (142.20)
Chesapeake	.052(.082)	.052 (.095)	.051(.075)	.049(.087)	.052 (.084)
Celegans (N)	71.07(73.44)	71.8(77.8)	72.3(75.52)	76.71(79.68)	73.11(77.38)
Celegans(M)	22.26 (31.90)	21.20 (30.99)	28. 36 (26.91)	24.26 (22.13)	21.50 (29.33)
Dolphin	.066(.085)	.063 (.091)	.062 (.086)	.067 (.084)	.065 (.093)
Football	.052(.91)	.05(.90)	.05(.94)	.049(.91)	.05 (1.00)

Table 5. The execution time in seconds of the original and the proposed method (given in parenthesis).

5 Conclusions and Future Work

In this paper we proposed a preprocessing step for agglomerative community detection, based on finding stable communities. Since stable communities are groups of vertices that naturally form a community, this preprocessing step will veer the agglomeration towards a more ”correct” solution. Additionally the number of stable communities indicates whether the network has any intrinsic community structure at all.

Our stable community detection heuristic can effectively find communities for many of the benchmarked networks, and is also effective in the complementary case, that it does not find stable communities where none are expected. However the heuristic has still room for improvement, particularly in correctly identifying external communities, and in improving the execution time. The current implementation is cost efficient for only networks of 5000 vertices or lower. One of our future research plans is to develop a better version of stable community detection algorithm.

We also observe that the preprocessing step helps in increasing the modularity in most cases. We had anticipated that the Rand index, which denotes the similarity between different partitions to also improve, but that is not the case, and we are investigating other techniques to amortize the effect of vertex ordering on community detection while achieving high modularity.

References

1. Barabasi, A.L., Jeong, H., Ravasz, E., Neda, Z., Schuberts, A., Vicsek, T. Evolution of the social network of scientific collaborations. *Physica. A.* 311, 590614 (2002)
2. D. Baird, R.E. Ulanowicz.The seasonal dynamics of the Chesapeake Bay ecosystem. *Ecol. Monogr.* 59: 329-364.(1989).
3. V.D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre. Fast unfolding of community hierarchies in large networks. *J. Stat. Mech.* 2008 (10):

4. Boguna, M., Pastor-Satorras, R., Vespignani: Epidemic spreading in complex networks with degree correlations. *Statistical Mechanics of Complex Networks.* Lecture Notes in Physics, vol. 625, pp. 127147 (2003)
5. U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172188, 2008.
6. Clauset, A., Newman, M.E.J. and Moore, C. Finding community structure in very large networks. *Phys. Rev. E.* 70(6), 66111 (2004)
7. Dimacs Testbed http://www.cc.gatech.edu/dimacs10/downloads.shtml
8. J. Duch and A. Arenas, Community Identification using Extremal Optimization. *Physical Review E*, 72, 027104, (2005).
9. B. H. Good, Y.-A. de Montjoye and A. Clauset The performance of modularity maximization in practical contexts. *Phys.ical Review. E*, 81, 046106 (2010).
10. P.Gleiser and L. Danon , *Adv. Complex Syst.* 6, 565 (2003).
11. M. Girvan and M. E. J. Newman, *Proc. Natl. Acad. Sci.* USA 99, 7821-7826 (2002).
12. Karrer, B., Levina, E. and Newman, M.E.J. Robustness of community structure in networks *Physical Review E*, Vol. 77, No. 4. (2008)
13. D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, *Behavioral Ecology and Sociobiology* 54, 396-405 (2003).
14. M.E.J. Newman, M. Girvan. Finding and evaluating community structure in networks.Phys. Rev. E 69(2), 026113 (2004)
15. M.E.J. Newman, *Network. An Introduction.* Oxford University Press, USA; 1 edition (May 20, 2010)
16. Political Books. http://www.orgnet.com/
17. M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. Notices of the *American Mathematical Society.* 56, (2009).
18. W. Rand, Objective criteria for the evaluation of clustering methods. *J. Am. Stat. Assoc.* 66 (336), 846850 (1971).
19. J. Riedy, D. A. Bader, K. Jiang, P. Pande, R. Sharma. Detecting Communities from Given Seeds in Social Networks. Technical Report. http://hdl.handle.net/1853/36980
20. Voevodski, K., Teng, S.H., Xia, Y.: Finding local communities in protein networks. *BMC Bioinformatics* 10(10), 297 (2009)
21. D. J. Watts and S. H. Strogatz, *Nature* 393, 440-442 (1998).