High quality multi-core multi-level algorithm for community detection

Suely Oliveira and Rahil Sharma*

Department of Computer Science,
University of Iowa,
Iowa, IA-52246, USA
Fax: +1319-335-0731
Email: suely-oliveira@uiowa.edu
Email: rahil-sharma@uiowa.edu
*Corresponding author

Abstract: One of the most relevant and widely studied structural properties of networks is their community structure or clustering. Detecting communities is of great importance in various disciplines where systems are often represented as graphs. Different community detection algorithms have been introduced in the past few years, which look at the problem from different perspectives. Most of these algorithms, however, have expensive computational time that makes them impractical to use for large graphs found in the real world. Maintaining a good balance between the computational time and the quality of the communities discovered is a well-known open problem in this area. In this paper, we propose a *multi-core multi-level* (MCML) community detection algorithm based on the topology of the graph, which contributes towards solving the above problem. MCML algorithm on two benchmark datasets results in detection of accurate communities. We detect high modularity communities by applying MCML on *Facebook Forum* dataset to find users with similar interests and *Amazon product* dataset. We also show the scalability of MCML on these large datasets with 16 Xeon Phi cores.

Keywords: parallel algorithm; multi-level; multi-core; community detection.

Reference to this paper should be made as follows: Oliveira, S. and Sharma, R. (2017) 'High quality multi-core multi-level algorithm for community detection', *Int. J. Computational Science and Engineering*, Vol. 15, Nos. 3/4, pp.311–321.

Biographical notes: Suely Oliveira is a Professor in the Computer Science Department at the University of Iowa. She received her PhD from the University of Colorado in 1993. She was at the Australian National University from 1993 to 1994, and at Texas A&M University from August 1994 to May 1998. She has published over 50 articles in the areas of numerical analysis, parallel algorithms, scientific computing, combinatorial scientific computing, and is co-author of the book *Writing Scientific Software* published by Cambridge Press University and *Building Proofs: A Practical Guide* published by World Scientific.

Rahil Sharma received his BE in Computer Engineering from the University of Mumbai and MS in Computer Science from the University of Iowa, USA. He is currently working towards his PhD in Computer Science at the University of Iowa. His research interests include areas of algorithm development, high performance parallel computing, and big data analytics.

1 Introduction

Most complex systems like World Wide Web, social networks (Facebook's user interaction network), biological interaction networks [protein-protein interaction (PPI) networks, chemical bonding networks], paper citation networks, etc. can be represented as graphs. Of all the tools used to analyse networks, the most relevant and widely studied tool is community detection or clustering. A community in a network is a set of nodes which are densely connected with each other and sparsely connected to the other nodes in the network. The interpretation of communities in these graphs varies based on its applications. For example, in biological networks like PPI networks, proteins with similar functional modules will lie in the same community. This allows identification and prediction of functional modules in PPI networks (Lee et al., 2013; Oliveira and Seok, 2008; Wang and Qian, 2013). We have successfully applied community detection algorithm to identify and predict functional modules in PPI networks (Oliveira and Sharma, 2015). Similarly, in a social network like Facebook, users with a common interest or acquaintance will lie in the same community.

Community detection in a network also extracts the structural properties of the network (Girvan and Newman, 2002) and the various interactions in the network (Barabasi and Oltvai, 2004). There is no universally accepted

Copyright © 2017 Inderscience Enterprises Ltd.

definition for community detection. Hence, most of the recent work in this area does not have a community structure defined in its literature, but has a quality function defined to quantify how well the network is divided into communities. So the community detection problem focuses on optimising this quality function (Newman, 2004). One of the quality functions often used is *modularity* (Newman and Girvan, 2004). Most of these algorithms are computationally very expensive and hence impractical for use on large networks. Tackling large volumes of graph-structured data requires parallel multi-core directives to achieve scalable algorithms. We present a multi-core multi-level (MCML) community detection algorithm which achieves a good balance between scalability and quality of the communities detected. We summarise our main contribution in this paper as follows:

1 We propose an MCML community detection algorithm which achieves a good balance between scalability and quality of the communities detected, compared to other algorithms in the current state-of-the-art.

2 We show that the quality of the results obtained by the MCML algorithm for benchmark datasets with ground truth is highly accurate.

3 We show that, applying MCML to datasets without ground truth, detects communities roughly as meaningful as other well known algorithms in the current state-of-the-art (Pons and Latapy, 2005; Rosvall and Bergstrom, 2008; Yang and Leskovec, 2013; Prat-Pérez et al., 2014; Blondel et al., 2008; Newman and Girvan, 2004), etc., and in some cases even better (Facebook Forum). The comparison is done using modularity as the metric.

The MCML algorithm has the parameters *strength*, *maximum size*, and *minimum size* which enable us to extract communities of desired strength (i.e., strong interactions and desired size). The parameter strength varies from 0 to 1. If we are required to find the strongest and the most critical interaction in the given network, we can do so by setting the value of the parameter strength to 1.

The remainder of this paper is organised as follows: In Section 2, we describe related work. In Section 3, we describe various stages of the MCML algorithm along with its parallel implementation. In Section 4, we describe the computational results of applying MCML algorithm on two small benchmark datasets (i.e., karate club and dolphin club), followed by large datasets like Facebook Forum and Amazon product network. We end the paper by giving implementation details in Section 5 and conclusion in Section 6. Also note that, the words 'network' and 'graph' are used inter-changeably throughout the paper.

2 Related work

Community detection is an interesting problem in the domain of *graph partitioning*. Interest in community detection problem started with the new *partitioning* approach by Girvan and Newman (2002) and Newman and Girvan (2004); where the edges in the network with the maximum betweenness are removed iteratively, thus splitting the network hierarchically into communities. Similar algorithms were proposed later on, where attributes like 'local quantity', i.e., number of loops of a fixed length containing the given edge (Radicchi et al., 2004) and a complex notion of 'information centrality' (Fortunato et al., 2004), is used to decide removal of edges. *Hierarchical clustering* is another major technique used for community detection, where based on the similarity between the nodes, an agglomerative technique iteratively groups vertices into communities. There are different existing methods to choose the communities to be merged at each iteration. Newman (2004) and Vieira et al. (2014) developed an algorithm which starts with all the nodes as individual community and iteratively merge them to optimise the 'modularity' function. Many other algorithms in the literature of community detection, like one proposed by De Meo et al. (2011) and Hashimoto et al. (2012) rely heavily on *modularity maximisation*. *Label propagation* is another well known technique used for community detection, which finds communities by iteratively spreading labels across the network. Raghavan et al. (2007) proposed an algorithm, where each node picks the label in its 1-neighbourhood that has the maximum frequency. These labels are permitted to spread synchronously and asynchronously across the network until near stability are attained in the network. This method has some limitations, where in large communities dominate the smaller one's in the network, this phenomenon is called 'epidemic spread'. This limitation was resolved by Soman and Narang (2011) and Oliveira and Sharma (2015). Liu et al. (2013) used affinity propagation, which is similar approach to label propagation, for finding communities/clusters in images. In this paper, the algorithm we propose uses label propagation ideas and also prevents 'epidemic spread' in the network, thus avoiding extremely large communities which dominates the entire network. Some community detection algorithms use *random walks* as a tool. The idea is that, due to the higher density of internal edges, the probability of staying inside the community is greater than going outside. This approach is used in Walktrap (Pons and Latapy, 2005) and Infomap (Rosvall and Bergstrom, 2008) algorithms. A thorough review on community detection algorithms for networks is given in (Fortunato, 2010).

Community detection algorithms are a well studied research area, but achieving strong scalability along with detecting high quality communities is sill an open problem. One of the recent parallel algorithms developed to detect disjoint community structures based on maximising weighted network partitioning is given in Pons and Latapy (2005). Recently, Soman and Narang (2011) proposed a scalable parallel algorithm for community detection, based on label propagation, which is optimised for GPGPU architectures. This algorithm just works on local information which drives the high scalability of this algorithm. Prat-Pérez et al. (2014) proposed a scalable

community detection algorithm, which partitions the graph by maximising the weighted community clustering (WCC), a recently proposed community detection metric based on triangle analysis (Prat-Pérez et al., 2012). Some other works which focused on developing parallel implementation for existing community detection heuristics is given in Rytsareva et al. (2014). In this paper, we propose a shared memory-based community detection algorithm, which achieves a good balance between *scalability* and *quality* of the communities discovered.

3 MCML algorithm

In this section, we present the MCML algorithm involving a preprocessing stage, where each edge is assigned a strength based on the topology of the graph. Then based on the strength requirement of the communities, weak edges are removed and coarser graph instances are recursively created by identifying and removing communities, using the node with highest centrality each time. We recursively apply this step until every node is assigned to a community.

3.1 Preprocessing: edge strength assignment

The MCML algorithm finds communities in a graph $G(V, E)$ where V represents the nodes/vertices and E represents the edges between the nodes, by assigning strength to the edges initially. It is desirable to assign an edge strength value that most accurately represents the topological structure of the graph in the MCML algorithm. Since we do not have any prior knowledge of the community structure, we assign a strength value to each edge based on the significance of that edge to the other nodes in the graph, and to the nodes at the end points of that edge. For each edge $e(i, j)$ (where i and j are nodes) in the fine graph G, the topological edge strength value $\alpha(i, j)$ assigned to it is the ratio of number of triangles that edge $e(i, j)$ participates in to the total number of triangles containing node i.

If the strength value of an edge $e(i, j)$ is greater than other edges in the 1-neighbourhood of i then, node i and node j are more likely to be in the same community. Whereas on the contrary, if edge $e(i, j)$ has lower strength value than most other edges in the 1-neighbourhood of i, then node i and node j are less likely to be in the same community. Mathematically,

$$\alpha(i, j) = \frac{t_{(i,j)}}{\sum_{(i,k)} \left(t_{(i,k)} \right)}; \quad k \in N_i \tag{1}$$

where N_i is the 1-neighbourhood of i, and $t_{(i,j)}$ is the total number of triangles whose sides contain edge (i, j).

The MCML algorithm also works well with weighted graphs, where the edges are assigned weights w_{input} as an input. To get the total weight of an edge, we simply have to take product of the topological edge strength value, with its input weight.

$$\alpha_{total}(i, j) = \alpha(i, j) \times w_{input}(i, j) \tag{2}$$

After this we normalise the edge strength for all the edges, such that they range in between 0 and 1.

3.2 Remove weak edges

The procedure of removing the weak edges from the fine graph G is based on the required strength β of the communities. Edges with $\alpha(i, j) < \beta$ are removed. This simply means that, for low values of β, fewer edges will be removed from the fine graph G, as compared to when β has higher values. After deleting these edges we might label some nodes as *non-community nodes*, i.e., the degree of these nodes is zero. For higher values of β we get a higher number of non-community nodes and more stronger, smaller and significant communities are extracted. Whereas for lower values of β we get smaller number of non-community nodes, and higher number of nodes are assigned a community.

3.3 Multilevel coarsening

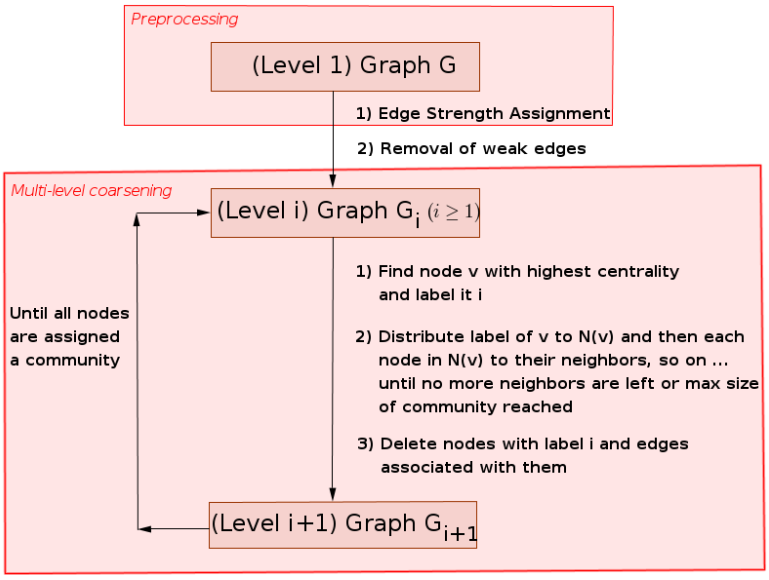
Let G_i $(i \geq 1)$, be the graph obtained by removing weak edges (i.e., $\alpha(j, k) < \beta$) from G. We apply the following coarsening step recursively to extract meaningful community structure.

Multilevel coarsening: We select a node v from G_i, having highest centrality and label it i. Now we distribute this label i assigned to v, to all the neighbours of v, denoted by $N(v)$. We continue distributing labels to the neighbours of all the nodes with label i. We do this until there no more neighbours are left to send the label, or the maximum required size of the community is reached. Then we obtain a coarse graph G_{i+1} by removing all the nodes with label i from G_i, along with their associated edges. We continue this process recursively until all the nodes are assigned a community. The communities having number of nodes less than the *minimum number* required in a community, labels all the nodes in that community as non-community nodes. This idea of recursively deleting communities is along the same lines as the one used in Jancura et al. (2012). The general schema for the algorithm is shown in Figure 1.

3.4 Parallel implementation

Parallel shared-memory-based, multi-core implementation, for each stage of our MCML algorithm is described in this section.

Figure 1 MCML algorithm's general schema (see online version for colours)



3.4.1 *Parallel preprocessing*

In the preprocessing stage, we designate a master thread, which divides the graph roughly into k equal parts, where k is the number of cores/threads available. We perform a k partition on the input graph. We can use an existing k way graph partitioning library like KaHIP, METIS, PMETIS, etc. to divide the graph into k parts. The master thread then assigns each of these k parts to k threads individually (including itself), as shown in Figure 2. Then each thread computes the edge strengths in the part of the graph assigned to them, using equation (1). The inter-partition edges, which are the dashed edges in Figure 2, are excluded in this computation. Once all the threads have completed their edge strength assignment computations, the master thread merges the k parts of the graph back together and computes the edge strengths of the previously excluded dashed edges.

3.4.2 *Parallel weak edge removal*

In weak edge removal stage of the algorithm, we again partition the preprocessed graph into k parts and assign each part to each of the k cores/threads individually, in the similar way we did in the preprocessing stage. Each core removes the edges having strength less than threshold value (β) from the corresponding part of the graph they process. Note that, here we do not have to worry about the inter-partition edges (dashed edges in Figure 2) because, if they have strength less than the threshold they will be removed, else will be restored, by both the threads they are assigned to. Once all the threads have completed their weak edge removal process, the master thread merges all k parts of the graph back together.

3.4.3 *Parallel multilevel coarsening*

In the multilevel coarsening stage of the algorithm, we designate a master thread, which first finds the node with highest degree centrality in the graph and labels it i ($i \geq 1$). We then create a global queue, such that all the k cores point to the rear-end of this queue, as shown in Figure 3. The highest centrality node is then pushed into this global queue. The master thread is then assigned to this node, based on our construction of the queue. Then the master thread distributes label i to 1-neighbourhood of this node and also add the new nodes it discovers in the 1-neighbourhood to the queue. Similarly in the consequent rounds, the threads are assigned nodes from the queue as shown in Figure 3 and each thread distributes label i to a 1-neighbourhood (nodes that have not yet received the label i) of the node assigned to it, along with adding the newly discovered nodes to the queue. So at a given time, there are k nodes that are assigned to k cores in a cyclic fashion, which can simultaneously propagate its labels. In Figure 3, initially the master thread, i.e., core 1 finds node 1 (red node) which is the highest centrality node and adds it to the queue after assigning label i to it. Node 1 is then assigned to core 1, which distributes label i of node 1 to the nodes in its 1-neighbourhood (green nodes) which have not yet been labelled. Along with label distribution, it also add these nodes to the queue. Nodes 2, 3, 4, 5, 6, 7, and 8 are added to the queue. In the second round, nodes 2, 3, 4, and 5 are assigned to cores 1, 2, 3, and 4, respectively in a cyclic fashion from the rear-end of the queue. Each of the cores then follow the same steps that was followed by core 1 in the initial round.

Figure 2 Parallel preprocessing (see online version for colours)

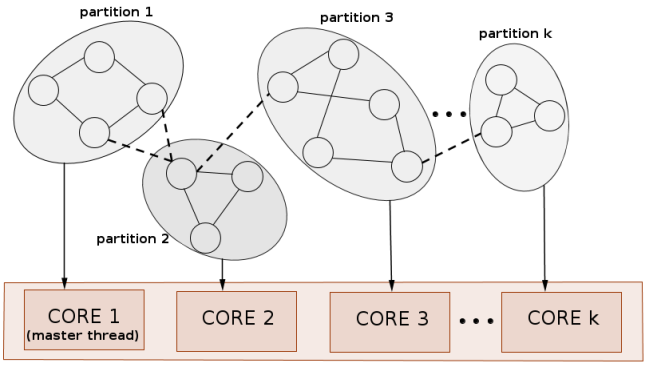
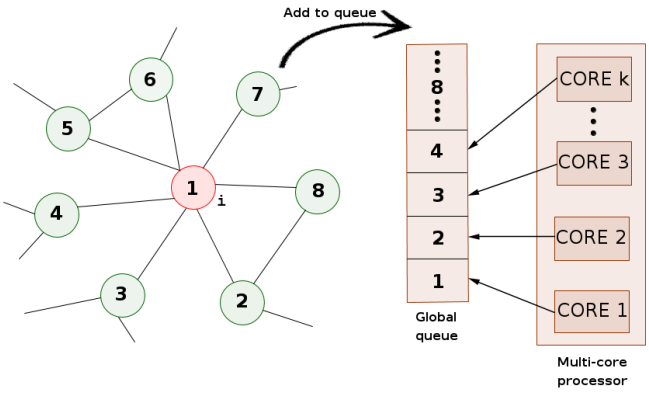


Figure 3 Parallel multilevel coarsening (see online version for colours)



We place appropriate *barriers* and *write locks* to the queue in order avoid *race conditions* between threads. This process continues in cyclic fashion, until the queue is empty (disjoint component found) or maximum desired size of the community is reached. After finding a community, the k threads remove the community with label i from the graph (using a trivial *parallel for loop*) and the same process is iteratively applied on the remaining graph with label $i + 1$. We continue this until the algorithm terminates, i.e., all nodes are assigned to a community. Note that we did not perform a graph partition in this stage to avoid nodes of the same community to be assigned to multiple threads. The pseudo code for the MCML algorithm is given in Algorithm 1.

Algorithm 1 MCML algorithm

Require: Graph $G(V, E)$, β, *max size*, *min size*

return Community of each node
for each thread T **do**
  Assign k blocks of nodes and edges to each thread T
  for each Edge $e(i, j)$ assigned to thread T **do**
    Find strength $\alpha(i, j)$ using equation (1)
  end for
  for each Edge $e(i, j)$ assigned to thread T **do**
    if $(\alpha(i, j) < \beta)$ **then**
      Delete $e(i, j)$
    end if
  end for
  while (All nodes are assigned to a community) **do**
    for each Node assigned to thread T **do**
      Find node v with highest centrality, label it i
    end for
    while (No neighbours left or max sized reached) **do**
      Assign nodes and edges to each thread T
      Distribute label to the neighbours of the nodes, with label i
    end while
    for each Node assigned to thread T **do**
      Delete node with label i and associated edges
    end for
  end while
end for
return community label for each node

4 Computational results

4.1 Benchmark datasets

We use the benchmark datasets, karate club (Zachary, 1977) and dolphin club (Lusseau et al., 2003) to determine the *quality* of the results obtained by applying MCML algorithm. Since these two datasets have ground truth communities, we measure the quality of the results based on the *accuracy* metric, i.e., the number of nodes correctly assigned by MCML algorithm to the community they actually belong to in real life. We run MCML algorithm for various values of β (0, 0.1, 0.4, 0.6, 1.0). We also have a plot showing the number of nodes marked as non-community nodes, versus different values of β. Comparison of various community detection algorithms on Karate and Dolphin club benchmark datasets is shown in Table 1. The comparisons are made on the basis of the number of communities detected, number of correct matches and incorrect matches of the community nodes with its ground truth communities.

4.1.1 Karate club

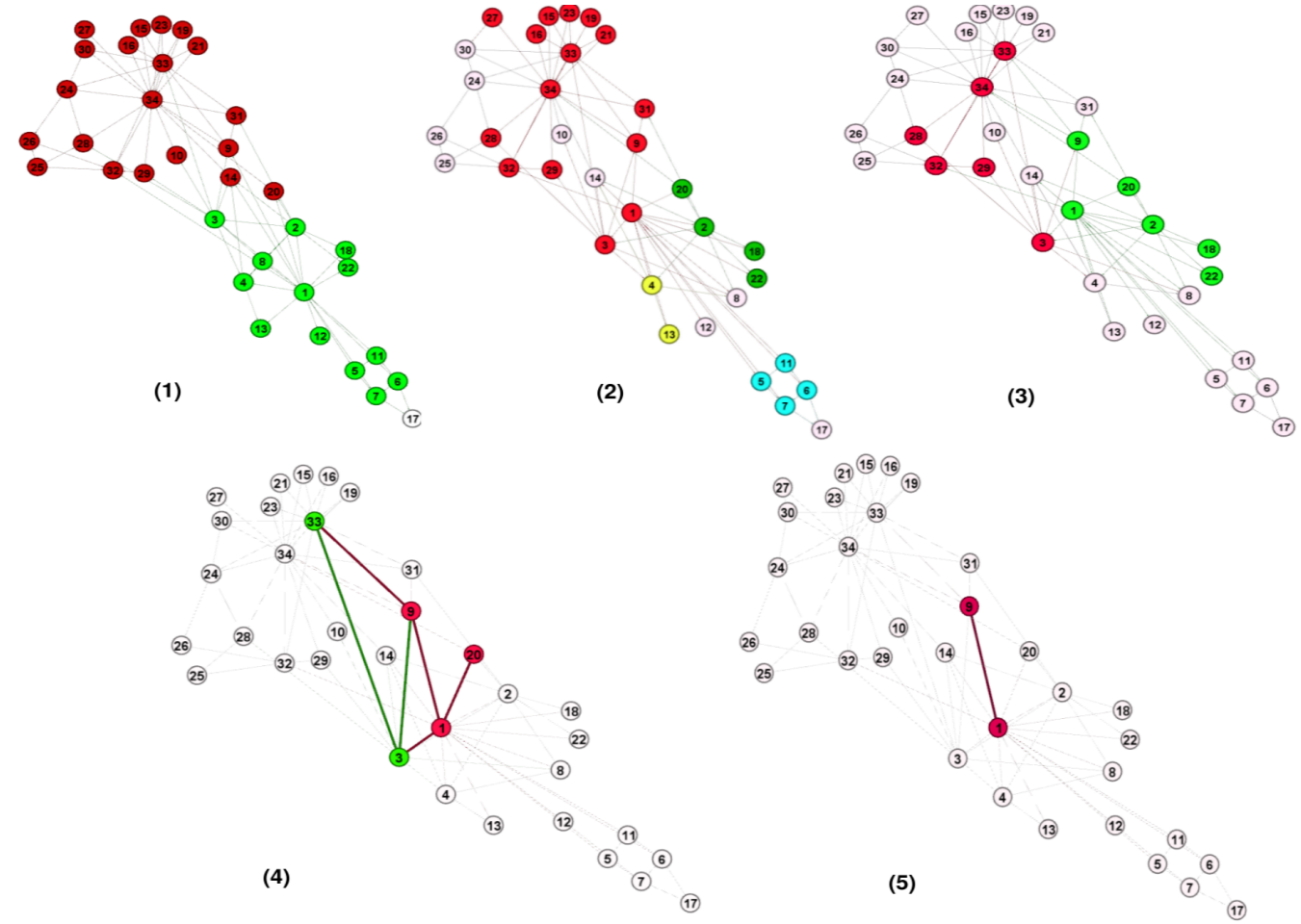
This is a social network of friendships between 34 members of a karate club. It contains 156 edges and the dataset is unweighted and undirected. The real life known partition of this graph is into two groups. The group breaks down into 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 17, 18, 22 and 9, 10, 14, 15, 16, 19, 20, 21, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34. In Figure 4(1), where $\beta = 0.0$ and maximum size = 34, when compared to the ground truth communities of this network, all the nodes are correctly grouped, except node number 17, which becomes the non-community node. All the nodes in white are non-community nodes. The nodes with different colours belong to different communities. The accuracy plot for groupings made by MCML algorithm is shown in Figure 5. In Figure 4(5), where $\beta = 1.0$, we extract the

strongest link in the club, which is between node 1 and 9. In Figure 7(a), we can see that, as the value of β increases the number of non-community nodes increases and the more stronger and smaller communities are extracted. The edges extracted represent stronger connections compared to edges associated with non-community nodes.

4.1.2 Dolphin club

This is an undirected social network of frequent associations between 62 dolphins (nodes) in a community living off Doubtful Sound, New Zealand. Dolphin club is an unweighted network containing 159 edges. In Figure 6(1), where $\beta = 0.0$ and maximum size = 62, when compared to the ground truth communities of this network, 56 nodes are correctly grouped, three nodes become non-community nodes and three nodes are incorrectly grouped. These three incorrectly grouped nodes form a new community, since they have much stronger connection amongst each other, than with community with the red label. All the nodes in white are non-community nodes. Accuracy plot is shown in Figure 5. In Figure 6(5), where $\beta = 1.0$ and we extract the strongest link in the club, which is between node 51 and 46. In Figure 7(b), we can see that as the value of β increases the number of non-community nodes increases. On the basis of the above experiments on the two benchmark datasets, we can conclude that the higher the value of β the stronger and smaller is the extracted community.

Figure 4 Karate club, (1) $\beta = 0.0$ (2) $\beta = 0.1$ (3) $\beta = 0.4$ (4) $\beta = 0.6$ (5) $\beta = 1.0$ (see online version for colours)



Note: Edges retrieved/highlighted in (4) and (5) have stronger connections than other edges in the graph.

Table 1 Comparing various community detection algorithms for karate and dolphin club benchmark datasets with $\beta = 0.0$

Algorithm	*Karate club (34 nodes)*			*Dolphin club (62 nodes)*		
	No. of communities	*Correct*	*Incorrect*	*No. of communities*	*Correct*	*Incorrect*
Jancura et al (2012)	3	24	10	3	46	16
Raghavan et al. (2007)	2	27	7	4	45	17
Soman and Narang (2011)	2	29	5	2	49	13
MCML	*2*	*33*	*1*	*3*	*56*	*6*

Figure 5 Accuracy plot: karate club and dolphin club (see online version for colours)

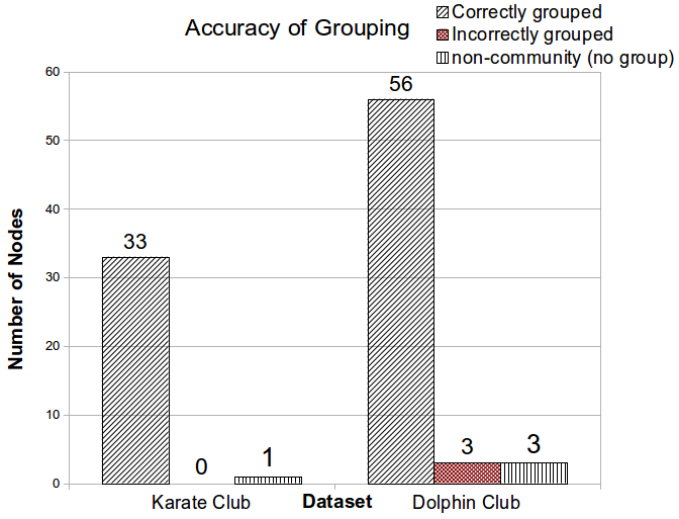
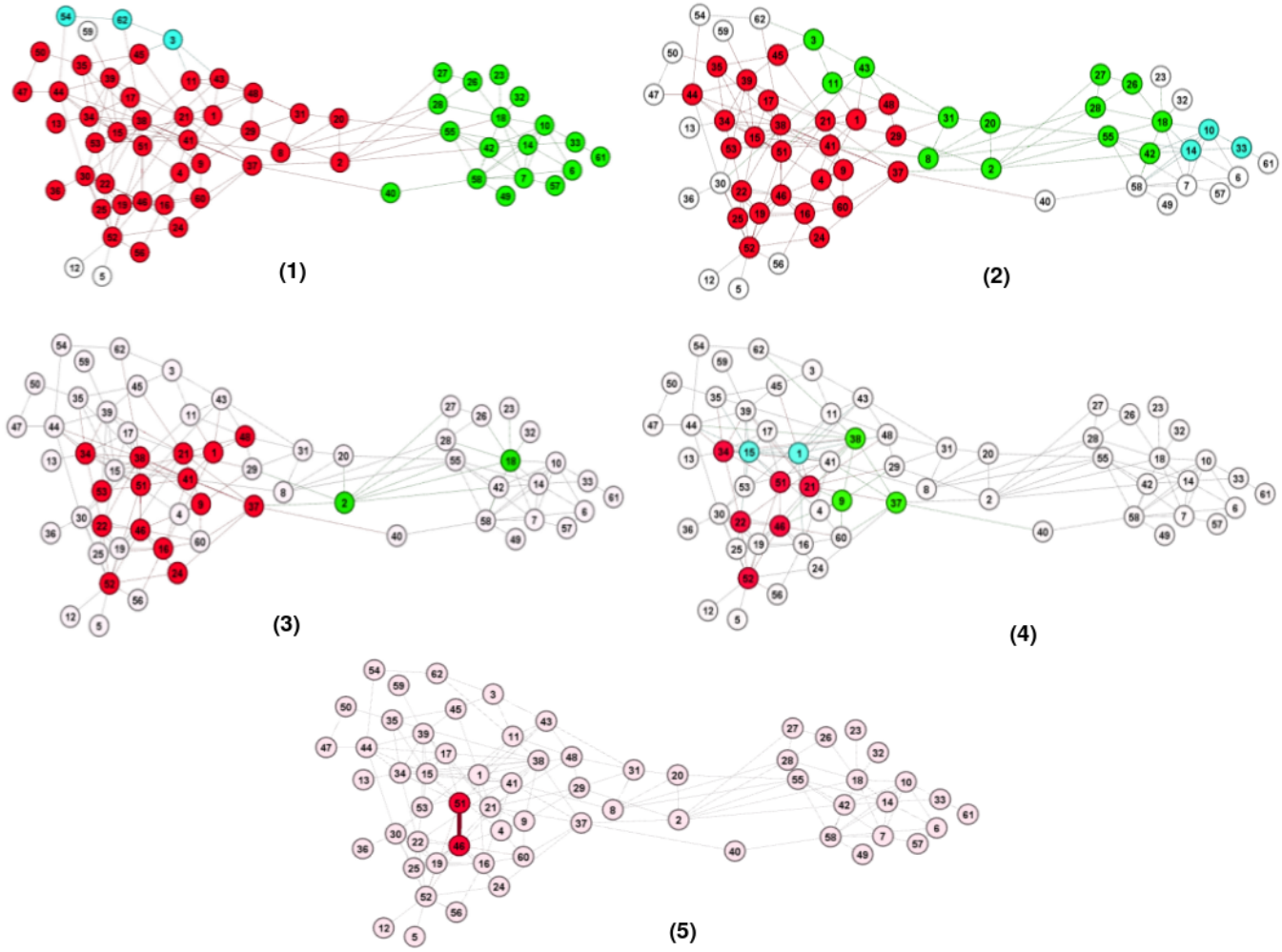
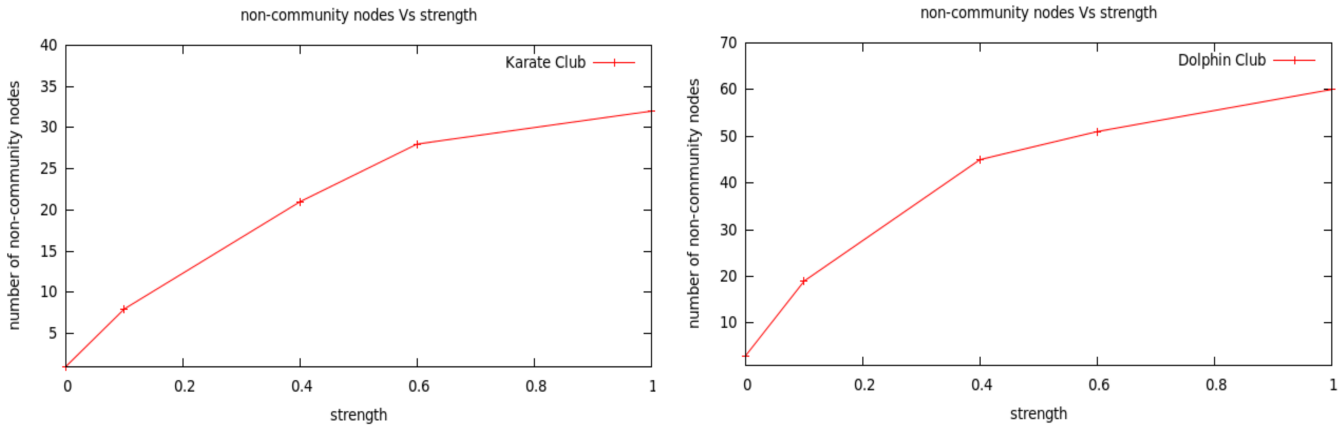


Figure 6 Dolphin club, (1) $\beta = 0.0$ (2) $\beta = 0.1$ (3) $\beta = 0.4$ (4) $\beta = 0.6$ (5) $\beta = 1.0$ (see online version for colours)



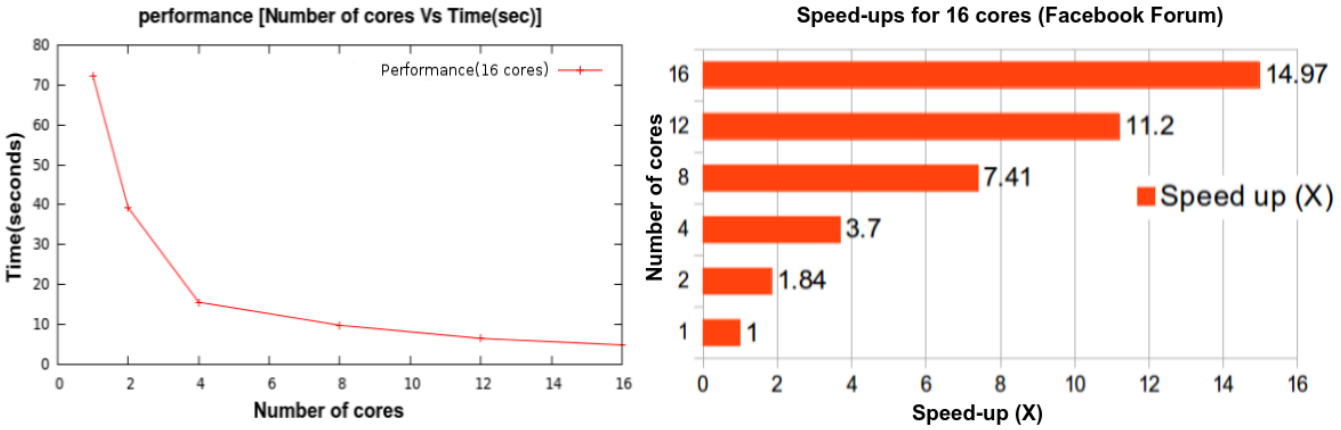
Note: Edge retrieved/highlighted in (5) has the strongest connection than any other edge in the graph.

Figure 7 Non-community nodes vs. strength, (a) karate club (b) dolphin club (see online version for colours)



(a) (b)

Figure 8 Facebook Forum, (a) running time vs. number of cores (b) speed up (see online version for colours)



(a) (b)

4.2 *Facebook Forum dataset*

This dataset is obtained from Facebook online social network. The main focus in this network is on users' activity in the forum. The forum represents a 2-mode network between primary nodes which are 899 *users* and secondary nodes which are 522 *topics* in the forum. It is a weighted network where the weights represent the number of messages a user posted on a particular topic. We use the preprocessed version of this dataset for our experiments, where 2-mode network is transformed into a 1-mode network maintaining the primary nodes, which are users and contain 142,761 edges. This dataset is available from http://toreopsahl.com/datasets/#online_social_network.

Figure 9(a) represents the format of original dataset. User A posts three messages on topics 1 and four messages on topic 2. User B posts one message on topics 1 and 5 messages on topic 2. When this dataset is preprocessed to eliminate the secondary nodes, we have two directional links between A and B, i.e., A to B which has weight 7 and B to A having weight 6, as shown in Figure 9(b). The 1-mode projection of a weighted 2-mode network is based on the weights the two nodes have, directed towards common nodes. The two nodes interact with the common node, and Figure 9 shows how to project it onto a directed weighted 1-mode network. This dataset does not have ground truth communities, so we use modularity to determine the quality of the communities found. The quality comparison based on *modularity* and computational time for Facebook Forum dataset is given in Table 2. The modularity achieved for communities detected by MCML ($\beta = 0.15$) is 0.3566, which is the best so far in the present state-of-the-art. So even though we do not achieve the best running times as compared to other well known algorithms like (Blondel et al., 2008), we manage to maintain a good balance between the quality of the results and running times.

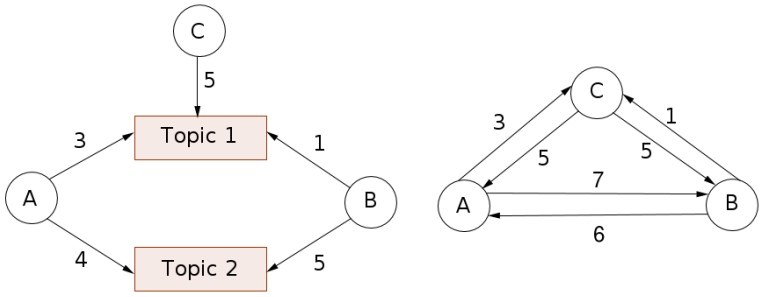
Finding communities in this interesting dataset implies, finding groups of users sharing similar interests. This information can be used by social networking sites to provide friend suggestions to users, or suggestions to join a particular community forum having common interests. The performance of MCML algorithm, on this dataset is shown in Figure 8, where we achieve speed-ups up to 14.97 times using 16 cores. This is close to a k fold improvement using k core processor where ($k \geq 1$).

Table 2 Comparing various community detection algorithms for Facebook Forum and Amazon datasets based on modularity and computational time using 16 cores

Algorithm	*Facebook Forum*		*Amazon*	
	Modularity	*Time (sec)*	*Modularity*	*Time (sec)*
Newman and Girvan (2004)	0.0488	-	-	-
Pons and Latapy (2005)	0.2031	-	0.451	-
Rosvall and Bergstrom (2008)	0.1372	-	0.470	-
Raghavan et al. (2007)	0.1733	47	0.210	> 10,000
Blondel et al. (2008)	0.3458	3	-	-
Yang and Leskovec (2013)	-	-	0.125	1890
Prat-Pérez et al. (2014)	-	-	0.295	15
Wang and Qian (2013)	-	-	0.510	4800
MCML	*0.3566*	*4.83*	*0.494*	*2389*

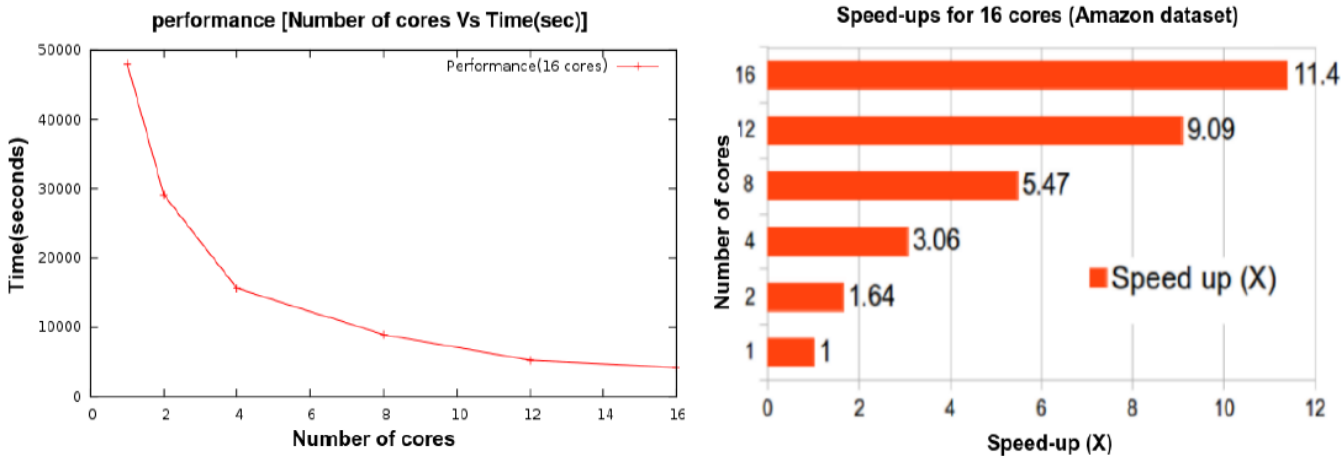
Notes: The blank values are not available in the literature of this research area. To get the computational time we include all the three stages of the algorithm.

Figure 9 (a) 2-mode weighted network (b) preprocessed 1-mode weighted network (see online version for colours)



(a) (b)

Figure 10 Amazon, (a) running time vs. number of cores (b) speed up (see online version for colours)



(a) (b)

4.3 Amazon dataset

This dataset represents a graph of products, where each node is a product and there is an edge between two products if they have been co-purchased frequently. This dataset has 334,863 nodes and ≈ 1 million edges. This dataset has a 151,037 ground truth communities, in which the top 5,000 communities are the most significant. We use this dataset (http://snap.stanford.edu/data/index.html) in our experiments, to show that MCML algorithm gives fairly good performance and speed-up when applied to this dataset, and also, we do not degrade the quality of the results while achieving this.

For $\beta = 0.1$ and *maximumsize* = 80 which is the largest community in the ground truth community data, it takes ≈ 7.34 hrs to extract communities in this dataset using one

core and ≈ 39.82 minutes using 16 cores (i.e., speed-up of 11.4 times). In Figure 10, we show the time taken to find communities in this dataset for 1, 2, 4, 8, 16 cores and corresponding speed-ups respectively. The quality comparison based on modularity and computational time for Amazon dataset is given in Table 2. The modularity achieved for communities detected by MCML ($\beta = 0.1$) is 0:494, which is better than most of the other algorithms in the present state of art. So even though we do not achieve exceptional running times as compared to other well-known algorithms like Prat-Pérez et al. (2014) and Yang and Leskovec (2013), we manage to maintain a good balance between the quality of the results and running times.

5 Implementation details

We implemented the MCML algorithm using C++ and graph boost libraries. The simulations for the benchmark datasets and the Facebook Forum dataset are done on Processor-Intel Core i73770, 3.4 GHz and Turbo Boost enabled Memory-16GB DDR3 – 1,600 RAM; Linux machines. These machines have four cores with hyper-threading enabled. The simulations for the Amazon dataset is done on a system running on CentOS 6.3, a Linux operating system based on Red Hat Linux, with 512GB Nodes, 32 GB RAM, 2.9 GHz, and 16 Xeon Phi cores. All the results obtained are averages of five runs. We use OpenMP directives for implementing parallel MCML algorithm. All the plots are done using Gephi and Gnuplot.

6 Conclusions

In this paper, we develop a MCML community detection algorithm, which achieves a good balance between scalability and quality of the communities discovered. We show that the quality of the results obtained by the MCML algorithm for benchmark datasets with ground truth is highly accurate. We also compare MCML with other well known algorithms for datasets without ground truth, based on modularity metric for quality analysis, and conclude that MCML can detect communities roughly as meaningful as other known algorithms and in some cases even better (Facebook Forum). We manage to maintain a good balance between the quality of the results and running times as compared with the present state-of-the-art, which is a well known challenging problem in this area. We conclude that assigning edge strengths based on the topology of the given graph is key for ensuring good quality results. Our experiments also show good scalability and speed-up achieved by our MCML algorithm. The design of the MCML algorithm achieves scalability for some datasets (Facebook Forum) that is close to a k fold improvement in a k core processor, where ($k \geq 1$).

The MCML extracts disjoint communities, so one of our future research directions is to extend the ideas behind the topological feature analysis of the graph to assign edge strengths, performed by the MCML, to detect overlapping communities. Also, partitioning the network into sub-networks to achieve highest level of parallelism requires more cores. So it will be worthwhile to see if MCML can be extended to use message passing interface (MPI) in order to exploit cores of multiple machines, without incurring too much communication overhead. It will also be interesting to see whether the MCML can be modified to exploit GPU cores which would provide even higher scalability.

Contribution

Suely Oliveira and Rahil Sharma have equal contribution towards all aspects of this research paper.

Acknowledgements

We would like to thank the reviewers for their time and helpful advice.

References

Barabasi, A-L. and Oltvai, Z.N. (2004) 'Network biology: understanding the cell's functional organization', *Nature Reviews Genetics*, Vol. 5, No. 2, pp.101–113.

Blondel, V.D., Guillaume, J-L., Lambiotte, R. and Lefebvre, E. (2008) 'Fast unfolding of communities in large networks', *Journal of Statistical Mechanics: Theory and Experiment*, No. 10, P10008.

De Meo, P., Ferrara, E., Fiumara, G. and Provetti, A. (2011) 'Generalized louvain method for community detection in large networks', *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference*, IEEE, pp.88–93.

Fortunato, S. (2010) *Community Detection in Graphs*, Physics Reports, Vol. 486, No. 3, pp.75–174, Elsevier.

Fortunato, S., Latora, V. and Marchiori, M. (2004) 'Method to find community structures based on information centrality', *Physical Review E*, Vol. 70, No. 5, p.056104.

Girvan, M. and Newman, M.E.J. (2002) 'Community structure in social and biological networks', *Proceedings of the National Academy of Sciences*, National Acad. Sciences, Vol. 99, No. 12, pp.7821–7826.

Hashimoto, T., Chakraborty, B. and Shirota, Y. (2012) 'Social media analysis – determining the number of topic clusters from buzz marketing site', *Int. J. of Computational Science and Engineering*, Vol. 7, No. 1, pp.65–72.

Jancura, P., Mavroeidis, D. and Marchiori, E. (2012) 'DEEN: a simple and fast algorithm for network community detection', *Computational Intelligence Methods for Bioinformatics and Biostatistics*, pp.150–163, Springer.

Lee, J., Gross, S.P. and Lee, J. (2013) *Improved Network Community Structure Improves Function Prediction*, Scientific Reports, Vol. 3, Nature Publishing Group.

Liu, J-C., Liang, Y-C. and Lin, S-W. (2013) 'Selection of canonical images of travel attractions using image clustering and aesthetics analysis', *Int. J. of Computational Science and Engineering 2013*, Vol. 8, No. 4, pp.324–335.

Lusseau, D., Schneider, K., Boisseau, O.J., Haase, P., Slooten, E. and Dawson, S.M. (2003) 'The bottle nose dolphin community of doubtful sound features a large proportion of long-lasting associations', *Behavioral Ecology and Sociobiology*, Vol. 54, pp.396–405.

Newman, M.E.J. (2004) 'Fast algorithm for detecting community structure in networks', *Physical Review E*, Vol. 69, No. 6, p.066133, APS.

Newman, M.E.J. and Girvan, M. (2004) *Finding and Evaluating Community Structure in Networks*, Physics Reports, Vol. 69, No. 2, p.026113, APS.

Oliveira, S. and Seok, S-C. (2008) 'A matrix-based multilevel approach to identify functional protein modules', *International Journal of Bioinformatics Research and Applications*, Vol. 4, No. 1, pp.11–27, Inderscience.

Pons, P. and Latapy, M. (2005) 'Computing communities in large networks using random walks', *Computer and Information Sciences-ISCIS 2005*, Springer, pp.284–293.

Prat-Pérez, A., Dominguez-Sal, D. and Larriba-Pey, J-L. (2014) 'High quality, scalable and parallel community detection for large real graphs', *Proceedings of the 23rd International Conference on World Wide Web, International World Wide Web Conferences*, pp.225–236.

Prat-Pérez, A., Dominguez-Sal, D., Brunat, J.M. and Larriba-Pey, J-L. (2012) 'Shaping communities out of triangles', *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pg.1677–1681.

Radicchi, F., Castellano, C., Cecconi, F., Loreto, V. and Parisi, D. (2004) 'Defining and identifying communities in networks', *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 101, No. 9, pp.2658–2663.

Raghavan, U.N., Albert, R. and Kumara, S. (2007) 'Near linear time algorithm to detect community structures in large-scale networks', *Physical Review E*, Vol. 76, No. 3, p.036106, APS.

Rosvall, M. and Bergstrom, C.T. (2008) 'Maps of random walks on complex networks reveal community structure', *Proceedings of the National Academy of Sciences*, National Acad. Sciences, Vol. 105, No. 4, pp.1118–1123.

Rytsareva, I., Chapman, T. and Kalyanaraman, A. (2014) 'Parallel algorithms for clustering biological graphs on distributed and shared memory architectures', *Int. J. of High Performance Computing and Networking*, Vol. 7, No. 4, pp.241–257.

Soman, J. and Narang, A. (2011) 'Fast community detection algorithm with GPUs and multicore architectures', *Parallel & Distributed Processing Symposium (IPDPS), 2011 IEEE International*, pp.568–579.

Vieira, V.d.F., Xavier, C.R. and Ebecken, N.F.F. and Evsukoff, A.G. (2014) 'Modularity based hierarchical community detection in networks', *Computational Science and Its Applications – ICCSA 2014*, Springer pp.146–160.

Wang, Y. and Qian, X. (2013) 'Functional module identification in protein interaction networks by interaction patterns', *Bioinformatics*, btt569.

Oliveira, S. and Sharma, R. (2015) 'Identification and prediction of functional protein modules using a bi-level community detection algorithm', in final revision at *International J. Bioinformatics, Research and Application*, Inderscience.

Yang, J. and Leskovec, J. (2013) 'Overlapping community detection at scale: a nonnegative matrix factorization approach', *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ACM, pp.587–596.

Zachary, W.W. (1977) 'An information flow model for conflict and fission in small groups', *Journal of Anthropological Research, JSTOR*, pp.452–473.