

ABSTRACT

It is a well-known fact that complex systems in nature, society and technology are represented as networks. Uncovering the community structure in these networks is one of the most important and well-studied problem in this area [29]. In real-life complex networks, the vertices may belong to different communities at the same time, resulting in overlapping communities. Although the identification of overlapping communities is crucial to understand the structural and functional properties of a network, many of the traditional community detection algorithms tend to ignore them.

The main aim of this project is to develop an algorithm for identification of overlapping local communities, since so far the detection of overlapping communities was done globally and not locally. The importance of this local identification of the overlapping communities is that it does not require the knowledge of the entire structure of the network.

The main contributions and achievements of this project are:

- A thorough research in the area of community detection algorithms, and a comparison of the most important related algorithms, in order to select the one which fits the requirements of the project, see pages 3-17.
- The modification and extension of the algorithm proposed by Aaron Clauset [7], to tackle the problem of the overlapping communities, see pages 18-21.
- The development of a novel algorithm which detects the overlapping local communities of a node, see pages 21-34.
- The definition of a quality criterion for the evaluation of the communities identified by the novel algorithm, see pages 23-33.
- The testing of the created algorithm with computer-generated and real-life networks to evaluate the algorithm's performance and correct functionality, see pages 23-33, 35-44.

Despite the fact that we spend a lot of hours on the project, and despite the amount of work carried out, there is still space for improvements and extensions.

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr Steve Gregory for his great support and his useful comments on the project.

Finally, I would like to thank the professor Aaron Clauset for sharing his GPL version of the source code of the local modularity optimization algorithm.

CONTENTS

1. INTRODUCTION AND CONTEXT.....	1
1.1. AIMS AND OBJECTIVES	2
2. BACKGROUND AND PREVIOUS WORK.....	3
2.1. INTRODUCTION.....	3
2.2. MODULARITY.....	3
2.2.1. MODULARITY FOR DIRECTED NETWORKS.....	4
2.2.2. MODULARITY FOR NETWORKS WITH OVERLAPPING COMMUNITIES....	6
2.3. LOCAL METHODS FOR COMMUNITY DETECTION.....	9
2.3.1. LOCAL MODULARITY.....	9
2.3.2. OTHER METHODS.....	14
2.4. SUMMARY.....	17
3. PROJECT DESIGN.....	10
3.1. INTRODUCTION.....	18
3.2. PROBLEM SPECIFICATION.....	18
3.3. THE ALGORITHM.....	19
3.3.1. THE EXISTING ALGORITHMS.....	19
3.3.2. THE NEW ALGORITHM.....	21
3.3.3. EVALUATION OF THE COMMUNITIES.....	23
3.4. SUMMARY.....	34
4. EXPERIMENTAL RESULTS.....	35
4.1. INTRODUCTION.....	35

4.2. THE ZACHARY KARATE CLUB.....	35
4.3. THE SOCIAL NETWORK OF DOLPHINS.....	39
4.4. BOOKS ABOUT US POLITICS.....	41
4.5. SUMMARY.....	44
5. PROJECT EVALUATION.....	45
5.1. INTRODUCTION.....	45
5.2. ACHIEVEMENT OF AIMS AND OBJECTIVES.....	45
5.3. EVALUATION OF PROJECT'S MODULES.....	46
5.4. SUMMARY.....	47
6. FUTURE WORK.....	47
7. CONCLUSION.....	48
8. BIBLIOGRAPHY.....	49

1. INTRODUCTION AND CONTEXT

It is well known that graphs are used widely to represent different kind of structures in complex systems in nature and society. The Internet [1], World Wide Web [2], social networks [3], citation networks [4,5] and biological networks [6] are the most common examples of complex systems represented as graphs. In real networks, the distribution of edges is not homogeneous like in a random graph but is highly inhomogeneous with high concentrations of edges within certain groups of vertices [9]. These certain groups of vertices are called communities or clusters [10]. Although, an exact definition of what community is, does not exist, Newman [11] gave the following definition which is widely accepted: a community is a subgraph containing nodes which are more densely link to each other than to the rest of the graph. The concept of community can be found everywhere. From groups of people which have the same nationality, the same town, the same working and friendship cycles to online communities, communities in biology, in computer science, in economics, etc [9]. Due to the important properties of communities (high internal connectivity, high robustness, low path length among nodes [12]) many new algorithms for community detection have been proposed in the last few years. However, the construction of an efficient algorithm for community detection is highly non-trivial due to the fact that most of the existing algorithms require global knowledge of the graph's topology, which is unfeasible for large and dynamic networks. Also, a common feature of complex networks is to have nodes which belong to more than one community, resulting in overlapping communities. For example, people usually belong to different social communities according to their nationality, friends, hobbies, etc. [28]. The next figure illustrates the notion of overlapping communities (copied from [28]).

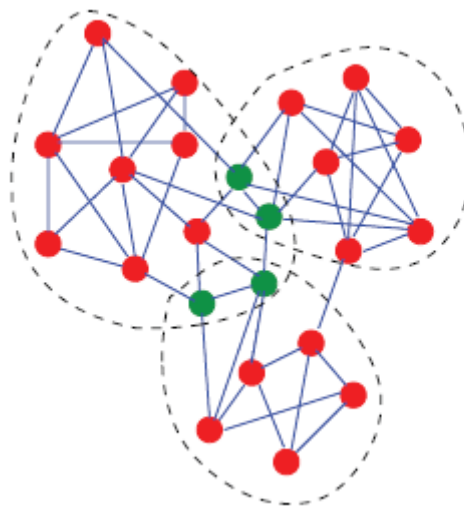


FIGURE 1

Most known divisive algorithms find separated communities and cannot uncover the overlapping community structure, [10, 20] and therefore lower the quality of the detected communities and lose important information about the network's structure [28].

In order to evaluate the goodness of a partition many quality functions were introduced. The most famous quality function, which was introduced by Newman and Girvan [11], is called *modularity* and it is considered the most widely accepted metric for community structure evaluation. Nevertheless, *modularity* requires knowledge of the entire structure of the graph, and also the algorithms which are based on the *modularity* optimization for community detection, deliver separated partitions of the graph. In order to overcome the first drawback of *modularity*, its localized version was introduced by Clauset [7]. This localized version is called *local modularity* and it is independent of the network's global properties, but it also ignores the overlapping communities in a network.

The principal goal of this project is the construction of a novel algorithm for overlapping local community detection. This algorithm consists of a novelty in the area of the community detection algorithms, since so far the detection of overlapping communities was done globally and not locally, i.e., without the knowledge of the network's global parameters.

1.1 AIMS AND OBJECTIVES

The aim of this project is, as mentioned before, to create a novel algorithm which will identify the overlapping local communities in a given network.

The objectives to meet this aim are the following:

- Investigate the algorithm introduced by Aaron Clauset [7] which detects a single community for a given node by maximizing the *local modularity* in a greedy fashion.
- Extend the above algorithm to identify not only a single community for a given vertex but also the overlapping communities of the vertex.
- Decide a criterion which will determine if an overlapping community identified by the algorithm is actually a “good” community for the node.
- Test the algorithm with real-life and artificial networks, with disjoint and overlapping communities, in order to evaluate the correctness of the algorithm.
- Accomplish all the aforementioned objectives by C++ programming.

2. BACKGROUND AND PREVIOUS WORK

2.1 INTRODUCTION

Community detection in complex networks is a very wide field. The purpose of this chapter is to summarize the most interesting findings in the above field. At the next section, the definition of *modularity* is given which is the most popular quality function for evaluating communities in networks. The following two sections are devoted to the two of the most interesting for us extensions of modularity: modularity for directed networks and modularity for networks with overlapping communities. Finally, local methods for community detection are presented.

2.2 MODULARITY

As said before *modularity* is the most popular and most used quality function for evaluating a particular division of a network. It is based on the idea that a community exists if the density of edges in this partition of a graph is higher than the density of edges in the same partition of the graph if the edges were randomly placed [11]. The above definition depends on the chosen *null-model*. This model is a duplicate of the original graph (same number of vertices, same number of edges and same degree distribution) without community structure. The *modularity* is defined as follows [12]:

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij}) \delta(C_i, C_j),$$

where m is the total number of edges in the graph and A_{ij} is the adjacency matrix defined as:

$$A_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are connected} \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, in the *null-model* which is a random graph, the probability P_{ij} for the node i to be connected to the node j is defined as:

$$P_{ij} = \frac{k_i k_j}{4m^2}, \quad \forall i, j$$

where k_i and k_j are the degrees of i and j respectively. Therefore, the equation for the *modularity* for the undirected graph becomes:

$$Q = \frac{1}{2m} \sum_{i,j \in V} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

where

$$\delta(c_i, c_j) = \begin{cases} 1 & \text{if } i \text{ and } j \text{ belong to the same community} \\ 0 & \text{otherwise.} \end{cases}$$

Seemingly, a high value of *modularity* indicates that the corresponding partition is a good one. Although the maximum value of *modularity* cannot be reached due to the fact that *modularity* optimization is an NP-complete problem [17], the following algorithms can find good approximation of this maximum: simulated annealing [18], greedy search [19], extremal optimization [20] and spectral optimization [21].

Firstly, simulated annealing is a probabilistic procedure for global optimization. Its aim is to find the global optimum of a function F , by transiting from one state to another in the space of the possible states. Simulated Annealing is considered the most accurate method [22] but it is very slow and only suitable for small graphs (up to 10^4 vertices). Secondly, greedy search is an agglomerative hierarchical clustering method introduced by Newman [23]. The basic idea is to divide the graph to n clusters, with each cluster containing only 1 vertex (no edges) and add connect two clusters with an edge at each step. The choice of the edge is such that the new partition gives the maximum increase of *modularity*. Although the many improvements which are made on this technique it is still consider not as accurate as the others. Furthermore, extremal optimization is a heuristic search procedure which aims to be as accurate as simulated annealing but faster [24] and represents a good tradeoff between accuracy and speed. Finally, spectral optimization is a fast method to optimize the *modularity* by using the eigenvalues and the eigenvectors of a special matrix called *modularity matrix*.

2.2.1 MODULARITY FOR DIRECTED NETWORKS

In the past, when the problem of identifying communities in directed networks occurred, the solution was to ignore the edge direction and to identify the communities with methods suitable for undirected networks. Due to this approach useful information contained in the edge direction were discarded. Moreover, it is a fact that many interesting networks including the World Wide Web, biological networks and food webs are directed networks, so an extension of the *modularity* optimization approach to deal with directed networks was crucial. Therefore, Arenas et al. [25] proposed a generalization of *modularity* to directed networks. This generalization indicates that given two vertices A and B , A with high out-degree but low in-degree and B with high in-degree but low out-degree, the probability of an

edge to be oriented from A to B is higher than the probability for the edge to be oriented from B to A. Consequently, since *modularity* should be higher for *surprisingly* configurations, an edge from B to A should make bigger contribution to the *modularity*. In order for this to work, the above mentioned *null-model* is replaced by its directed equivalent. For this new null model an edge between the vertices i and j exists with probability [12]:

$$P_{ij} = \frac{k_i^{\text{out}} k_j^{\text{in}}}{m} \quad \forall i, j$$

Then, the equivalent for the *modularity* equation which was defined before is [16]:

$$Q = \frac{1}{m} \sum_{ij} \left[A_{ij} - \frac{k_i^{\text{in}} k_j^{\text{out}}}{m} \right] \delta_{c_i, c_j}$$

This *modularity* can now be used for community detection using the same methods as before (simulated annealing, greedy search, and so on) for *modularity* maximization. Leicht and Newman [16] used this generalized *modularity* to create an algorithm based on the spectral optimization of the *modularity* which takes into account the edge direction and is able to recover the known community structure from a network.

Another approach utilizing the generalized *modularity* is the one described in [26]. This work exploits direction information, as described by Leicht and Newman [16] but with a different approach. Their method suggests that in order to find the communities in a directed network a transformation of this network into a new weighted graph is necessary. The probability of an edge to be oriented from j to i when the links are assigned randomly while keeping the direction is now [26]:

$$p_{ij} = \frac{k_j^{\text{out}} k_i^{\text{in}} / 2m}{k_j^{\text{out}} k_i^{\text{in}} / 2m + k_i^{\text{out}} k_j^{\text{in}} / 2m}$$

Then, they introduced they introduced the *relatedness* which indicates that when a *surprisingly* configuration happens like the one mentioned before when a directed link run from B to A, then the *relatedness* between A and B is stronger and a higher weight is given to that link. Moreover, small p_{ij} indicates stronger *relatedness* for the direction from j to i . The definition of the *relatedness* which is also the weight of the link between the vertices i and j for the new undirected graph is the follow [26]:

$$w_{ij} = A_{ij}(1 - p_{ij}) + A_{ji}(1 - p_{ji})$$

This new undirected network w_{ij} now contains all the direction information from the directed network and it is now ready to be examined using the methods suitable for undirected networks for community detection.

2.2.2 MODULARITY FOR NETWORKS WITH OVERLAPPING COMMUNITIES

It is a fact that real-world networks are never divided into sharp communities and their vertices are shared between these communities. This fact is obvious in social networks in which people belong to many different communities according to their social relationships and interactions. Many of the methods based on *modularity* optimization place each vertex into just one community and no overlaps among communities are allowed.

An interesting algorithm which extends *modularity* to tackle the problem of the overlapping communities in a directed network is the one proposed in [12]. The idea is that due to the fact that overlapping communities are allowed, each vertex can belong to many communities at the same time with a different strength. Therefore, an array of *belonging factors* $[a_{i,1}, a_{i,2}, \dots, a_{i,C}]$ was introduced which indicates how strongly a vertex i belongs to each of the communities. Furthermore, the coefficient of belonging to community c of an edge l which is oriented from vertex i to vertex j was introduced [12]:

$$\beta_{l,c} = \mathcal{F}(\alpha_{i,c}, \alpha_{j,c})$$

where the function F is arbitrary and can be defined for example as the product or the maximum value of the belonging coefficients. The modified *modularity* is defined as [12]:

$$Q_{ov} = \frac{1}{m} \sum_{i,j \in V} \left[r_{ij} A_{ij} - s_{ij} \frac{k_i^{\text{out}} k_j^{\text{in}}}{m} \right]$$

where r_{ij} and s_{ij} are the belonging coefficients which indicate the contribution of a vertex to the *modularity* of a given community. The definition of *modularity* can be further modified to present the evidence of the contribution of the *modularity* given by each community [12]:

$$Q_{ov} = \frac{1}{m} \sum_{c \in C} \sum_{i,j \in V} \left[r_{ijc} A_{ij} - s_{ijc} \frac{k_i^{\text{out}} k_j^{\text{in}}}{m} \right]$$

In order for this method to work, the aforementioned *null-model* has to be changed. Due to the overlapping communities, the probability of a vertex to belong to a

community with belonging factor $a_{i,c}$ does not depend on the probability that any other vertex does belong to the same community with belonging factor $a_{j,c}$. This fact didn't stand for the standard version of *modularity* in which the probability for two vertices to belong to the same community is higher if these two vertices are neighbors. Therefore, the new modified *null-model* for the overlapping communities is a duplicate of the graph but no particular community partition can be derived by its structural properties.

For the optimization of the modified *modularity* which is considered as fitness function a genetic algorithm is used to detect the overlapping communities in the network. This algorithm is based on the use of *chromosomes* which each one is represented by a matrix $M=(a_{i,c})$. This matrix is shown in the next figure (copied from [12]):

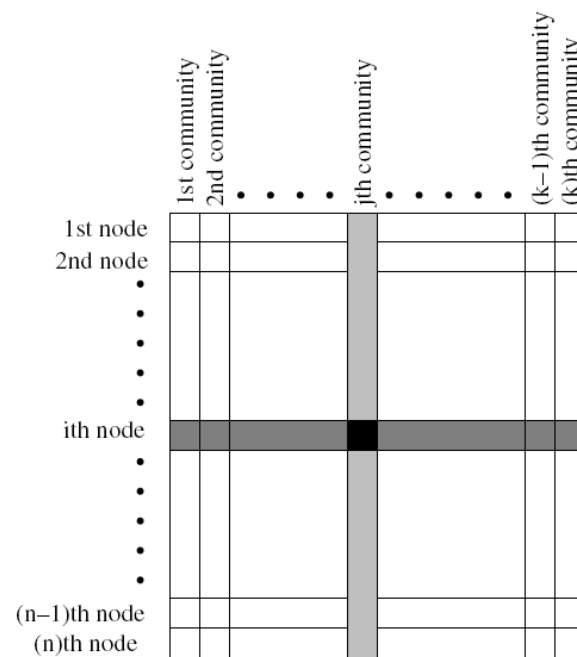


FIGURE 2

The algorithm is the following [12]:

Require: n_epochs , $n_individuals$, n_comm

- 1: $population = CreatePopulation(n_individuals, n_comm)$
- 2: **for** ($i = 0$ to n_epochs) **do**
- 3: $population.Fitness()$
- 4: $population.SortingAndSelection()$
- 5: $population.CrossOver()$
- 6: $population.Mutation()$
- 7: $population.CleanUp()$
- 8: **end for**

The variable n_epochs is the number of simulation cycles, $n_individuals$ is the number of individuals of the population and n_comm is the number of the overlapped

communities that the algorithm will try to find (not mandatory field). At the first step the chromosomes of the individuals are initialized by setting their belonging factors to a random number between [0.0 - 1.0]. At each iteration the following steps are taking place. Firstly, the fitness evaluation is taking place. For each individual's chromosome the modified *modularity* function is calculated. The individuals with the higher *modularity* value are included in the next generation. Some of the rest of the individuals which are going to be included in the next generation are created from scratch, while the others are obtained from the *crossover* mechanism. The aim of the *crossover* mechanism is to create better individuals by exchanging a portion of the *chromosomes* of two different individuals from the previous generation, hoping that the new individual will inherit their genetic structure. In order to achieve that, the *crossover* mechanism selects a random community from an individual A and copies it at the exact same place of the individual B. Then, the mutation operation selects a number of individuals and changes the factor of belonging to a random community for random a vertex. This way the chances of finding the optimal solution are improved. Finally, at the *cleanup* a vertex i and a community c are selected from a given chromosome and the average belonging factor for community c for the neighbors of i is computed along with the average belonging factor for community c that the vertices that are not neighbors of i . The following equation represents the above [12]:

$$\text{avgNeigh}(i, c) = \frac{\sum_{v \in \text{Neighbourhood}(i)} \alpha_{v,c}}{|\text{Neighbourhood}(i)|}$$

$$\text{avgNotNeigh}(i, c) = \frac{\sum_{v \notin \text{Neighbourhood}(i)} \alpha_{v,c}}{n - |\text{Neighbourhood}(i)|}$$

Afterwards, these two values are compared and if the first one is greater than the second then the belonging factor of vertex i for the community c is increased by a small value, else is decreased by a small value. This operation aims to improve the quality of the graph partition into overlapping communities by driving the algorithm towards a meaningful solution. The worst case scenario for the above algorithm has complexity $O(|C|*n^2)$ where $|C|$ is the number of the communities and n is the total number of the vertices in the network.

2.3 LOCAL METHODS FOR COMMUNITY DETECTION

2.3.1 LOCAL MODULARITY

Local modularity is a measure of community structure defined by Clauset [7] which depends only on the topology of some known portion of a graph. This local version of *modularity* is very useful when we lack global knowledge of a graph's topology or when the network is too large, takes time to construct or when the network are too dynamic like the World Wide Web. Clauset's algorithm is based on the greedy maximization of the *local modularity* in order to detect the local communities. Given a graph G and a known portion of the graph C the algorithm discovers one vertex-at-a-time the unknown portion of the graph, U by selecting a vertex which yields the largest increase of the *local modularity* R . The quality of the partition C in G is defined by the fraction of known adjacencies that are completely internal to C [7]:

$$\frac{\sum_{ij} A_{ij} \xi(i,j)}{\sum_{ij} A_{ij}} = \frac{1}{2m^*} \sum_{ij} A_{ij} \xi(i,j),$$

where A_{ij} is defined as before and ξ_{ij} is 1 if u_i and u_j are both in C and 0 otherwise. Further, the partition C is considered as a good one if the equation above has a large value, fact that indicates that the number of the internal connections in C is larger than the number of the connection to the unknown portion of the graph. Apart from the known portion of the graph C and the unknown portion of the graph U the boundary B has to be defined. B is a subset of C which includes the vertices which have at least one neighbor in U . Figure 1 illustrates the division of the graph into the three components C , B and U (copied from [7]).

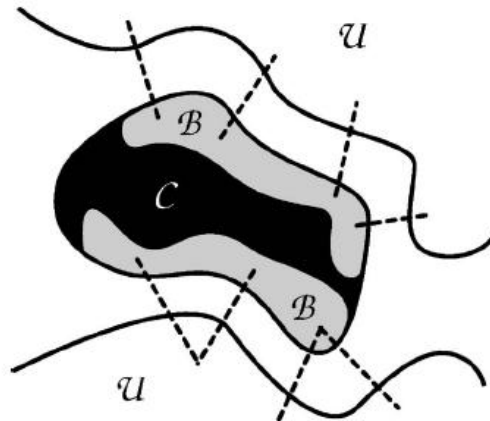


FIGURE 3

Obviously, a sharp boundary denotes a good partition because this partition will have few connections from its boundary to the unknown portion of the graph. Clauset's *local modularity* is the ratio of the number of edges having both endpoints in C (I) with the number of edges with one or more endpoints in B (T) and is defined as follows [7]:

$$R = \frac{\sum_{ij} B_{ij} \delta(i, j)}{\sum_{ij} B_{ij}} = \frac{I}{T},$$

where $\delta(i, j)$ is 1 when either u_i belongs to B and u_j belongs to C or vice versa, and 0 otherwise. Also, B is defined as [7]:

$$B_{ij} = \begin{cases} 1, & \text{if vertices } i \text{ and } j \text{ are connected,} \\ & \text{and either vertex is in } B; \\ 0, & \text{otherwise.} \end{cases}$$

The algorithm for the greedy maximization of the *local modularity* starts by selecting a source vertex and put it in C. The neighbors of this vertex are placed in U. At each iteration of the algorithm for each vertex that belongs to U is calculated the value ΔR which is the change in *local modularity* if the specific vertex is added to C. Then, the vertex with the largest value of ΔR is chosen to be placed in C (and to B if is necessary) and the neighbors of this vertex are placed in U. From the equation of the *local modularity* can be derived the equation of ΔR as follows [7]:

$$\Delta R_j = \frac{x - Ry - z(1 - R)}{T - z + y},$$

where x is the number of edges in T which have an endpoint the u_j , y is the number of edges that will be added at T and z is the number of edges that will be removed from T. The above algorithm continues until the algorithm discovers the entire enclosing component or until the number of vertices at C reaches a given number of vertices k. The pseudocode of this algorithm is the following (copied from [7]):

```

add  $v_0$  to  $\mathcal{C}$ 
add all neighbors of  $v_0$  to  $\mathcal{U}$ 
set  $\mathcal{B}=v_0$ 
while  $|\mathcal{C}| < k$  do
  for each  $v_j \in \mathcal{U}$  do
    compute  $\Delta R_j$  from Eq. (5)
  end for
  find  $v_j$  such that its  $\Delta R_j$  is maximum
  add that  $v_j$  to  $\mathcal{C}$ 
  add all new neighbors of that  $v_j$  to  $\mathcal{U}$ 
  update  $R$  and  $\mathcal{B}$ 
end while

```

This greedy optimization runs in time $O(k^2d)$ when it explores k vertices and d is the average degree of the graph.

Another algorithm that uses only local information to detect communities in a network was proposed in [8]. This algorithm is quite similar to Clauset's algorithm with the difference that is based on a table that describes the network and on a virtual cache memory. The objective of the authors of this paper was by using Clauset's *local modularity* to obtain better detection results and computational performance. The modified *local modularity* is [8]:

$$Q_l = \frac{L_{in}}{L_{in} + L_{out}},$$

where L_{in} is the number of edges which have both their endpoints in the extracted community (C in Clauset's algorithm) and L_{out} is the number of edges that have only one vertex in the extracted community. The basic idea of this algorithm is shown in the next figure (copied from [8]).

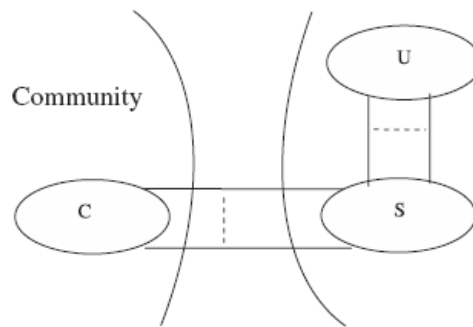


FIGURE 4

At each iteration, the algorithm searches the cluster S to find the vertex that causes the largest improvement to the *local modularity* of the cluster C . This edge is added to C and then the cluster S is updated from the cluster U . This procedure is repeated until the *local modularity* of the cluster C cannot be improved anymore or until the algorithm searched the whole network. In order for this algorithm to work a table T is needed. This table has 4 columns and describes the whole network. The first column of the table called *node_num* and is the serial number of the vertex. The second column is called *cnn_str* and is a string of the vertices which the vertex is connected. Moreover, *deg* is the degree of the vertex and *comm_num* is the serial number of the community which the vertex belongs. If the vertex does not belong to a community then this number is zero. Due to the fact that this algorithm needs the information about vertex degrees frequently a hash table H is needed in order to retrieve this information at a constant time.

The following sub-algorithm extracts a community from the network and can be repeated until all of the network's communities are extracted. Firstly, the table T is initialized setting the *comm_num* of each row to zero. Thereafter, the hashtable H is initialized by mapping each vertex to its degree. A random vertex is selected and added to community C_i (by setting the column *comm_num* = i). Also initialize cluster S and *local modularity* q to zero. Secondly, the hashtable H is searched for the vertices which are neighbors of the vertices in C_i and their *comm_num* column is zero. These vertices are added to S . If no such vertices exist the algorithm jumps to the fifth step. At the third step the value of the *local modularity* q is calculated for each vertex in S . If for a specific vertex the value of the *local modularity* is lower than the current value of the *local modularity* twice consequently then the vertex is deleted from S (virtual cache). After, the vertex with the max value of q is selected and added to C_i and the current value of q is updated. If no such vertex exists then again the algorithm jumps to the fifth step. The second and the third steps are repeated until one of them jumps to the fifth step in which the C_i is the extracted community. A flow-chart which represents the above algorithm is the following (copied from [8]):

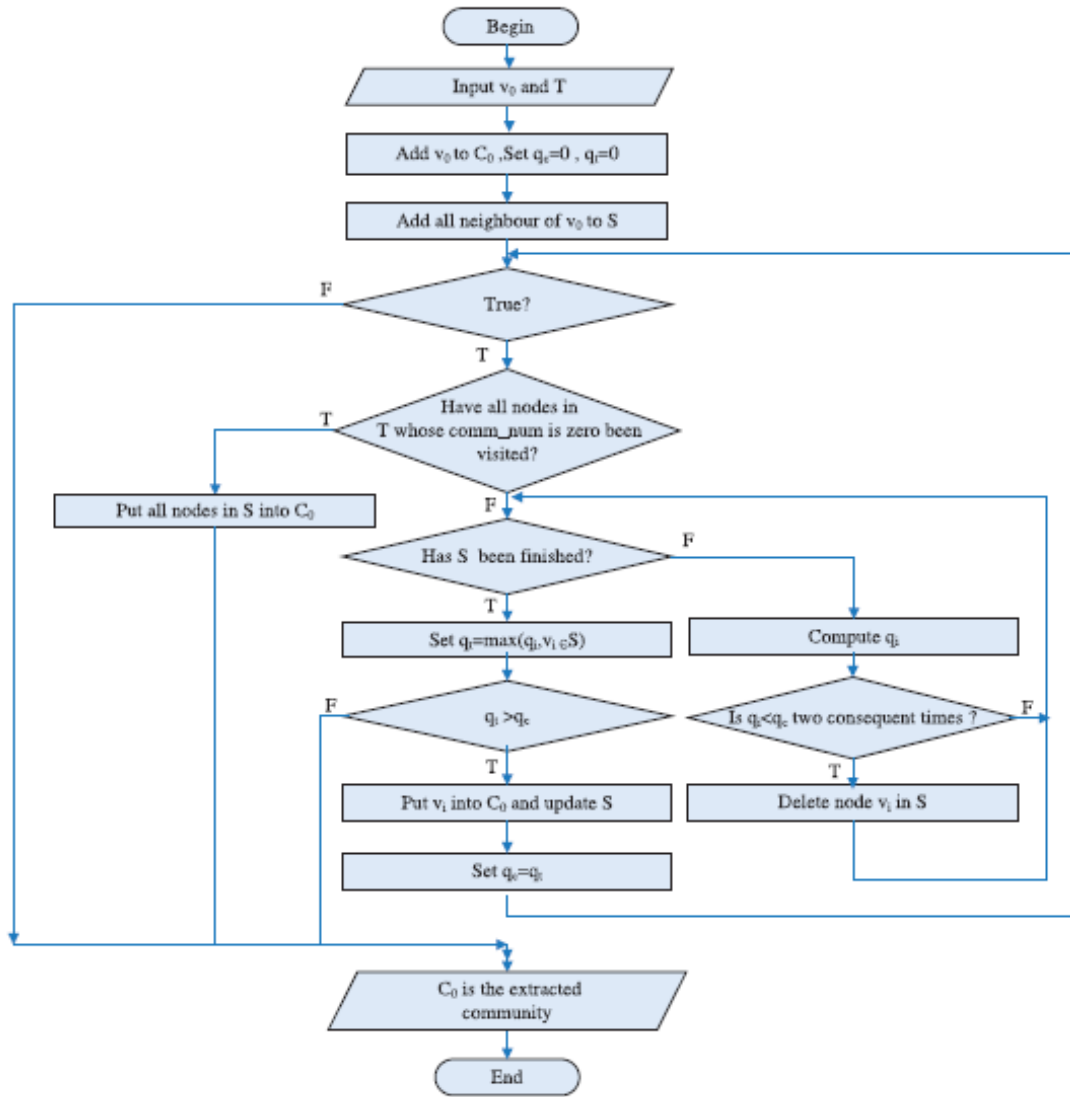


FIGURE 5

As said before the above sub-algorithm extracts one community from the graph and in order to detect all of the communities the algorithm has to be repeated in the remaining sub-network. The procedure of extracting the communities of the network can be simulated by the following binary tree (copied from [8]):

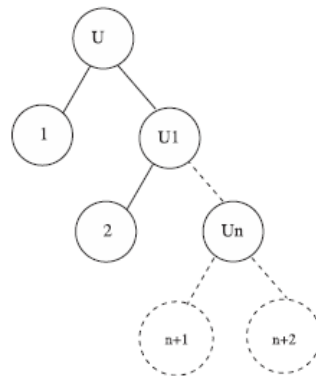


FIGURE 6

in which the left-leaf vertices is the extracted communities, the right-non-leaf vertices are the undivided sub-networks which need to be further analyzed and the right-leaf vertices are the final extracted communities. The worst running time of this algorithm is $O(k^2n)$ where n is the total number of vertices and k is the mean vertex degree of the total network. Due to the fact that a virtual cache is used (penalization of the vertices which do not improve the *local modularity* of the cluster C in two consequent steps), the running time of the algorithm is much faster than the worst running time mentioned above. It is worth to notice that the beginning of the algorithm with a different initial vertex may affect the final community partition. However, the final community partitions are acceptable indicating that the proposed algorithm detects the communities in a network successfully.

Both of the algorithms above are applied to a set of computer-generated networks but also and to real-network models. The interesting fact is that the second algorithm outperforms Clauset's algorithm significantly when computer-generated networks with known community structure are analyzed. Specifically, the second algorithm classifies approximately 80% of the vertices correctly even when the community structure is quite weak, in contrast to Clauset's algorithm which classifies on average more than 50% of the vertices [7, 8].

2.3.2 OTHER METHODS

Local modularity is not the only method for detecting the communities of a graph without requiring knowledge of the entire network. A local method called *l-shell* proposed in [27] is very useful when a member of a network tries to find his own communities, although the authors proposed a general method to identify all of the communities in a network. The *l-shell* method is based on the idea of an *l shell* which expands outwards a starting vertex. The first shell contains the neighbors of the starting vertex, the second shell contains the next-to-nearest neighbors of the starting vertex and so on. At each iteration of the algorithm, the change in the total emerging degree of the expanded *l shell* is compared with a predefined threshold α ; if this change exceeds the threshold then the vertices of the new shell are added to the cluster, else the algorithm stops. The total emerging degree is the sum of the emerging degrees of all vertices on the leading edge (set of all vertices which are l steps away from vertex j) of the *l shell* and is defined as [27]:

$$K_j^l = \text{total emerging degree of a shell of depth } l \text{ starting from vertex } j.$$

The emerging degree of a vertex is the number of edges that connect this vertex to vertices that the *l shell* has not yet reached. Therefore, the change in the total emerging degree is defined as [27]:

$$\Delta K_j^l = \frac{K_j^l}{K_j^{l-1}}$$

The pseudocode which describes how a vertex's community is determined is the following (copied from [27]):

```

s ∈ V; // s is the starting vertex

 $K_s^{d-1} \leftarrow 1$ ;

 $Q \leftarrow$  empty queue; // search queue

 $C \leftarrow$  empty queue; // community queue

enqueue s → Q;

 $K_s^d \leftarrow \text{emerging}(Q, C, G(V, E))$ ;

 $\Delta K_s^d \leftarrow \frac{K_s^d}{K_s^{d-1}}$ ;

while  $\Delta K_s^d > \alpha$  do
     $K_s^{d-1} \leftarrow K_s^d$ ;

    foreach  $q \in Q$  do
        dequeue  $q \leftarrow Q$ ;

        enqueue  $q \rightarrow C$ ;

        enqueue neighbors(q) → Q;
    end

     $K_s^d \leftarrow \text{emerging}(Q, C, G(V, E))$ ;

     $\Delta K_s^d \leftarrow \frac{K_s^d}{K_s^{d-1}}$ ;

end

```

From the above pseudocode it is obvious that the spreading of the *l shell* is controlled by the threshold *a*. If this threshold is too big then the spreading of the *l shell* will stop sooner, and if the threshold is too small the *l shell* will be spreading until the entire graph is visited. So, intermediate values of the threshold *a* are the more appropriate and yield to good partitions of the graph. The efficiency of this algorithm depends on the position of the start vertex. If the start vertex's position is close to some vertices which are not belong to a community or the position of the start vertex is equidistant

to the above mentioned set of vertex and to some vertices which belong to a community then the *l shell* will expand along two or more communities at the same time. This algorithm successfully finds a vertex's community if the position of the start vertex is equidistant from the boundary of its community. To overcome this problem the authors suggested the use of a *membership matrix* which slows down the procedure ($O(n^3)$) although the algorithm without this matrix is very fast.

The *Bridge Bounding* methodology which was presented in [28] is very similar to the *l shell* methodology. The proposed algorithm explores the network around a starting vertex, adding vertices to the community until it identifies the edges which act as the boundaries of the community. In order to identify which of the edges act as bridges between the communities, the use of local network topology functions is essential. These functions examine the network structure around an edge and quantify on which extent this edge act as a bridge. Examples of these functions are the *betweenness* and the *edge clustering coefficient*. The algorithm starts with a vertex s and adds the vertex's neighbors to the community if these vertices do not belong to any other community and if the edges connecting these vertices to s are not community boundaries. This procedure is repeated for all the new vertices in the community and finishes when no new vertices fulfill the above requirements, so adding new vertices to the community is impossible. The algorithm for the *Bridge Bounding* is the following (copied from [28]):

```

Require: Seed node  $s \in G = (V, E)$ 
Require: Community mapping  $g_C : V \rightarrow \mathbf{P}$ 
Require: Bridge function  $b : E \rightarrow [0.0, 1.0]$ 
1:  $C_s = \emptyset$ 
2: Frontier set  $F = \{s\}$ 
3: while  $|F| > 0$  do  $\{F$  is non-empty $\}$ 
4:    $c \leftarrow F.\text{pop}()$ 
5:    $C_s \leftarrow C_s \cup \{c\}$ 
6:    $C_U \leftarrow C_U \setminus \{c\}$ 
7:   for all  $n \in N(c)$  such that  $e_{cn} = (c, n) \in E$  do
8:     if  $g_C(n) = C_U$  and  $b(e_{cn}) \leq B_L$  then
9:        $F.\text{push}(n)$ 
10:    end if
11:  end for
12: end while
13:  $\mathbf{P} \leftarrow \mathbf{P} \cup C_s$ 

```

FIGURE 7

The drawback of this algorithm is that there is no obvious way to set the threshold B_L which identifies the edges which act as community boundaries from the others. Moreover, when the topology function is based on the edge clustering coefficient, the detection of the community structure is successful in synthetic networks but unsuccessful with networks of scale-free topology. To overcome this problem they

used a measure which calculates the weighted sum of the edge clustering coefficient of the edge and the mean edge clustering coefficient of the edges consisting its neighborhood. By applying the above measure they achieved the smoothing of the local bridging function and therefore smoother estimates of the bridging properties of the edges are derived. The version of the algorithm which uses the above measure has time complexity $O(d^2m+d^2n)$ where d is the average degree of the graph and m is the number of edges and n is the number of vertices.

2.4 SUMMARY

The aim of this chapter was to present the related work which was already done in the area of community detection algorithms. The most popular quality function called modularity was defined, along with its extensions to cover directed networks and networks with overlapping communities.

3. PROJECT DESIGN

3.1 INTRODUCTION

This chapter describes all the necessary algorithms and procedures followed in order to create the overlapping local community detection algorithm. All the implemented or investigated issues are described thoroughly.

Firstly, the specification of the problem is given, and then, the necessary procedures for the construction of the algorithm are described. Finally, a quality criterion for the evaluation of the identified communities is defined and established with the assistance of benchmark graphs.

3.2 PROBLEM SPECIFICATION

In order to develop an algorithm for identification of the overlapping local communities of a given node in a network, a general measure of local community structure had to be selected. This measure had to be based only on the topology of a known portion of the network. Due to the great deal of attention that the inference of community structure has attracted, a wide variety of quality measures and their corresponding algorithms exist. Therefore, in order to choose the most appropriate quality measure, a thoroughly research and comparison of the measures had to take place.

Once the quality measure is decided, the next issue to consider was, which nodes, apart from the source node, we had to choose, such that their natural community includes the source node and resulting the source node's overlapping communities. The above issue is formidable, since a node may be included in many communities which start from different nodes.

Last but not least, after the identification of the overlapping local communities of the source node, a quality criterion had to be decided indicating if the found communities are as good as the source node's natural community. This is a difficult task and requires a lot of investigation because so far there is not a criterion indicating the goodness of the local communities. Also, this task is the one which is going to reveal the effectiveness and the correct functionality of the created algorithm.

3.3 THE ALGORITHM

The development of the innovative algorithm for overlapping local community identification requires the usage and the extension of an existing algorithm which identifies the natural community for a given node.

3.3.1 THE EXISTING ALGORITHMS

Initially, in order to identify the appropriate quality function and the corresponding algorithm which was going to be used as the basis for the creation of the new algorithm, several approaches had to be considered.

The first approach followed was introduced by Lancichinetti, Fortunato and Kertesz [28] and was based on the local optimization of a fitness function. Their work was motivated by the existence of overlapping communities in networks and by the existence of hierarchy of modules due to the community nesting, i.e., small communities form larger communities. The hierarchical structure of a network is presented in the next figure (copied from [28]).

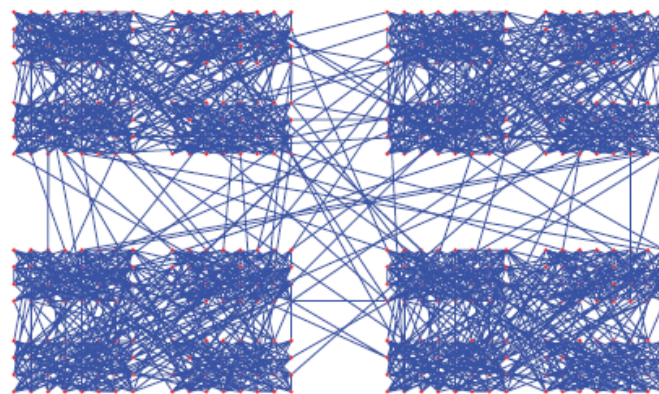


FIGURE 8

Consequently, their goal was to create an algorithm which detects both the overlapping communities, and the network's hierarchical organization. Their method achieves that by discovering the natural community of each node in the network. However, due to the fact that the natural communities of many nodes usually coincide, the created algorithm discovers only the natural communities of nodes which are not yet assigned to any community. The algorithm recovers all the communities without detecting the natural community of every node, which would be an expensive operation, since the nodes in the communities are either overlapping nodes or not [28].

The part of the above algorithm which fits our needs and which was attempted to be extracted from the overall procedure is the identification of the natural community of a node. This can be achieved by the greedy maximization of a fitness function defined as followed:

$$f_G = \frac{k_{in}^G}{(k_{in}^G + k_{out}^G)^\alpha}$$

where k_{in} is the total internal degree, k_{out} is the total external degree and α is a resolution parameter which tunes the size of the communities. The total internal degree is the double of the number of the edges which are internal in the community, and the total external degree is the number of edges which connect the community to the rest of the graph. The procedure which is followed for the detection of the natural community is to loop over all of the neighbors of the community to identify the one which results the largest fitness. Then the fitness of each node in the community is recalculated and all the nodes with negative fitness are removed from the community. The fitness of each node is calculated by subtracting the fitness of the community with the specific node included, and the fitness of the community with the node not included as followed [28]:

$$f_G^A = f_{G+\{A\}} - f_{G-\{A\}}.$$

Despite the good performance of the above algorithm, the source code, which was written in C++, was unstructured and scattered to many files, facts that made the understanding and modification of the algorithm extremely difficult. Therefore, after a lot of effort and a lot of time which was spent on the modification of the source code, the decision to follow a different approach was the only solution.

The second approach was an algorithm based on a greedy optimization of the aforementioned *local modularity* and it was introduced by Clauset [7]. This algorithm was described at the previous chapter *Background and Previous Work*, but for the sake of clarity, the algorithm's main goals will be pointed out again. The definition of the *local modularity* R is crucial for both Clauset's algorithm and for our novel algorithm, so it is presented again [7]:

$$R = \frac{\sum_{ij} B_{ij} \delta(i,j)}{\sum_{ij} B_{ij}} = \frac{I}{T},$$

Here, the *local modularity* is defined as the ratio of the edges which are internal (I) to the known portion of the graph (community), to the total number of edges (T). The total number of edges is defined as the number of the internal edges along with the edges which are external to the known portion of the graph, i.e., the number of edges connecting each node of the community to the rest of the graph. The basic idea of Clauset's algorithm is to detect the natural community of a source node, by adding to the known portion of the graph the node which yields the largest change in *local modularity* R , as a result of joining the node to the community. The natural community of the vertex is discovered by exploring the graph one vertex-at-a-time. Initially, the community denoted by C , includes only the source node. The nodes which are not included in the community but they are adjacent to the nodes inside the

community are placed in a deferent set of vertices denoted by U . At each iteration of the algorithm, a vertex which belongs to U and which results the largest change in the *local modularity* is joined to the community. Also, the set U is updated by adding the additional unknown adjacent vertexes. The procedure stops when the enclosing community is discovered [7]. The next figure shows the pseudocode for the above procedure (copied from [7]).

```

add  $v_0$  to  $\mathcal{C}$ 
add all neighbors of  $v_0$  to  $\mathcal{U}$ 
set  $\mathcal{B}=v_0$ 
while  $|\mathcal{C}|<k$  do
  for each  $v_j \in \mathcal{U}$  do
    compute  $\Delta R_j$  from Eq. (5)
  end for
  find  $v_j$  such that its  $\Delta R_j$  is maximum
  add that  $v_j$  to  $\mathcal{C}$ 
  add all new neighbors of that  $v_j$  to  $\mathcal{U}$ 
  update  $R$  and  $\mathcal{B}$ 
end while

```

FIGURE 9

Obviously, since Clauset's algorithm successfully discovers the natural community of a given vertex, was the most suitable algorithm to extend in order to achieve the detection of the overlapping local communities.

3.3.2 THE NEW ALGORITHM

Overlapping communities are the result of the sharing of the nodes between different communities. In order to identify the communities which a node belongs to, we have to choose specific nodes, such that their natural community includes the source node. But how do we know which nodes to try? One solution is to try all of the nodes in the network and select the nodes which their community includes the source node. Nonetheless, this solution is highly inappropriate, since it is time consuming and requires the knowledge of the network's global topology. Therefore, the new algorithm is based on the following assumption: A node belongs to its natural community, plus to the communities which are included to an extended neighborhood of the node. So, it is more than enough to detect all the natural communities of the neighbors of the source vertex to have a completed set of the vertex's overlapping communities. Someone can tell that you can find communities which include the source vertex if you start from a vertex which is not a neighbor of your vertex. It is worth to mention that, the goal of an algorithm which identifies the overlapping local

communities of a node is **not** to identify **all** of its overlapping communities but **only** these overlapping communities which are considered **as good as the node's natural community**. So, the community of a node which includes the source node but it is not in source node's neighborhood, does not necessarily consists a good overlapping community for the source node.

In order to achieve the identification of the overlapping local communities of a source vertex, the next steps must be followed. Firstly, the natural community of the source vertex must be identified using the algorithm which was introduced by Clauset [7], and which was described earlier in this report. The natural community of the source vertex along with the value of the *local modularity* for this community is stored in a file.

Secondly, all the nodes which are adjacent to the source vertex must be identified and for each one of them, the Clauset's algorithm must be re-run in order to identify the overlapping communities. This can be achieved by exploring the neighborhood of the source vertex, one vertex-at-a-time and for each adjacent node perform the above procedure to detect its natural community, and the corresponding *local modularity*. After the completion of the above process, a certain number of files, equal to the number of the source vertex's neighbors, are generated, and each file contains the natural community of each one of the neighbors. Also another file is generated which contains the value of the *local modularity* for each one of the extracted communities.

Although all the natural communities around the source vertex are generated, not all of them include the vertex and obviously not all of them are good overlapping communities for our source vertex. So, the next step is to identify which of the extracted communities includes the starting vertex. This is a trivial thing to do and can be achieved by exploring each one of the identified communities, and point out the communities which include our source vertex. Then, the maximum value of the *local modularity* is calculated for comparison purposes.

Consequently, through the above procedure we achieved to identify the natural community of the source node, to extract the natural communities of the neighbors of the starting vertex and to exclude those communities which do not overlap with the source node's natural community, i.e., they do not include the source node in their community. The only thing which remains, and which is the most important in the whole procedure, is to define a criterion which will indicate the quality of the found communities, and also if these communities are as good as the source node's natural community.

3.3.3 EVALUATION OF THE COMMUNITIES

As mentioned before, a quality criterion is necessary in order to evaluate the goodness of the found overlapping local communities. The definition of this kind of criterion is a demanding task to achieve, since this criterion does not exist for overlapping local communities, and also requires the use of benchmark graphs to evaluate the findings.

Benchmark graphs are a widely used option for the most of the researchers who want to evaluate their algorithms. In contrast to artificial graphs, with a built in community structure, which do not reflect the properties of nodes and communities, benchmark graphs take into consideration the heterogeneity in the distributions of node degrees and the size of the communities. Moreover, the benchmark graphs which were introduced by Lancichinetti and Fortunato [29] were initially unweighted, undirected and have the possibility to have overlapping communities, an option which is vital for the testing of our algorithm. Thereafter, Lancichinetti and Fortunato extended their algorithm to the case of weighted and directed networks. The generation of the communities and the classification of the nodes to the available communities at [29] is equivalent to generating a bipartite network. To illustrate that point, imagine a bipartite network in which the first set includes the nodes and the second set includes the communities. The next figure shows the aforementioned bipartite network (copied from [29]).

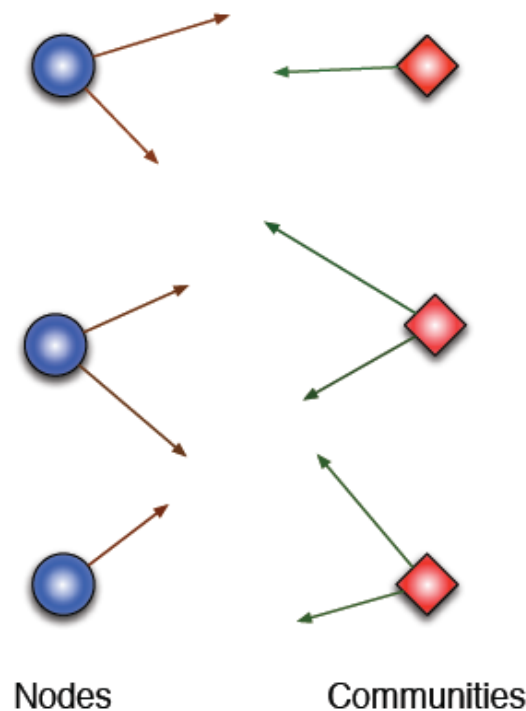


FIGURE 10

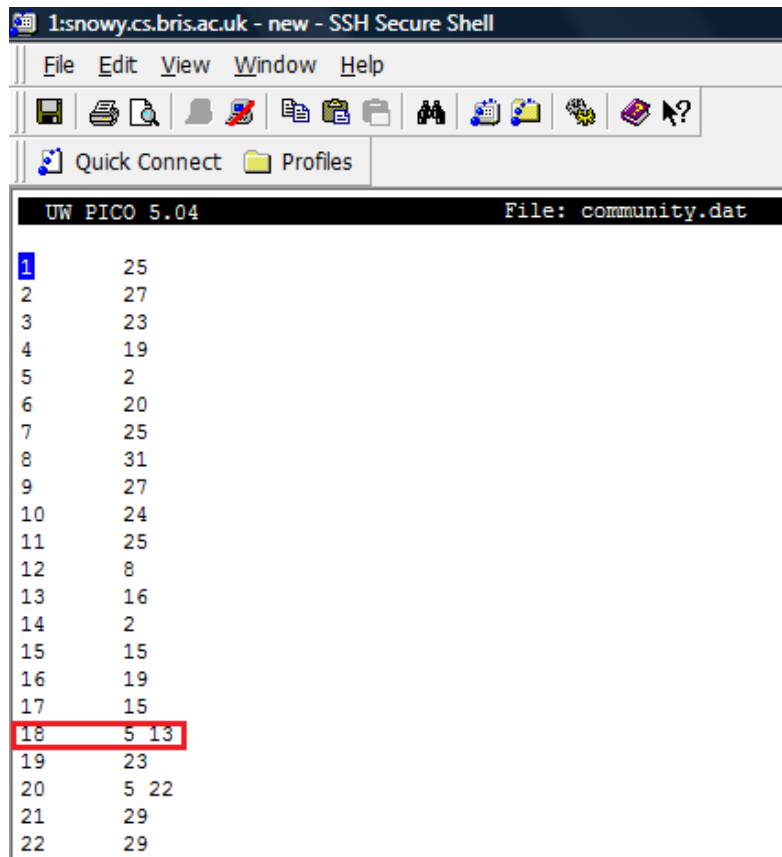
Each node has as many links as the number of its memberships to the communities and each community has as many links as its size [29].

For testing purposes, the goal is to create different kinds of these benchmark networks by varying the available parameters. The first parameter which the algorithm requires to be initialized is the number of nodes in the network, N . Secondly, the parameter k ,

indicates the degree of the nodes, and the parameter *maxk* indicates the maximum degree which a vertex is possible to have. Furthermore, the parameter *mu* is the mixing parameter which denotes how many of the neighbors of a vertex have at least one membership to a community in common with this vertex. Also, *t1* and *t2* are the minus exponent of the degree sequence and the minus exponent for the community size distribution respectively. The parameter *minc* is the minimum number for the community sizes, and *maxc* is the maximum number for the community sizes. Finally, the parameter *on* indicates the number of the overlapping nodes, i.e., how many of the nodes belong to more than one communities, and the *om* indicates the number of memberships of the overlapping nodes, i.e., in how many communities the overlapping nodes are allowed to participate [29].

In order to start evaluate the performance of the newly created algorithm we have to generate the first benchmark graph. The first goal is to assure by testing, that if overlapping nodes exist, then the algorithm can successfully find the communities which the node belongs to. To achieve that, a benchmark graph is created with the following parameters: the number of total nodes *N* in the network is set to 1000, the number of overlapping nodes *on* is set to 20 and the number of memberships of the overlapping nodes is set to 2. By initializing the algorithm with the above values, the algorithm will produce a benchmark network with 1000 nodes, 980 of them with only one membership and 20 of them with two memberships to the communities. The other parameters are set to the following values: the maximum degree of the node *maxk* is set to 50, the mixing parameter *mu* is set to 0.2, the minus exponent of the degree sequence *t1* is set to 2, the minus exponent for the community size distribution *t2* is set to 1, the minimum number for the community sizes *minc* is set to 20 and the maximum number for the community sizes *maxc* is set to 50. The total number of edges of the graph is 7410.

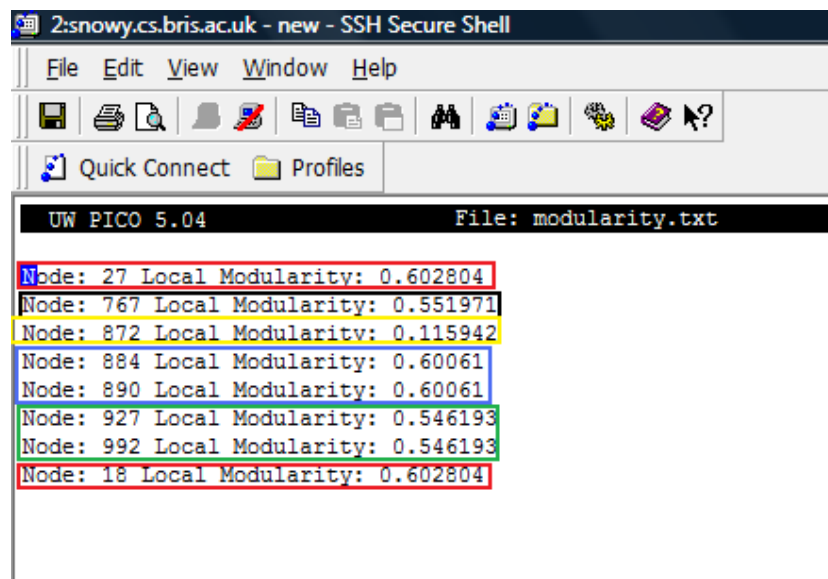
To meet the first goal, which is to assure that if a node belongs to more than one community, then these communities are identified by the algorithm, we have to choose a random vertex from the benchmark network which is generated with the above mentioned parameters, and run our algorithm with that vertex as a starting vertex. Let's assume that we selected the vertex 18. This vertex belongs to the communities denoted by the number 5 and 13, as shown in the following screenshot which displays the vertexes, and the communities which these vertexes belong to (left column: vertexes, right column: communities). So, if we test our algorithm using the vertex 18 as the starting vertex, the algorithm should detect only the two communities in which the vertex belongs to.



Index	Value
1	25
2	27
3	23
4	19
5	2
6	20
7	25
8	31
9	27
10	24
11	25
12	8
13	16
14	2
15	15
16	19
17	15
18	5 13
19	23
20	5 22
21	29
22	29

FIGURE 11

When we run the overlapping local community detection algorithm, we get the following results. At first, a file is generated which includes all the neighbors of the starting vertex, which their natural community overlaps with the community of the vertex, along with the value of the *local modularity* for each of the identified overlapping communities. Note that these communities are just the communities which include the starting vertex but not all of them are good overlapping communities for our node. The above mentioned file is displayed in the next figure.



Node	Local Modularity
Node: 27	0.602804
Node: 767	0.551971
Node: 872	0.115942
Node: 884	0.60061
Node: 890	0.60061
Node: 927	0.546193
Node: 992	0.546193
Node: 18	0.602804

FIGURE 12

The communities which have the same value for the *local modularity* are identical communities. As it can be observed from the above screenshot of the file, the algorithm detects six different communities which are denoted by the different colors. The red color denotes the natural community of the source vertex 18 which is also displayed in the next figure.

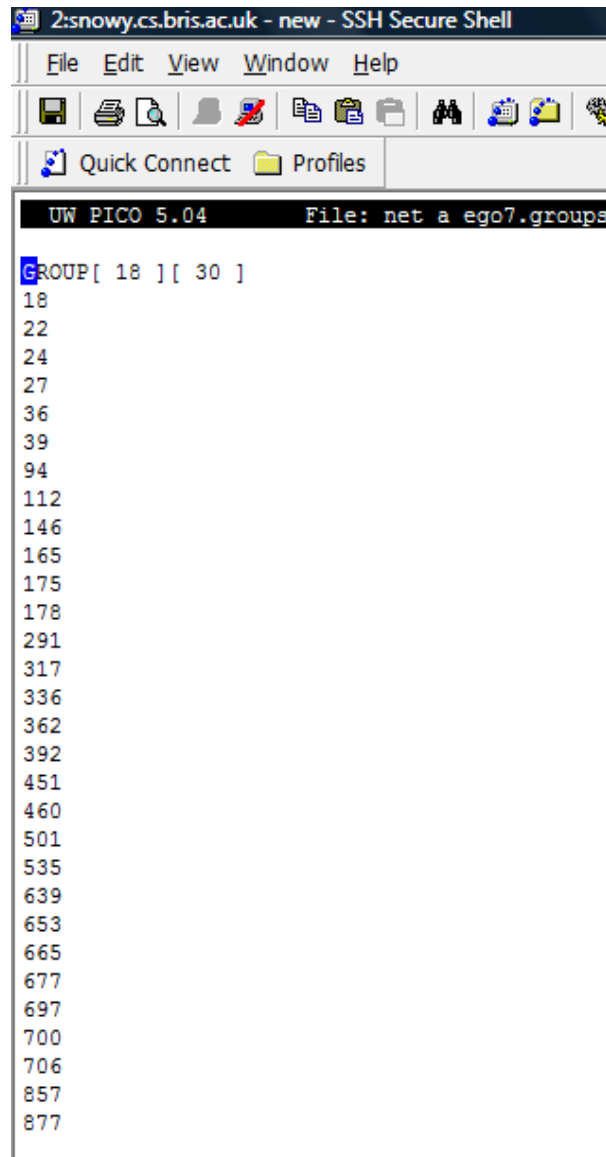
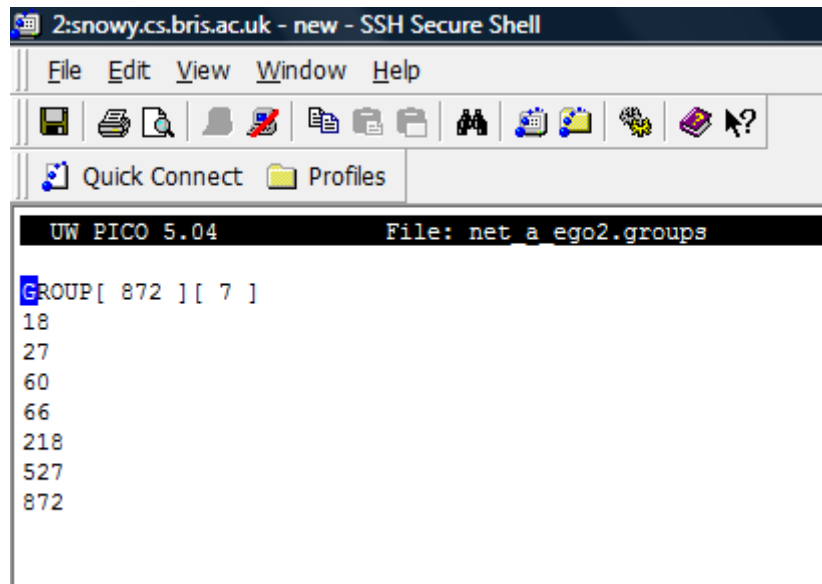


FIGURE 13

The community with the yellow color has very low *local modularity* value, and without any consideration this community is not included in the list of the possible overlapping communities for the starting vertex. This low value of the *local modularity* can be explained if the community which starts from the neighbor 872 is investigated. This community is displayed in the next figure.



```
2:snowy.cs.bris.ac.uk - new - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
UW PICO 5.04 File: net_a ego2.groups
GROUP[ 872 ][ 7 ]
18
27
60
66
218
527
872
```

FIGURE 14

The first line of the file indicates firstly, the name of the neighbor of the source vertex which its community was identified by our algorithm, in this case is the node 872, and secondly the size of the community which is 7. Clearly, the above community does not fulfill the requirements which are necessary for defining it as a good overlapping community for the starting vertex, for following reason. The parameter *minc* for the benchmark graph, which is the minimum number for the community sizes, was set to 20, but the identified community has community size 7. The above fact, along with the low *local modularity* value, exclude immediately the community from the candidate overlapping communities.

It is notable from figure 12, that the community with the blue color has as high modularity as the natural community of the source node. Therefore, this community needs further investigation in order to conclude if it is the second community in which the source node belongs to, as the benchmark graph indicates. The next figure shows the file containing the natural community of the node 884 (which is the same with the natural community of the node 890).

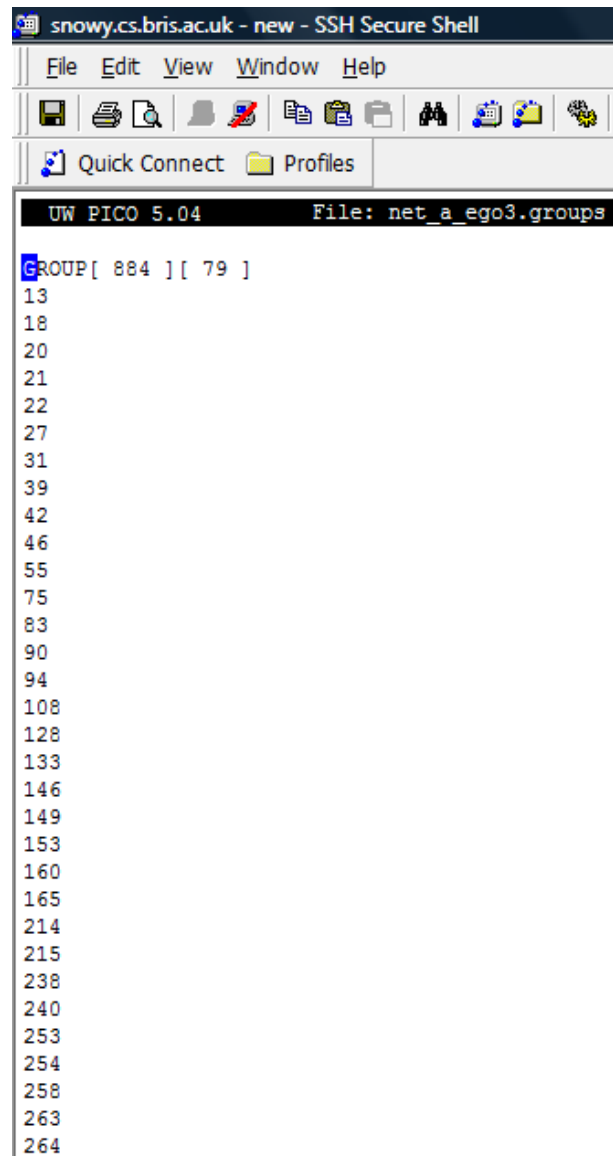


FIGURE 15

As it can be noticed, the natural community of the node has size 79 (not all the nodes of the community can be displayed in the figure). However, the benchmark graph clearly indicates through the *maxc* parameter, which is the maximum number of the community sizes, that the maximum size of a community is 50. Therefore, this natural community, although its *local modularity* is as good as the *local modularity* of the source node, is an inadequate community, since there is not a community with this size in the benchmark graph. The same situation exists for the communities with the green color. As it can be seen in the next figure, the size of the community of the node 927 and 992 is 115, and exceeds the maximum size of the communities in the benchmark graph.

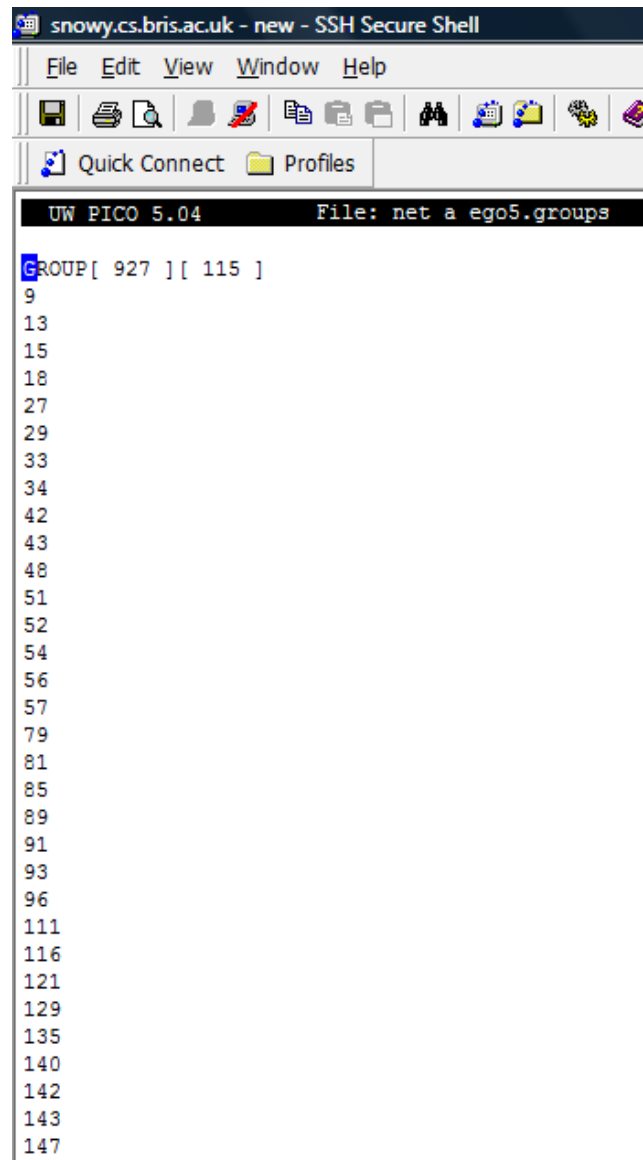
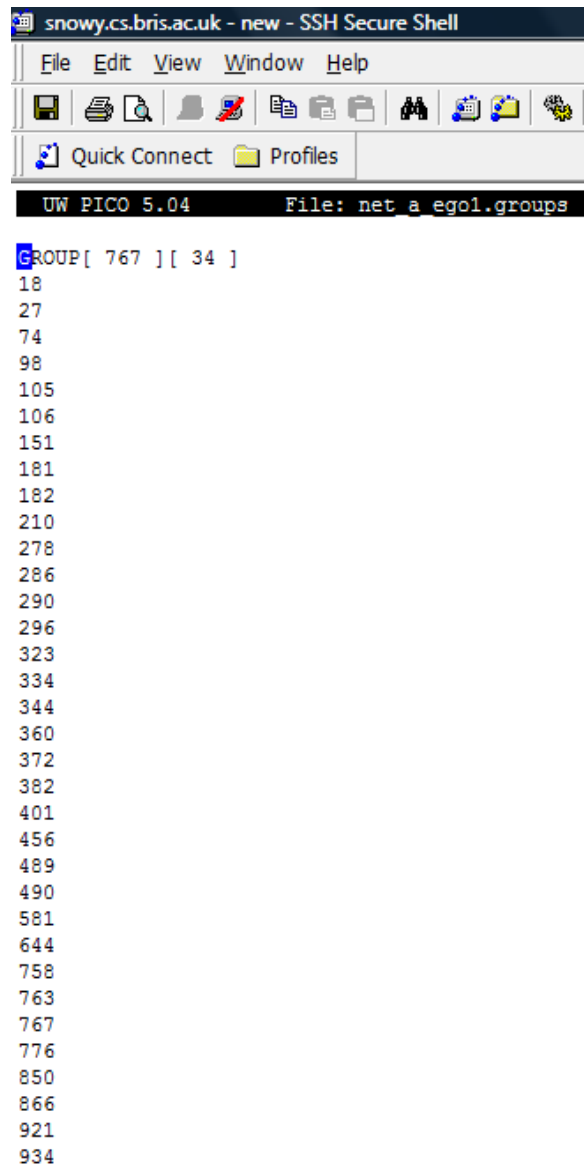


FIGURE 16

The last community which needs further investigation is the community with the black color at the Figure 12. This community is the natural community of the node 767, and its *local modularity* is 0.551971, which is lower than the value of the *local modularity* of the source node which is 0.602804. However, it is still an accepted value for the *local modularity* and therefore the community must be examined. Figure 16 shows that the natural community of the node 767 has size 34, which is an accepted size for the current benchmark graph. So, this community which includes the starting vertex 18 is the second overlapping community which node 18 belongs to.



The screenshot shows a terminal window titled "snowy.cs.bris.ac.uk - new - SSH Secure Shell". The window has a menu bar with "File", "Edit", "View", "Window", and "Help". Below the menu bar is a toolbar with various icons. At the bottom of the window, there is a status bar that reads "UW PICO 5.04" and "File: net_a_ego1.groups". The main content area of the terminal displays the following text:

```
GROUP[ 767 ][ 34 ]
18
27
74
98
105
106
151
181
182
210
278
286
290
296
323
334
344
360
372
382
401
456
489
490
581
644
758
763
767
776
850
866
921
934
```

FIGURE 17

Apparently, the above investigation denotes that the first goal is achieved. Node 18, which is the starting vertex for the new algorithm, has two overlapping communities in the benchmark graph. These two overlapping communities were successfully identified by the new algorithm. Also, our algorithm was also able to identify the other communities around the starting vertex but these communities do not fulfilled the necessary requirements in order to define them as good overlapping communities of the starting vertex. The correct performance of the algorithm will be further evaluated and tested after the definition of the criterion, in order to get more accurate conclusions.

Now that we are sure that the algorithm identifies the overlapping local communities of a node, it is time to define the quality criterion which will indicate if the found communities are as good as the source node's natural community. The benchmark

graphs of Lancichinetti and Fortunato [29] will be used again to help us define the criterion.

The first assumption is that if two (or more) communities **overlap slightly**, i.e., have few vertices in common (as the next figure shows), and they both have a **high** value of *local modularity*, then these two (or more) communities are **good** overlapping local communities for the starting vertex, and we keep both of them.

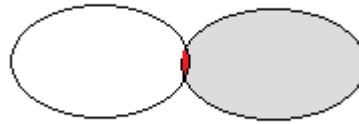


FIGURE 18

This assumption can be confirmed using the example on the benchmark graph which was created before. At the above mentioned example, the starting vertex 18 has two overlapping local communities. As it can be noticed from Figure 13 and Figure 17, which show the overlapping communities of the starting vertex, the two communities have only two vertexes in common, vertex 18 and vertex 27. Also, both of the communities have high *local modularity*. Thereupon, the assumption defined before is correct and is the first part our quality criterion.

The second assumption is that if two (or more) communities **overlap a lot**, i.e., have many vertices in common, as illustrated in the next figure, and **vary in local modularity**, then the community with the **highest** value of *local modularity* should be **selected** and the other community should be discarded.

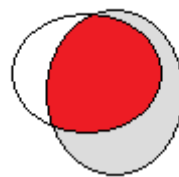


FIGURE 19

The above assumption can be confirmed by generating a new benchmark graph with 1000 nodes, 960 of them with only one membership and 40 of them with four memberships to the communities. The total number of edges is 7590. Vertex 72, which was randomly selected from the benchmark graph, is going to be the starting vertex for our algorithm. This vertex is a part of four communities in the benchmark graph. As it can be observed from the following screenshot of the file of the *local modularity* of the communities, 6 different communities were identified by the overlapping local community detection algorithm which are denoted with different colors (the red color indicates the natural community of the starting vertex).

```

2:snowy.cs.bris.ac.uk - new - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
UW PICO 5.04 File: modularity.txt
Node: 540 Local Modularity: 0.577982
Node: 939 Local Modularity: 0.557927
Node: 949 Local Modularity: 0.611554
Node: 951 Local Modularity: 0.619469
Node: 961 Local Modularity: 0.596154
Node: 966 Local Modularity: 0.557927
Node: 996 Local Modularity: 0.602339
Node: 72 Local Modularity: 0.577982

```

FIGURE 20

However, if our detection algorithm performs correctly, only four out of the six identified communities should be good overlapping communities for the starting vertex. Firstly, the natural community of the nodes 939 and 966 (blue) is excluded from the list of the candidate overlapping communities, since the size of this community exceeds the maximum community size. Figure 21 displays the aforementioned community.

```

2:snowy.cs.bris.ac.uk - new - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles
UW PICO 5.04 File: net_a_egol.groups
GROUP[ 939 ][ 79 ]
1
19
37
39
50
52
65
68
71
72
76
80
82
93
98
101
113
115
116
125
133
137
141
166
186
195
201
206
218
227
237
246

```

FIGURE 21

By implication, only five communities left for investigation. The sizes of the remaining communities are between the ranges indicated by the benchmark graph. It is notable that the natural communities of the vertexes 996 and 961 have many vertices in common. Specifically, 44 vertices are included to the both of the above communities. The files which include these communities are displayed in the next picture.

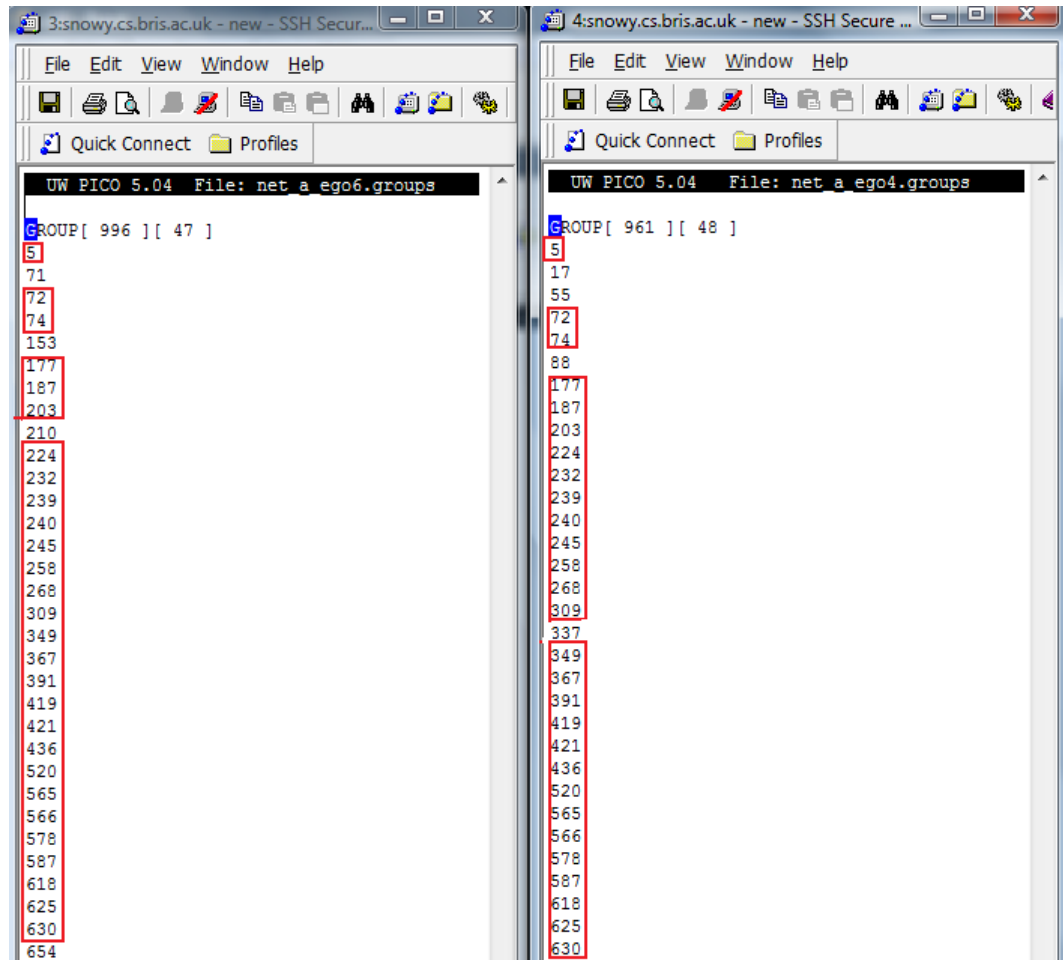


FIGURE 22

Also, the *local modularity* of the natural community of the vertex 996 is higher than the *local modularity* of the vertex 961. Therefore, the second assumption defined earlier is confirmed, since there is no reason to keep two communities which have 44 out of the 47 or 48 vertices in common. The community with the highest value of *local modularity* should be the accepted community. Thus, the second part of the quality criterion is defined, and the correctness of the created algorithm is established once again, since the 4 overlapping communities of the starting vertex which were indicated by the benchmark graph, were successfully identified by the newly created algorithm.

The third part of the quality criterion is implied from its second part. If a community is a **subset** of another community, i.e., all of the vertices of the first community are included in the second community, then the community with the **highest** value of

local modularity should be kept. The only constraint is that the community size should be between the accepted ranges for the community sizes of the graph.

3.4 SUMMARY

This chapter was dedicated to the detailed description of the construction of overlapping local community detection algorithm. The step-by-step procedure for the creation of the algorithm was described, and then the quality criterion was defined in order to decide which of the identified communities were as good communities as the natural community of the source vertex. The three parts of the criterion were established through the testing of the algorithm using benchmark graphs.

4. EXPERIMENTAL RESULTS

4.1 INTRODUCTION

Although, we applied the algorithm to computer-generated benchmark graphs at the previous chapter, is crucial to test the overlapping local community detection algorithm, using real-life networks. For this purpose we are going to use three famous real-life networks: the Zachary Karate Club, the Social Network of Dolphins and the books about US politics network.

4.2 THE ZACHARY KARATE CLUB

The Zachary Karate Club has been widely used as a benchmark by researchers who aimed to test their community detection algorithms. It is a social network of friendships between 34 people in the same karate club, at a US university in the 1970. Many community detection algorithms divide the network into two main communities, while other algorithms [20, 12] detect four smaller communities with higher values of modularity. Figure 22 shows the two main communities of the Zachary Karate Club at the left, and the real decomposition of the network to the four communities at the right (copied from [12]).

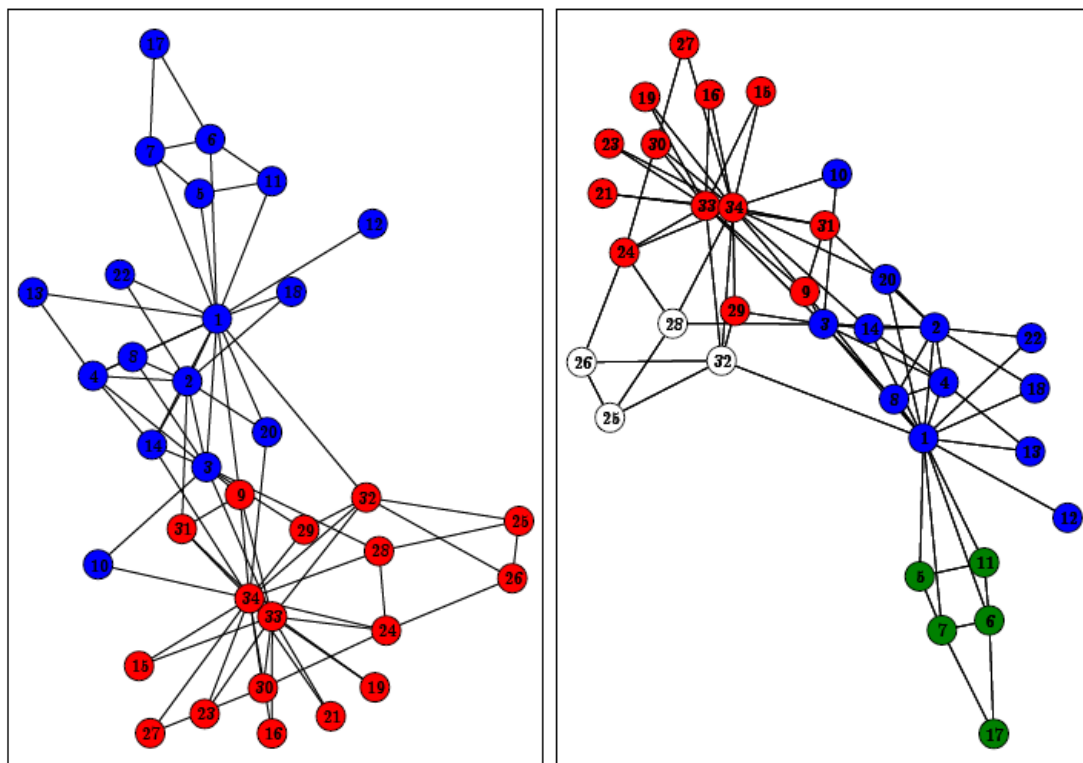


FIGURE 23

Obviously, overlapping nodes exist between the communities. These nodes (3, 9, 10, 14, 31) are often misclassified by many algorithms. Thereupon, it is worth to test our algorithm using the Zachary Karate Club in order to evaluate both the correctness of the identified partitions and the quality criterion which was defined before. Since node 3 seems to be the node which overlaps with the highest belonging coefficient, we select this node as the starting vertex for our algorithm. The file with the *local modularity* for the identified communities, along with the natural community of node 3 are displayed in the next figure.

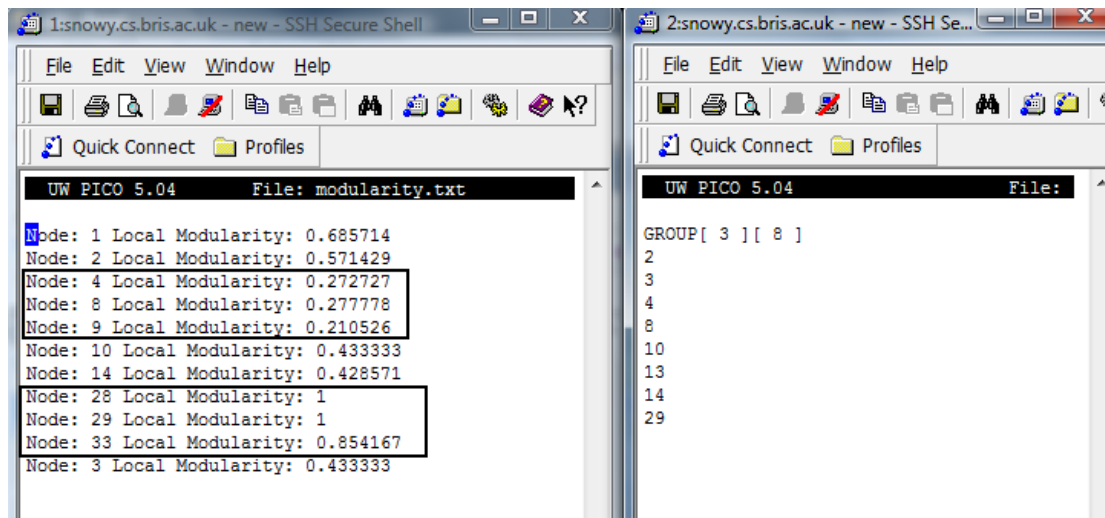


FIGURE 24

The natural communities of the nodes with the higher values of *local modularity*, along with the communities with the lower values, are excluded from the candidate overlapping communities of the starting vertex. This is due to the fact that the communities with *local modularity* equal to 1 or 0.854167 (nodes 28, 29, 33) include the whole network (or almost the whole network for node 33), and also the communities with the lower values of *local modularity* include a very small number of nodes, as indicated by the next figure.

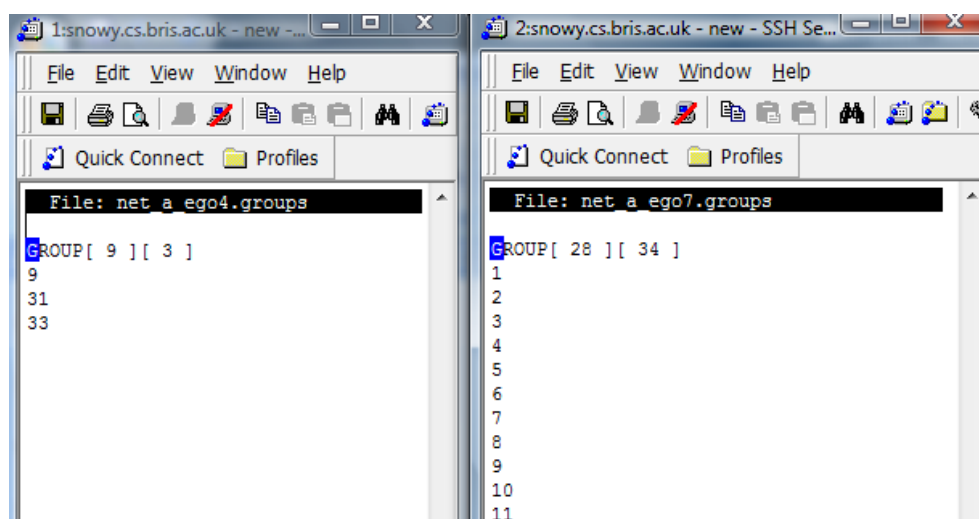


FIGURE 25

According to our quality criterion, if two communities overlap a lot, then we should keep the community which has the highest modularity and its size is between the community sizes boundaries of the network. The algorithm which was introduced by Clauset, and which we used as the natural community detection algorithm in our algorithm, detects the following natural community for the starting vertex.

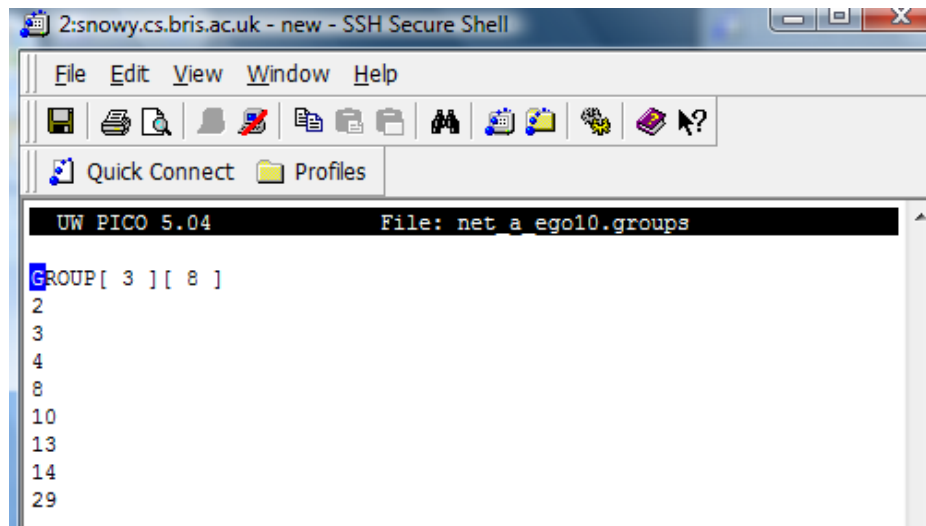


FIGURE 26

However, it is obvious from Figure 23 (right), that the natural community of node 3 includes a bigger number of nodes than the community identified by Clauset. If we investigate the other communities which were identified by our algorithm, and if we use our quality criterion, we conclude that the overlapping local community detection algorithm performs correctly, since the natural community of node 3, which is displayed at the right part of Figure 23, is identified by the algorithm. The following figure proves the above statement.

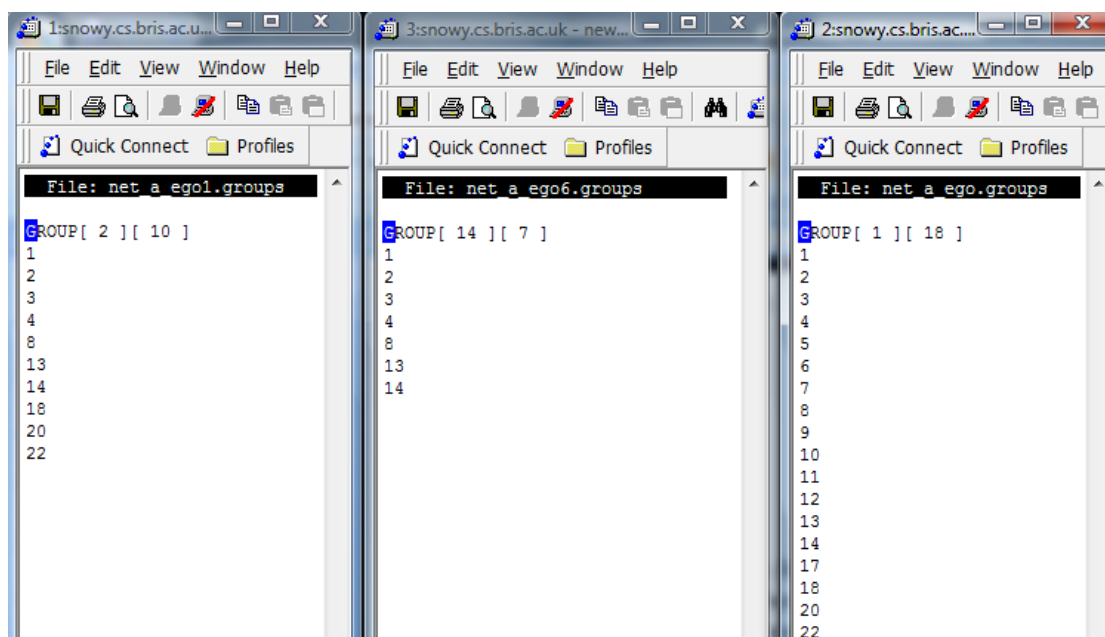


FIGURE 27

Apparently, the four communities displayed above (Figure 26-27) have a lot of vertices in common. So, which of them is a good community for the starting vertex? The above question can be answered using our quality criterion. The natural community of node 1 has size 18 which is a relatively high value for a community size in a network with total number of vertices equal to 34. Also, this community is the community identified by many traditional algorithms which divide the Zachary Karate network into only two partitions (Figure 23 left). So, due to the fact that the size of the community is relatively big, as the quality criterion indicates, this community is not an adequate one.

Thereafter, only three available communities remained for investigation. It is obvious that the three remaining communities highly overlap. Therefore, according the criterion which we defined earlier (*if two communities **overlap a lot**, and **vary in local modularity**, then the community with the **highest** value of local modularity should be **selected** and the other community should be discarded*), we should keep the community with the highest modularity. This community is the natural community of node 2. Surprisingly, the community identified by our algorithm and by the quality criterion is one of the four actual communities which were identified by [20, 12] and displayed in Figure 23 (right).

4.3 THE SOCIAL NETWORK OF DOLPHINS

Social networks exist not only between humans but also between animals. One of the most popular social networks of animals is the social network of 62 dolphins, in a community living in Doubtful Sound, New Zealand. This network was described in [30]. The dolphins are linked in the above network if they observed swimming together. Figure 28 displays the above social network (copied from [31]) and Figure 29 displays the partition of this network into four sub communities (copied from [12]).

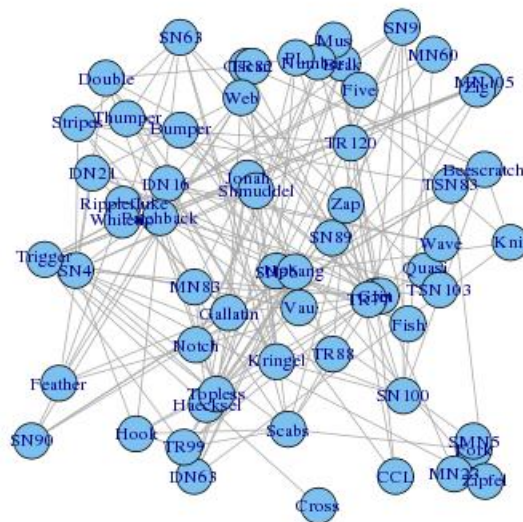


FIGURE 28

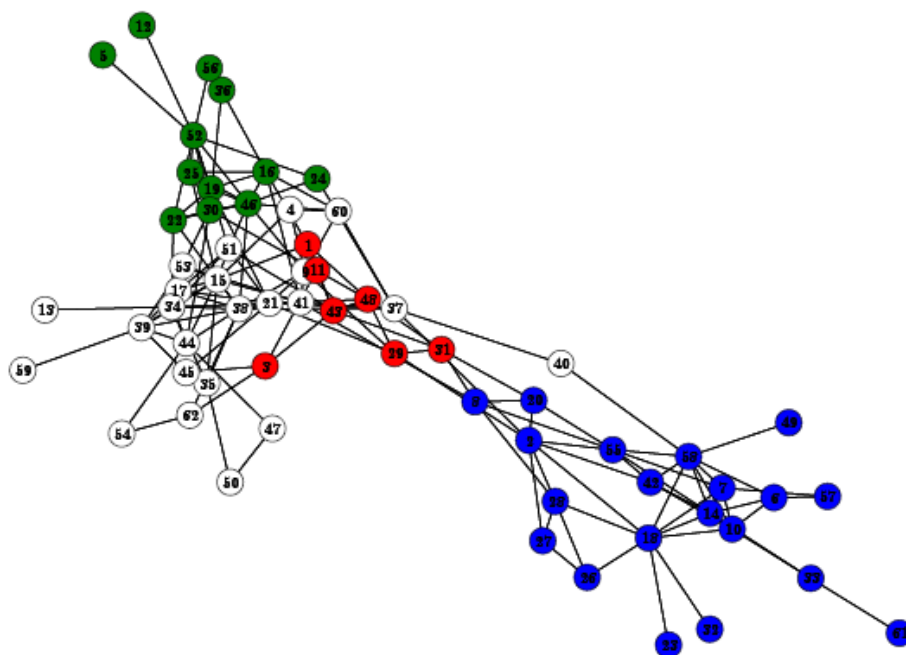


FIGURE 29

As it can be observed from the above picture, two main communities exist, the communities with the blue and white nodes. However, these communities are further divided to the sub communities with the red and green nodes. The two communities with the white and green colors represent the female dolphins and the communities with the red and blue color represent the male dolphins [12].

For testing purposes we select a node which belongs to the red community of the male dolphins, to confirm that the algorithm and the quality criterion successfully classify the node to the correct community and also find the node's overlapping communities. The starting vertex for the overlapping local community detection algorithm will be node 31. As it can be observed from the next figure the natural community which is identified by Clauset's algorithm does not have the highest value of *local modularity*.

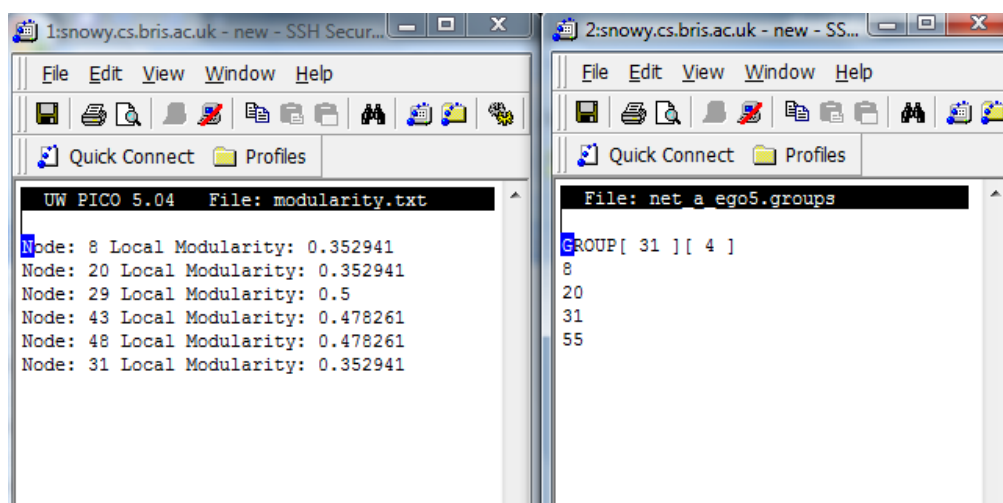


FIGURE 30

Also, the identified natural community of the source vertex does not correspond to the red community of Figure 29. So, we have to use the quality criterion on the identified overlapping communities of the source vertex and find the best one. The two overlapping communities which were identified by our algorithm are displayed in the next figure.

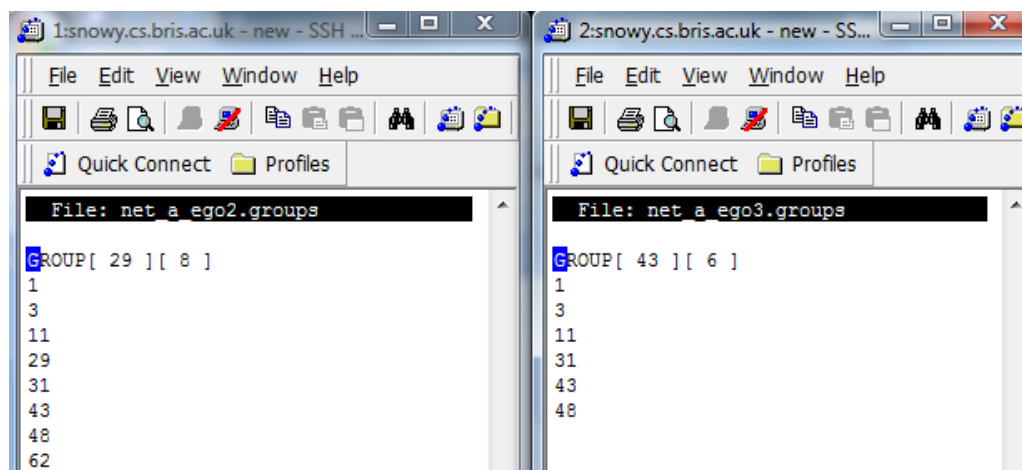


FIGURE 31

As it can be noticed, the two communities have many vertices in common, and in particular, the natural community of the node 43 is a subset of the natural community of the node 29. Therefore, according to the third part of our quality criterion, (*if a community is a **subset** of another community, then the community with the **highest** value of **local modularity** should be kept*), we should keep the community with the highest *local modularity*, which is the natural community of node 29. This community is also one of the actual sub communities (red) which the social network of dolphins is divided to.

4.4 BOOKS ABOUT US POLITICS

The political books network or the ‘PolBooks’ network is a complex information network. The nodes of this network represent political books sold by Amazon, and the linkage of two books indicates that a book has been purchased in combination with another book of the same topic. Another characteristic of the network is that each book has a label indicating if the book belongs to the ‘neutral’, ‘liberal’ or ‘conservative’ category, according to the users’ reviews. Although the common expectation about the partitions of the network is to have three main communities, most of the community detection algorithms divide the network into two main categories, the ‘liberal’ and the ‘conservative’ category, and to further sub-communities which are parts of the two main communities. The books which are labeled as ‘neutral’ are placed in the middle of the two communities and they belong either to the ‘liberal’ category or to the ‘conservative’ [12]. The ‘PolBooks’ network has 105 vertices and 882 edges, and is displayed in Figure 32.

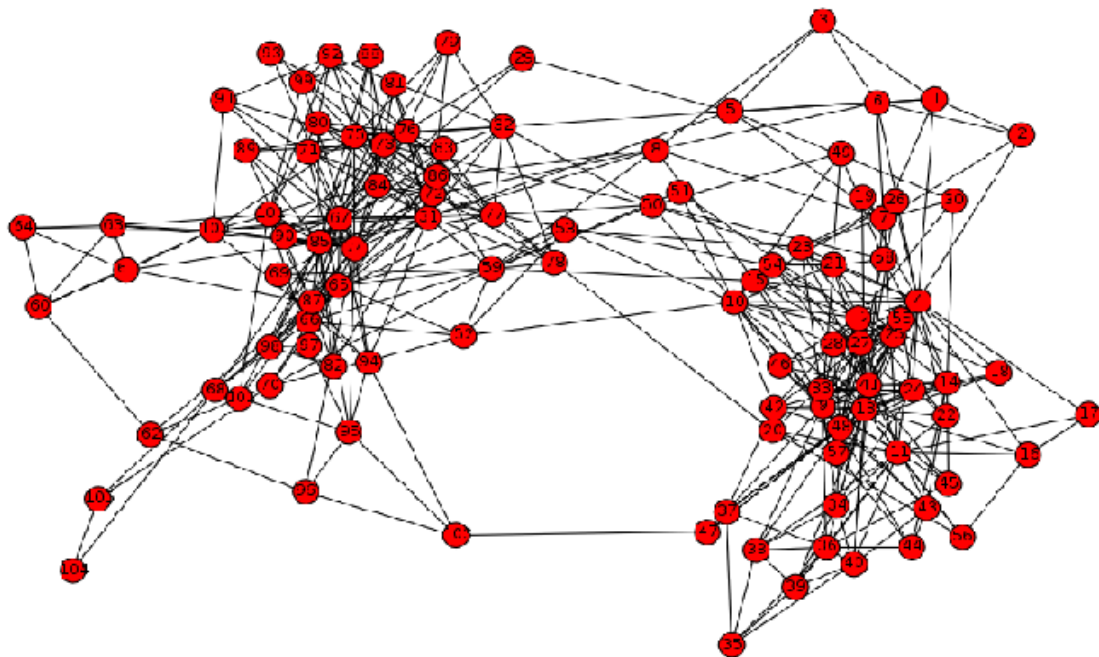


FIGURE 32

The right part of Figure 32 displays the community which includes the books labeled as ‘conservative’ and the left part of the figure displays the community of the books which are labeled as ‘liberal’.

To check how good our algorithm performs on bigger and more complex networks, we are going to use the ‘PolBooks’ network as a benchmark to evaluate the algorithm’s performance. The starting vertex should be a book which is labeled as ‘neutral’ in order to confirm the ‘neutral’ books do not compose a different community but they are included in the two major communities of the network. Node 29 is a ‘neutral’ book called ‘All the Shah's Men’ and as it can be seen from Figure 32, does not belong to the two tightly connected communities but it is somewhere in the middle.

The next figure shows the natural community of node 29 along with the file which contains the values of the *local modularity* for node’s neighbor communities.

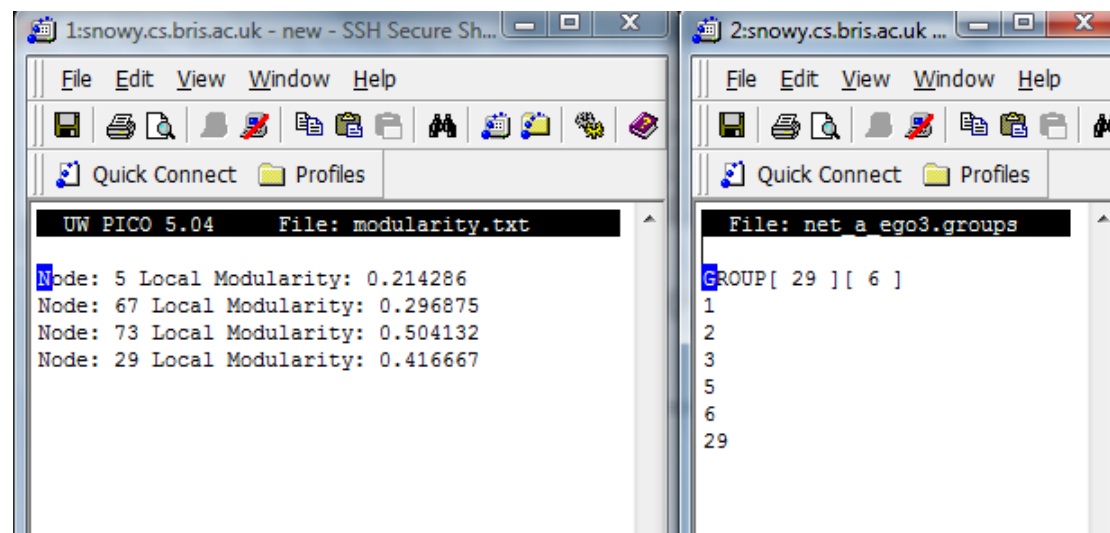


FIGURE 33

The first two communities are not subjects for investigation since the first one contains only three nodes and the second one has a low value of *local modularity*. Therefore the quality criterion is going to be used only on the two remaining communities: the natural community of node 29, and the natural community of node 73. The natural community of node 73 is displayed in the following figure.

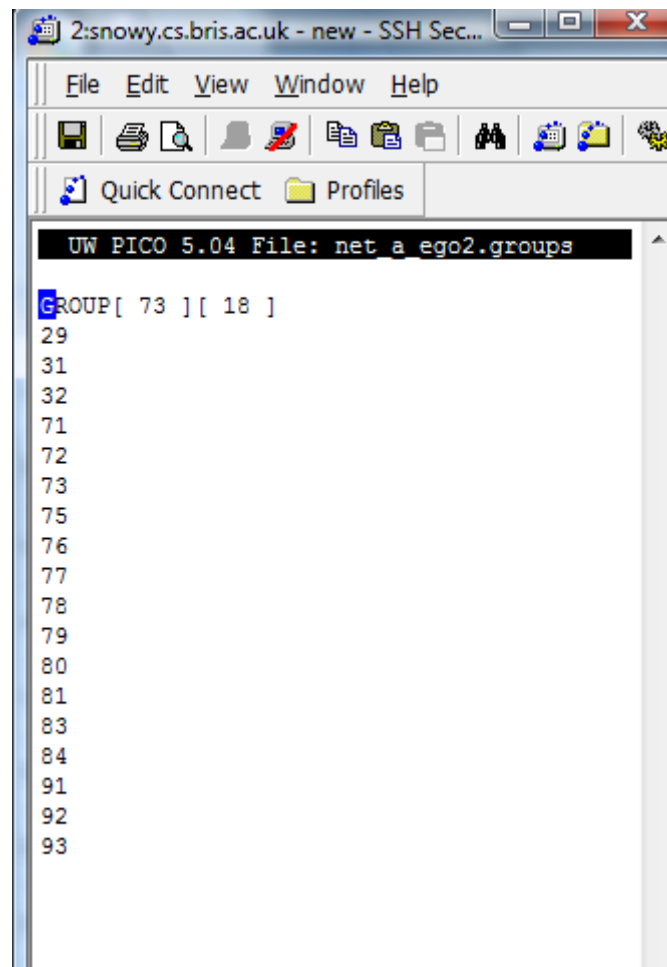


FIGURE 34

As it can be noticed from the two natural communities displayed above, node 29 belongs to two different communities which have in common only our source node. Also, the values of *local modularity* for both of the communities do not vary a lot. Therefore, as stated in the quality criterion, *(if two (or more) communities **overlap slightly**, and they both have a **high** value of **local modularity**, then these two (or more) communities are **good** overlapping local communities for the starting vertex, and we keep both of them)*, we should keep both of the identified communities.

Furthermore, it is notable that the identified overlapping communities are sub communities of the two major communities that exist in the ‘PolBook’ network, the ‘liberal’ community and the ‘conservative’ community. This can be proved if the file with the labels of the nodes is examined. All of the nodes of the natural community of node 73 have labels ‘liberal’ or ‘neutral’, while all of the nodes of the natural community of node 29 have labels ‘conservative’ or ‘neutral’. The above fact complies with the two partitions which the majority of the community detection algorithms identify.

4.5 SUMMARY

In this chapter we described the behavior of the overlapping local community detection algorithm when it has to detect the overlapping communities of a vertex which belongs to a real-life complex network. The networks which were used as a benchmark were the Zackary karate club network, the social network of dolphins and the books about US politics network. Our quality criterion was used in order to decide the adequate overlapping communities for the starting vertex, and as it was described before, the identified communities coincide with the communities which other community detection algorithms identified for these networks.

5. PROJECT EVALUATION

5.1 INTRODUCTION

The purpose of this chapter is to offer a critical evaluation of the project. Moreover, it analyses the extent to which the project has achieved its objectives, and also criticizes the choices which were made.

5.2 ACHIEVEMENT OF AIMS AND OBJECTIVES

The first objective of the project was to investigate thoroughly the existing algorithms for community detection, in order to find an algorithm which detects the natural community of a given vertex. The constraint which narrowed down the number of algorithms for investigation was that the algorithm had to be independent from the global parameters of the network, and to discover the natural community of the vertex locally, i.e., exploring the network one vertex at a time. Since the detection of communities in complex networks has attracted a great deal of interest, many algorithms exist which fulfill the above mentioned requirements. After a lot of research, we concluded to a single algorithm which was introduced by Aaron Clauset [7] and which behaved according to our needs. However, many other algorithms exist which might have better performance, but this is an issue for later discussion.

The second objective which was successfully achieved was the creation of an algorithm which detects the overlapping local communities of a starting vertex. In order to achieve that, the algorithm which was chosen before had to be extended to identify not only the natural community of the source vertex but also the communities of all of the neighbors of the vertex. Nevertheless, not all of the identified communities included our source vertex, so an examination of the detected communities had to take place in order to exclude the irrelevant communities.

Although the overlapping local communities of the source node were identified, not all of them were good overlapping communities for the node. Therefore, the objective was to create a quality criterion in order to classify these communities to adequate and inadequate communities for the node. Though, the establishment of such a criterion required the testing of the criterion's assumptions with benchmark graphs, to check their correctness.

Finally, since both the overlapping local community detection algorithm and the quality criterion were created, the only thing left to achieve was to test their performance on real-life complex networks. This achievement was realized using three well-known complex networks, the Zachary karate club, the social network of dolphins and the books about US politics network.

As it can be observed, all of the project objectives were achieved. Of course, many alternative choices exist and further improvements have to be made. These issues are going to be discussed in the following chapters.

5.3 EVALUATION OF PROJECT'S MODULES

This section is dedicated to the discussion and evaluation of the choices which were made during the project. We start the evaluation procedure with the choice of the algorithm which identifies the natural community of a source vertex. Clauset's algorithm was a fair solution since the maximization of *local modularity* provides good results for community detection. However, other choices which detect the natural community of a node faster, exist, and are worthwhile to compare them with the existing algorithm. Furthermore, Clauset's algorithm does not always identify the optimum community for the node. The identified community is sometimes either too small, or includes the whole network. This is due to the algorithm's 'stopping criterion' and requires further investigation to have the optimum solution.

A significant part of the project was dedicated to the development of the overlapping local community algorithm. The approach which was followed, that is to discover the natural communities of the neighbors of the source vertex, finds the most important overlapping local communities for the vertex. Nevertheless, communities which include the source vertex but do not start from a vertex's neighbor exist. So, even if the identified communities provide good results, there is no harm to check the results provided if we include to the procedure the communities which do not start from the source vertex's neighborhood.

Another decision we made, was the quality criterion which identifies the good overlapping communities for the node. Through the testing of the algorithm both with benchmark graphs and with real-life networks, we observed that the quality criterion chose the correct communities, as the graphs indicated. Of course, there is always room for improvements and further specifications which are going to be discussed at the following chapters.

Additionally, with a view to establish the quality criterion we use the benchmark graphs which were introduced in [29]. The decision to use the specific benchmark graphs was good due to the fact that they provide you with the ability to parameterize them according to your needs. Also, when the need to test both the created algorithm and the quality criterion emerged, the decision to use the Zachary karate club, the social network of dolphins and the books about US politics network was a proper one, since these networks are widely used by researchers who want to test their community detection algorithms. Yet, many other real-life and computer-generated can be used for further testing and evaluation of the algorithm.

5.4 SUMMARY

At this chapter all the achieved aims and objectives were discussed. Also the chapter provided a critical evaluation for each module of the project along with further ideas for improvement which are going to be discussed in the following chapter.

6. FUTURE WORK

Although all the defined aims and objectives were in general achieved, there is always room for improvements and extensions of the current work in order to achieve better performance and functionality.

First, Chapter 5 indicates that the decision of the algorithm for natural community detection provided fair results, but other algorithms exist which perform better. An interesting extension of the project would be to replace the current algorithm for natural community detection with another algorithm, and compare the results of these approaches to conclude which algorithm is the most appropriate.

Next to first, the newly-created algorithm is based on the assumption that the overlapping communities of a source vertex should start from a source vertex's neighbors. Also, as mentioned before, the goal of the algorithm is not to identify all of the overlapping communities of a source vertex, but only these overlapping communities which are considered as good as the vertex's natural community. However, the fact that a natural community of a node which is not in the neighborhood of the source vertex, might also include the vertex, should not be ignored. Thus, an interesting improvement of the algorithm would be to take into consideration the natural communities of the nodes outside the neighborhood of the source vertex.

A further extension of this project could be the more detailed definition of the quality criterion. So far, the quality criterion covers the cases of the communities which overlap a lot, or overlap slightly. Sometimes, none of the two extreme cases stands. So, the extension of the quality criterion to cover the intermediate cases would be a very useful.

At last, testing is very important for the evaluation and the detection of flaws in the performance of the algorithm. Thereupon, it is vital to apply our overlapping local community detection algorithm to even more computer-generated and real-life networks which have a known community structure. Also, it would be useful to compare the performance of our algorithm with other community detection algorithms.

7. CONCLUSION

The aim of this project was to develop a novel algorithm for overlapping local community detection. Overlapping communities are an important aspect of complex networks and therefore, an algorithm which detect the overlapping communities of a node, without any global knowledge of the graphs topology consist a novel contribution.

In order to create the aforementioned algorithm, a research had to be made on the related work which was already done in the area of community detection algorithms. Chapter 2 presented the most important findings in this area.

At Chapter 3, the project design was discussed. Firstly, the problem specification was given and then, all of the carried out work was discussed thoroughly. This chapter is divided into two main sections: the developing of the overlapping local community algorithm and the creation of the quality criterion for the community evaluation.

Chapter 4 presented the performance of the algorithm when it was applied to real-life networks. The step-by-step execution of the algorithm was displayed, and then the results were examined in comparison with the actual results which the real-life networks indicate.

Community detection in complex networks has attracted a great deal of interest, since communities uncover important information about the relationships between the nodes of the network. The most important outcome of this project is the developing of an algorithm for overlapping local community detection which is an issue of great importance in the area. Generally speaking the project achieved its main goals, but there is still space for improvements and extensions.

8. BIBLIOGRAPHY

- [1] M. Faloutsos, P. Faloutsos, and C. Faloutsos, *Comput. Commun. Rev.* **29**, 251 (1999).
- [2] J. M. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins, “The Web as a graph: measurements, models and methods,” in *Proceedings of the International Conference on Combinatorics and Computing*, No. 1627 in Lecture Notes in Computer Science (Springer-Verlag, Berlin, 1999), pp. 1–18.
- [3] S. Wasserman and K. Faust, *Social Network Analysis* (Cambridge University Press, Cambridge, 1994).
- [4] D. J. de S. Price, *Science* **149**, 510 (1965).
- [5] S. Redner, e-print physics/0407137.
- [6] T. Ito, T. Chiba, R. Ozawa, M. Yoshida, M. Hattori, and Y. Sakaki, *Proc. Natl. Acad. Sci. U.S.A.* **98**, 4569 (2001).
- [7] Clauset, A., 2005, *Phys. Rev. E* 72(2), 026132.
- [8] Wang, Xutao; Chen, Guanrong; Lu, Hongtao, A very fast algorithm for detecting community structures in complex networks, *Physica A*, Volume 384, Issue 2, p. 667-674.
- [9] S. Fortunato, arXiv:0906.0612 at www.arXiv.org.
- [10] Girvan, M., and M. E. J. Newman, 2002, *Proc. Natl. Acad. Sci. USA* 99(12), 7821.
- [11] Newman, M. E. J., and M. Girvan, 2004, *Phys. Rev. E* 69(2), 026113.
- [12] Nicosia V, Mangioni G, Carchiolo V and Malgeri M, 2009 *J. Stat. Mech.* P03024
- [13] D. Romero, J. Kleinberg, The Directed Closure Process in Hybrid Social-Information Networks, with an Analysis of Link Formation on Twitter. *Proc. 4th International AAAI Conference on Weblogs and Social Media*, 2010.
- [14] Kim, Youngdo, Son, Seung-Woo, Jeong, Hawoong, Community Identification in Directed Networks, *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, Volume 5. ISBN 978-3-642-02468-9, Springer Berlin Heidelberg, 2009, p. 2050.
- [15] R. Guimera, M. Sales-Pardo, L. A. N. Amaral, *Phys. Rev. E* 76, 036102 (2007).
- [16] Leicht, E. A., and M. E. J. Newman, 2008, *Phys. Rev. Lett.* 100(11), 118703.

- [17] Brandes, U., D. Dellling, M. Gaertler, R. G rke, M. Hoefer, Z. Nikolski, and D. Wagner, 2006, URL <http://digbib.ubka.uni-karlsruhe.de/volltexte/documents/3255>.
- [18] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Science* **220**, 671 (1983).
- [19] M. E. J. Newman, *Phys. Rev. E* **69**, 066133 (2004).
- [20] J. Duch and A. Arenas, *Phys. Rev. E* **72**, 027104 (2005).
- [21] M. E. J. Newman, *Proc. Natl. Acad. Sci. U.S.A.* **103**, 8577 (2006).
- [22] L. Danon, A. D  az-Guilera, J. Duch, and A. Arenas, *J. Stat. Mech.: Theory Exp.* (2005) P09008.
- [23] M. E. J. Newman, *Phys. Rev. E* **74**, 036104 (2006).
- [24] Boettcher, S., and A. G. Percus, 2001, *Phys. Rev. Lett.* **86**, 5211.
- [25] A. Arenas, J. Duch, A. Fern  ndez, and S. Go  mez, *New J. Phys.* **9**, 176 (2007).
- [26] Bagrow, J. P., and E. M. Bollt, 2005, *Phys. Rev. E* **72**(4), 046108.
- [27] Papadopoulos, S., A. Skusa, A. Vakali, Y. Kompatsiaris, and N. Wagner, 2009, eprint arXiv:0902.0871.
- [28] Lancichinetti A, Fortunato S and Kert  sz J 2009 Detecting the overlapping and hierarchical community structure in complex networks *New J. Phys.* **11** 033015
- [29] A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E*, 80(1):016118, 2009.
- [30] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, *Behavioral Ecology and Sociobiology* **54**, 396-405 (2003).
- [31] <http://networkdata.ics.uci.edu/index.php>

APPENDIX

PARTS OF THE SOURCE CODE

This appendix includes the most important parts of the source code, of the project which is implemented in C++. Due to the lack of space, other parts of the code are omitted.

DETECTION OF THE NATURAL COMMUNITY OF THE SOURCE VERTEX

```
for (t=1; t<ioparm.tf; t++)
{
    // find best join (store in dRmax)
    dRmax = findMaxDeltaR();
    if (dRmax.value < -60000.0)
    {
        cout << ">> WARNING: entire component
        found; stopping.\n"; showPairsList();
        ioparm.tf = t;
        break;
    }

    // adjust the boundary
    adjustBoundary();
```

```
    // track maximum R value
    Rmax = updateRStats(Rmax);

    // write stuff to disk
    writeToJoinsFile(false);
    // write stuff to the terminal
    writeToTerminal(false);
}

// find strongest association (via  $d(dR/dt)/dt$  --
// second derivative of  $R[t]$ )
double ddR = 0;
double ddRmin = 101010101010101.0;
for (t=1; t<ioparm.tf-1; t++)
{
    ddR = dR[t+1] - dR[t];    // appx 2nd derivative
    if (ddR < ddRmin)
    {
        ddRmin = ddR;
        tmax = t+1;
        rmax = R[tmax].y;
    }
}
```

IDENTIFICATION OF THE NEIGHBORS OF THE SOURCE VERTEX WHICH THEIR NATURAL COMMUNITY IS NOT YET DISCOVERED

```
for (t=1; t<=ioparm.tf; t++)
{
    curre = &e[t];
    while (curre != NULL)
    {

        if(v[t].index != myVertex && (v[curre-
>si].index == myVertex || v[curre->so].index ==
myVertex))
        {
            if(finished[t] != 1)
            {
                cout<<"-----New Starting
Vertex: "<<v[t].index<<endl;
                gparm.v = v[t].index;
                finished[t] = 1;
                exit = true;
                break;
            }
        }
        curre = curre->next;
    }
    if(exit == true)
    {
        exit = false;
        exitagain = true;
        break;
    }
}
```

CALCULATION OF THE CHANGE IN LOCAL MODULARITY AS A RESULT OF JOINING A VERTEX TO THE COMMUNITY

```
tuple computeDeltaRj(const int j)
{
    tuple dRj;
    edge *currente;
    list *current, *test;
    double x = 0.0;
    double y = 0.0;
    double l1 = 0.0;
    double l3 = 0.0;
    currente = &e[j];
    while (currente != NULL)
    {
        if (v[currente->si].loc == L_BOUNDARY)
        { x+=1.0; } // touch B
        else { y+=1.0; } // do not touch B
        currente = currente->next;
    }
    dRj.x = x;
    dRj.y = y;
    current = v[j].PL;
    while (current != NULL)
    {
        currente = &e[current->index];
        while (currente != NULL)
        {
            if (v[currente->si].sBL == 1)
            {
                if (v[currente->si].BL->index
                    == j)
                { l1 += 1.0; }
            }
        }
    }
}
```



```

    }
    else if (v[currente->si].loc == L_INTERIOR)
    { l3 += 1.0; }
    currente = currente->next;
}
current = current->next;
}
dRj.Z = l1*0.5 + l3;
if (y==0) { dRj.Z += x; }

dRj.value = ((x - dRj.Z) - (R[t-1].y * (y - dRj.Z))) /
(1.0 + To[t-1] + (y - dRj.Z));
    return dRj;
}

```

DETECTION OF THE COMMUNITIES WHICH INCLUDE THE SOURCE VERTEX

```

void getFiles(char *s, int p)
{
    ifstream myReadFile;
    int fin = 0;
    char output[100];
    int start = 0;
    int cnt = 0;

    char str_val [32] ;
    snprintf (str_val, sizeof (str_val), "%d",
myVertex) ;

```

```

myReadFile.open(s);
if (!myReadFile)
{
    cerr << "Unable to open file datafile.txt
GET FILES"<<s<<endl;
    exit(1);    // call system to stop
}
if (myReadFile.is_open())
{
    while (!myReadFile.eof())
    {
        if(fin == 1)
            break;
        myReadFile >> output;
        if(strcmp(output, "]") == 0)
        {
            start = 1;
            cnt++;

        }
        if(start == 1)
        {
            if(strcmp(output, str_val) ==
0)
            {
                compare[comparecnt++] =
p;
                fin = 1;
            }
        }
    }
}
myReadFile.close();

```