Chapter 4

Fuzzy Agglomerative Community Detection

This chapter studies belongingness of nodes to different communities. Extends the Reachability property introduced in chapter 3 for defining belongingness of nodes (membership degree), and incorporates it into community detection process to identify both disjoint and overlapping communities.

4.1 Introduction

In real-world scenarios, nodes exhibit degree of belongingness to different communities rather than having membership of single community. Identification of disjoint communities is not sufficient to meet the realities involving partial membership of nodes. Therefore, community detection algorithms not only have to sense network structures but also quantitative affiliations to multiple communities. Fuzzy community detection has been

introduced to measure the belongingness of nodes in different communities i.e. membership degrees. In contrast to disjoint community detection, fuzzy community detection not only senses qualitative affiliations to communities but also network structure [222]. Some fuzzy community detection methods have been proposed in recent years. However, many of them require prior information about communities (e.g. number of communities) [63, 186, 206], which may degrade the accuracy of communities.

In this chapter, overlapping aspect of communities is explored in addition to disjoint communities. A fuzzy agglomerative community detection algorithm (FuzAg) is designed to identify both disjoint communities and overlapping communities. Proposed FuzAg algorithm is free from giving number of communities in prior. The main contributions of the chapter are as follows:

- The concept of *Reachability* introduced in chapter 3 is extended further to uncover overlapping communities. Core nodes of communities are defined following the same principle as Reachability. Membership degrees of nodes to different communities i.e. the belongingness of nodes to different communities is defined involving the core nodes.

- Further exploring the first objective, equal opportunity is given to every node to form its own community. The notion of self-membership is introduced for this purpose.

- Defined anchors (see in subsection 4.2.1) using self-membership degree. Communities grow in reference to the anchors they are associated with.

- FuzAg algorithm is designed, where community identification follows agglomeration i.e. communities grow by accumulating new members in successive steps. Complexity of the algorithm is shown to be $O(n^2)$.

- Empirical results on real-world networks as well as on synthetic networks show the competency of FuzAg in identifying both disjoint and overlapping communities. Disjoint communities resulted by FuzAg algorithm are more accurate. Quality of overlapping communities are also better than other algorithms.

4.2 Proposed Approach

A complex network is represented as an undirected graph $G(V,E,W)$, where V is a set of n vertices, E is a set of m edges, and W is an $n \times n$ edge weight (or adjacency) matrix. Any element w_{ij} in W denotes the weight of the edge connecting node i and j. Community detection in networks is a process of finding $k \times n$ partition matrix U, where k is the number of communities and n is the number of nodes in the network. Each element u_{ji} in U defines the membership degree of node i in community j. Another row is added to the partition matrix U for defining self-membership degree for all nodes. Thus, $k+1$ rows and n columns are present in the extended partition matrix, which is referred as U^+. The $(k+1)^{th}$ row of U^+ is always specified for self-membership degree of all nodes. The elements of extended partition matrix U^+ has to satisfy the following constraints:

$$0 \leq u_{ji} \leq 1 \text{ and } \sum_{j=1}^{k+1} u_{ji} = 1, \forall i, j. \tag{4.1}$$

Each community is associated with an anchor node and the community grows with reference to that node. The agglomeration process starts with an arbitrary node as an anchor node and returns the final partition matrix U that contains membership degree of nodes to k communities associated with each of the k anchors. Note that the process returns U, not the U^+ i.e. self-membership degree has been omitted. Both disjoint and fuzzy (overlapping) partitioning are considered. For disjoint partitioning, the community label of node i

is assigned as follows:

$$L_i = \underset{j}{\operatorname{argmax}}\, u_{ji}, \forall u_{ji} \in U. \tag{4.2}$$

In the case of any conflict, i.e. if membership degree of any node is same for multiple communities, the node is assigned arbitrarily to one of the conflicting community. For fuzzy partitioning, the community label of node i is assigned as follows:

$$L_i = \bigcup j \text{ if } u_{ji} > \psi, \forall u_{ji} \in U \tag{4.3}$$

where, ψ is the minimum threshold value to acquire membership of any community. The community label assignment done with the Equation 4.2 and Equation 4.3 actually hardens the partitioning [79] to form disjoint communities using maximum membership rule and crisp overlapping communities using strict $\alpha-$cut respectively.

4.2.1 New Anchor Selection

The first anchor is selected arbitrarily. Thus, in the first iteration, one anchor is there so two rows are present in the extended partition matrix U^+, where 1^{st} row for membership degree of all nodes to the community associated with the anchor and $(k+1)^{th}$ row i.e. 2^{nd} row for self-membership degree. In the successive iterations, the row for self-membership degree in U^+ is considered to identify new anchors. As mentioned earlier, the last row of U^+ i.e. $(k+1)^{th}$ row is specified for self-membership degree. The nodes having non-zero self-membership degrees are considered as probable anchors. Any node can become an anchor if the node has sufficient self-membership degree. An anchor node is defined with $(k+1)^{th}$ row of U^+ as follows.

Definition 4.1 (Anchor)**.** Any node i will be an anchor if self-membership degree of that node has at least the value equal to total membership degrees to existing communities

associated with different anchors, i. e. any node i to become an anchor it should satisfy the following condition:

$$u_{(k+1)i} \geq \sum_{j=1}^{k} u_{ji}. \tag{4.4}$$

For all the newly selected anchors, a row is specified in U^+ to maintain the membership degrees of all nodes (including the anchor itself) to the community associated with the anchor.

4.2.2 Membership degree computation

The notion of mutual interest introduced in chapter 1 is explored further to define membership function. In the context of mutual interest, the relationship between two persons is treated as two way personal interests i.e. forward interest and backward interest. From the view point of community detection, forward interest is considered for a community to identify its members but the ultimate decision is up to the identified members whether they wish to be part of that community or not i.e. the backward interest. To decide whether a node belongs to a particular community, reachability of that node to the community is computed. Thus, reachability measures the backward interest of a node to any community. The reachability of a member node i to the community C is computed as follows:

$$\text{Reachability}(i,C) = \frac{\sum\limits_{j \in (C \cap N_i)} w_{ij}}{\sum\limits_{j \in N_i} w_{ij}} \tag{4.5}$$

where, N_i is the set of neighbors of node i and w_{ij} is the value of i^{th} row and j^{th} column in the weight matrix W. To compute membership degree of a node, core nodes for the

community associated with an anchor is defined first. Then membership degree of the node is computed using the concept of Reachability and core nodes.

Definition 4.2 (Core Nodes)**.** For an anchor h and extended partition matrix U^+, the core nodes X_h are defined as follows. First, the anchor h and its neighbors are considered as part of core nodes if they have non-zero membership degree. Second, the nodes that are not neighbors of anchor to become part of core nodes, they should have at least non-zero membership degree equal to the total membership degrees associated with the remaining anchors.

These core nodes are more likely to be the members of the community they represent. Membership degree of node i to the community associated with the anchor h is computed as follows:

$$\text{Membership}(i,h) = \frac{\sum\limits_{j \in (X_h \cap N_i)} w_{ij}}{\sum\limits_{j \in N_i} w_{ij}} \tag{4.6}$$

where, X_h is the set of core nodes associated with the anchor h, N_i is the set of neighbors of node i and w_{ij} is the value of i^{th} row and j^{th} column in the weight matrix W. If r^{th} row of U^+ is specified for the anchor h, the Equation 4.6 computes u_{ri} i.e. membership degree of node i to the r^{th} community.

In addition to the membership degree of communities, the notion of self-membership of all nodes is introduced by considering node itself as an anchor node. However, unlike the membership degree defined in Equation 4.6, core nodes are not considered for computing self-membership degree, instead independent nodes are considered.

Definition 4.3 (Independent Nodes)**.** Given a graph $G(V,E,W)$, set of k anchors and extended partition matrix U^+, based on the membership degrees in the current U^+ the independent nodes are defined as

$$I = V \setminus (\bigcup_{h=1}^{k} X_h). \tag{4.7}$$

The independent nodes are those nodes that does not belong to any of the core nodes associated with the anchors. The self-membership degree is computed for all nodes. The self-membership degree of node i is computed as follows:

$$\text{Self-membership}(i) = u_{(k+1)i} = \frac{\sum\limits_{j \in (I \cap N_i)} w_{ij}}{\sum\limits_{j \in N_i} w_{ij}} \tag{4.8}$$

where, k is the number of anchors, I is the set of all independent nodes, N_i is the set of neighbors of node i and w_{ij} is the value of i^{th} row and j^{th} column in the weight matrix W. The self-membership degree is utilized to identify new anchors.

After computing both membership degree and self-membership degree of all the nodes i.e. all $u_{ji} \in U^+$ for all $j \in [1 : k+1]$ and $i \in [1 : n]$, each column of U^+ is normalized to 1, to satisfy the constraint defined in Equation 4.1.

4.2.3 Redundant and False Anchor Removal

An anchor is a redundant anchor, if all core nodes of that anchor belong to the core nodes of other anchors. Any redundant anchor h is identified as follows. All the core nodes associated with anchor h that are appeared in the core nodes of other anchors are removed from the core nodes of h. If after removal, core nodes associated with h is empty then it is referred as a redundant anchor. All redundant anchors and their associated communities are removed from the extended partition matrix U^+ as follows. To remove the community associated with an anchor from the U^+, simply requires deleting the corresponding row, which is specified for that anchor.

An anchor is a false anchor, if all nodes associated with that anchor have membership degree less than the threshold value ψ. Communities associated with the false anchors will have no members or less membership as belongingness of members are very low.

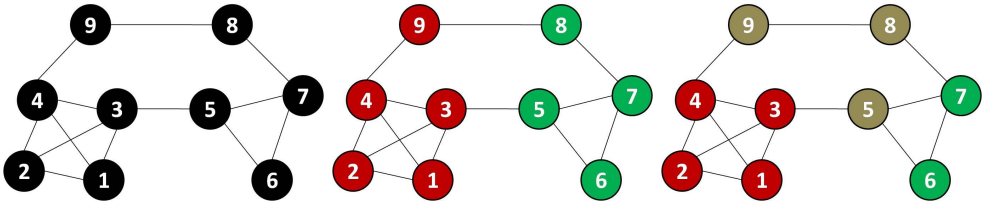
Algorithm 4.1: Fuzzy Agglomerative Community Detection (FuzAg)

```
Input: adjacency or weight matrix W, ψ
Output: partition matrix U ∈ R^{k×n}
anchorList ← initialized anchorList with any randomly selected node
U^+ ← {0}^{2×n}
rflag← 1
aflag← 1
repeat
    U^+ ← compute membership degree of all nodes with reference to current anchors
    if there exist any redundant or false anchor then
        remove all the redundant and false anchors
        remove corresponding rows from U^+ for all redundant and false anchors
        U^+ ← compute membership degree of all nodes with reference to remaining
        anchors
    else
        rflag← 0
    end if
    if any potential new anchor is identified then
        add all new anchors to the anchorList
        add a row in U^+ for each of the new anchors
    else
        aflag← 0
    end if
until rflag=0 and aflag=0 i.e. no redundant, false or new anchors are identified
U ← remove the row for self-membership from the U^+
return U
```

Identification of false anchors is very simple, just requires to check the specified row for the anchor in U^+ whether all nodes have membership degree less than that of ψ or not. If membership degree of all node are less than ψ, the anchor is identified as false, and the specified row is removed from U^+.

4.2.4 Fuzzy Agglomerative Community Detection

The FuzAg algorithm is designed based on the ideas developed above. The algorithm mainly does three tasks as part of community identification process, which are explained



(a) Unpartitioned network (b) Disjoint partitions (c) Fuzzy partitions

Figure 4.1: Communities identified with FuzAg algorithm on an example graph.

as follows. Firstly, the identification of new anchors, which is tackled with the help of self-membership degree. Secondly, updating membership degrees of all nodes with respect to current anchors using Equation 4.6 and Equation 4.8. Thirdly, the removal of redundant and false anchors. Algorithm 4.1 outlines the process of community detection involving the three tasks. The algorithm follows agglomerative approach. The process starts with an arbitrarily selected node as an anchor node. The community associated with the anchor node expands as the membership degrees of nodes are updated. In successive steps new anchors are identified and associated communities are expanded. However, those anchors are not selected on arbitrary basis as did in the case of starting node. Those anchors are selected on the basis of self-membership degree of each node that satisfies the condition mentioned in Equation 4.4. Both redundant anchors and false anchors are removed, and accordingly membership degree of nodes are updated. The process ends when no new anchors, redundant anchors or false anchors are found. Therefore, after completion of last iteration, self-membership degrees of all nodes become 0 i.e. all the entries of $(k+1)^{th}$ row of U^+ are 0. Thus, in final U^+, if only the membership degrees of nodes to communities is considered i.e. U, it will be a fuzzy partition. The algorithm returns U, where the row specified for self-membership degrees in U^+ is excluded.

Community detection with the proposed FuzAg algorithm is demonstrated with various intermediate results obtained in each iteration on an example graph shown in Figure 4.1a as follows:

Initialization:

$anchorList = \{2\}$

$$U^{+} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Iteration 1:

$anchorList = \{2\}$

$$U^{+} = \begin{bmatrix} 1 & 1 & 0.75 & 0.75 & 0.33 & 0 & 0 & 0 & 0.5 \\ 0 & 0 & 0.25 & 0.25 & 0.67 & 1 & 1 & 1 & 0.5 \end{bmatrix}$$

Redundant anchors={empty}

False anchors={empty}

New anchors=$\{5,6,7,8,9\}$

Iteration 2:

$anchorList = \{5,6,7,8,9,2\}$

$$U^{+} = \begin{bmatrix} 0.2 & 0.2 & 0.125 & 0.143 & 0.33 & 0.25 & 0.2 & 0.143 & 0 \\ 0 & 0 & 0.125 & 0 & 0.22 & 0.25 & 0.2 & 0.143 & 0 \\ 0 & 0 & 0.125 & 0 & 0.22 & 0.25 & 0.3 & 0.143 & 0.2 \\ 0 & 0 & 0 & 0.143 & 0.11 & 0.125 & 0.1 & 0.285 & 0.2 \\ 0.2 & 0.2 & 0.125 & 0.143 & 0 & 0 & 0.1 & 0.143 & 0.4 \\ 0.6 & 0.6 & 0.500 & 0.571 & 0.11 & 0.125 & 0.1 & 0.143 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Redundant anchors=$\{5,6,8,9\}$

False anchors={empty}

New anchors={empty}

Iteration 3:

$anchorList = \{7,2\}$

$$U^{+} = \begin{bmatrix} 0 & 0 & 0.25 & 0 & 0.6667 & 1 & 1 & 0.5 & 0.5 \\ 1 & 1 & 0.75 & 1 & 0.3333 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Redundant anchors={empty}

False anchors={empty}

New anchors={empty}

Process Terminates as there are no new anchors, no false anchors or no redundant anchors. Final partition matrix after completing the execution of the algorithm is:

$$U = \begin{bmatrix} 0 & 0 & 0.25 & 0 & 0.6667 & 1 & 1 & 0.5 & 0.5 \\ 1 & 1 & 0.75 & 1 & 0.3333 & 0 & 0 & 0.5 & 0.5 \end{bmatrix}$$

The final partition matrix U contains membership degrees of nodes for two communities. Disjoint communities and crisp overlapping communities obtained after hardening the partitions. are shown in Figure 4.1b and Figure 4.1c respectively. Maximum rule is considered for disjoint communities i.e. nodes are assigned to a community having highest membership degree. Strict-α cut rule is considered for crisp overlapping i.e. nodes are assigned to a community if the membership degree is more than the threshold ψ.

4.3 Time Complexity Comparison

The proposed FuzAg algorithm has three major computational elements: a) finding new anchors, b) membership degree computation, and c) identification of false and redundant anchors. Each computational element is analyzed separately and accumulated at the end.

Finding New Anchors: Self-membership degree of all nodes have to be checked to identify probable anchors, that costs $O(n)$. If p is the number of probable anchors, then to compute total membership degrees assigned to all communities and to identify finally the new anchors require $O(kp)$. Thus, total cost of finding new anchors is $O(n+kp)$. When $k = 1$, the number of probable anchors p can have maximum value $(n-1)$. Again, when $k = n$, the value of p becomes zero. Thus, upper bound of the complexity to find new anchors becomes $O(n)$.

Membership Degree Computation: This can be divided into three sub-problems. First, the identification of core nodes for all anchors, which costs $O(kn)$. Second, finding neighbors of all nodes, which costs $O(n^2)$. Lastly, reachability computation of all nodes with reference to all anchors. Cost of computing the reachability of a node to the core nodes associated with an anchor is $O(qy)$, where q is number of nodes in core and y is average degree of all nodes. The reachability has to be computed for all nodes, so the cost becomes $O(nkqy)$. Now, if $k = 1$, an anchor can have n number of core nodes so cost becomes $O(yn^2)$. Since average degree of all nodes is y, which is constant for a network so the cost becomes $O(n^2)$. If $k = n$, then number of core nodes is simply the neighbors of nodes so the resulting cost is $O(n^2y^2)$. Since y is a constant, the cost of reachability computation becomes $O(n^2)$. In addition to this, the summation of membership degrees of each node has to be normalized to 1, that costs $O(kn)$. Thus, overall cost to compute

Table 4.1: Time complexity comparison of proposed FuzAg with GAFCD, FMM/H2, CFGC and MDP

Algorithm	FuzAg	GAFCD	FMM/H2	CFGC	MDP
Complexity	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(n^3)$	$O(mnk^2)$
Reference	This work	[185]	[186]	[63]	[222]

membership degree becomes $O(kn)+O(n^2)+O(n^2)+O(kn)$. Therefore, upper bound of membership degree computation complexity is $O(n^2)$.

Identification of Redundant and False Anchors: To identify false anchors, membership degree of all nodes have to be analyzed for each of the anchors so the cost of false anchor identification is $O(kn)$. The number of anchors k can have maximum value n, so the upper bound of false anchor identification becomes $O(n^2)$. To identify a redundant anchor costs $O(qx)$, where q is the number of core nodes and x is the number of nodes that are already part of core nodes of any anchor. Redundancy has to be checked for all the anchors so total cost is $O(kqx)$. Maximum value x is n, when all nodes are part of core nodes of some anchor nodes. In that case, core nodes of any anchor node are simply its neighbors so cost becomes $O(kyn)$, where y is average degree of all nodes. Again, the number of anchors k can have maximum value n, so the upper bound redundant anchor identification becomes $O(n^2)$. Therefore, the complexity of identifying redundant and false anchors is $O(n^2)$.

By accumulating the costs of a) finding new anchors, b) membership degree computation, and c) identification of false and redundant anchors, the complexity for an iteration of FuzAg algorithm is deduced as

$$\begin{aligned} T(\text{FuzAg}) &= T(a)+T(b)+T(c) \\ &= O(n)+O(n^2)+O(n^2) = O(n^2). \end{aligned}$$

If the algorithm runs up to maximum t_{max} iterations, the complexity of FuzAg will be $O(t_{max} \times n^2)$. As pointed out by Su and Havens [185] in the time complexity computation of GAFCD that for significantly large n, the maximum iterations t_{max} can be considered as constant. Nevertheless, complexity computation of both MDP and FMM/H2 did not consider maximum iterations t_{max} [186, 222]. Thus, overall time complexity of FuzAg becomes $O(n^2)$. The time complexity of both FMM/H2 and GAFCD is reported as $O(n^2)$ in respective literature [185, 186]. The time complexity of MDP is reported as $O(mnk^2)$, where m is the number of edges and k is the number of communities [222]. In real-world networks, $m \gg n$ so computational complexity of MDP will be much larger than that of FuzAg. The membership degree computation of CFGC requires to measure distance between all pairs of nodes, which is the most expensive operation involved in CFGC. Measuring distance between all pairs of nodes costs $O(n^3)$. Thus, complexity of CFGC is $O(kn^3)$. However, k can be considered as constant if $k \ll n$, which is defined by the user in prior for CGFC like FMM/H2. Thus, complexity of CFGC becomes $O(n^3)$. Table 4.1 summarizes the time complexity of all the algorithms considered for comparison.

4.4 Empirical Analysis

4.4.1 Experimental Setup

Validation Metrics and Datasets: Both quality and accuracy of communities are important so both kinds of metrics are considered. Four quality metrics are considered for evaluating quality of disjoint communities, which include widely used Modularity [143], Coverage [21], ExtD [169] and AVI (see subsection 7.2.3 for detail). To evaluate crisp overlapping communities, generalized fuzzy modularity defined by Havens et al. [79] is considered. Four accuracy metrics are considered for measuring accuracy of disjoint

Table 4.2: Real-world networks used in the experiments

Name	$\|V\|$	$\|E\|$	k	**Network Description**
Dolphin	62	159	2	Dolphin network [122]
Email	1133	5451	-	E-main interchanges [73]
Football	115	613	12	USA college football [61]
Jazz	198	2742	-	Jazz musician network [62]
Karate	34	78	2	Zachary's karate club [218]
LesMis	77	254	-	Les Miserable network [102]
PolBooks	105	441	-	Books on US politics [104]
Sawmill	36	62	-	Sawmill network [133]
Strike	24	34	3	Labor strike network [132]
Words	112	425	-	Adjective and nouns [142]

communities, which include popularly used NMI [184], ARI [157], F-measure and Entropy [225]. Besides these metrics, number of communities (NoC) and iterations required by each algorithm is also presented. Experiments are carried out on ten real-world networks, listed in Table 4.2. Variety of (LFR) [109] benchmark graphs are also considered for experimentation.

Algorithms and Parameters: The performance of FuzAg is compared with four state-of-the-art fuzzy community detection algorithms MDP [222], CFGC [63], FMM/H2 [186], and GAFCD [185]. Both FMM/H2 and GAFCD require to define ranges of number of communities $[k_{min}, k_{max}]$ for different networks. The CFGC algorithm also requires defining number of communities k in prior. Therefore, any random number of communities from the listed ranges in Table 4.3 are considered for an independent run on respective networks. Besides number of communities, the FMM/H2, GAFCD, CFGC have several other parameters as detailed below. The parameters of FMM/H2 are: $\varepsilon_Q = 10^{-8}$, $t_{max} = 500$ and the minimum value of a partition entry $\varepsilon_U = 10^{-6}$. The parameters of GAFCD are: maximum number of generations $g_{max} = 200$, crossover probability $pc = 0.9$, mutation probability $pm = 1.0$, number of fuzzy partitions with the same number of communities $npc = 10$, copy percentage $cp = 0.1$, termination criterion $\varepsilon_Q = 10^{-5}$, $occ_{max} = 100$ and

Table 4.3: Range of number of communities (k) for GAFCD, FMM/H2 and CFGC on each network

Network	Dolphin	Email	Football	Jazz	Karate
Range of k	[2, 10]	[2, 30]	[2,15]	[2, 10]	[2, 10]
Network	LesMis	PolBooks	Samill	Strike	Words
Range of k	[2, 10]	[2, 10]	[2, 10]	[2, 10]	[2, 10]

maximum number of iterations for FMM/H2 used in GAFCD $t_{max} = 100$. The parameters of CFGC are: range of weighting exponent m, [0, 1000] and weight parameter $\omega = 1/n$, where n is number of nodes in the network. The MDP algorithm has two threshold parameters: threshold for propagation termination $\varepsilon_s = 10^{-4}$ and threshold to filter out seed nodes $T_s = 3$ for 10 real-world networks and $T_s = 30$ for LFR graphs. Proposed FuzAg has two parameters, the threshold parameter ψ and maximum number of iterations t_{max}, which are considered as $\psi = 0.25$ and $t_{max} = 100$ for the experiments.

4.4.2 Performance Analysis

Value-based Analysis

For value based analysis, each of the community detection algorithms is executed for 100 independently initialized runs on each of the ten networks. Mainly three values are presented: mean, standard deviation and best values against each of the nine metrics.

Quality Comparison: The performance of proposed FuzAg algorithm is compared with GAFCD, FMM/H2, MDP, and CFGC, in terms of best values of four quality metrics for the predicted disjoint communities on ten networks: Dolphin, Email, Football, Jazz, Karate, LesMis, PolBooks, Sawmill, Strike, and Words. Table 4.4 presents the results of the community detection algorithms in terms of best Modularity, Coverage, ExtD, and AVI values along with corresponding number of communities (k) and required iterations

Table 4.4: Best quality metrics' values with $(k/iter)$ for disjoint communities predicted over 100 runs

Networks	**Algorithms**	**Quality Metrics**			
		Modularity	**Coverage**	**ExtD**	**AVI**
Dolphin	GAFCD	0.5900(4)	0.8050(4)	0.0224(4)	0.6530(4)
	FMM/H2	0.7249(3/4)	0.8491(3/5)	0.0298(3/4)	0.7349(3/4)
	MDP	0.6886(4)	0.8050(4)	0.0233(4)	0.6257(4)
	CFGC	0.7037(2/3)	0.8239(2/3)	0.0417(2/3)	0.6314(2/3)
	FuzAg	**0.7451**(4/5)	**0.8742**(4/5)	**0.0209**(4/5)	**0.7520**(5/5)
Email	GAFCD	0.6848(11)	0.7052(11)	0.0039(11)	0.5374(9)
	FMM/H2	0.6814(6/11)	0.7019(6/11)	0.0031(6/11)	0.5215(6/11)
	MDP	0.6222(17)	0.7507(17)	0.0043(17)	0.2624(17)
	CFGC	0.6697(2/3)	0.6949(2/3)	0.0052(2/3)	0.4539(2/3)
	FuzAg	**0.6912**(14/4)	**0.7705**(17/5)	**0.0025**(14/4)	**0.5411**(14/4)
Football	GAFCD	0.6934(7)	0.7651(7)	0.0262(7)	0.6037(7)
	FMM/H2	0.7685(3/3)	0.7483(3/3)	**0.0213**(3/3)	**0.7321**(3/3)
	MDP	0.6510(7)	0.7194(7)	0.0308(7)	0.5487(7)
	CFGC	0.6051(2/4)	0.6688(2/4)	0.0622(2/4)	0.4988(2/5)
	FuzAg	**0.7790**(9/4)	**0.7949**(9/4)	0.0316(11/4)	0.5211(11/4)
Jazz	GAFCD	0.5536(4)	0.7294(4)	0.0524(4)	0.5452(4)
	FMM/H2	0.6484(2/4)	0.8812(2/4)	0.0332(2/4)	0.7611(2/4)
	MDP	**0.6873**(3)	0.8424(3)	0.0183(3)	**0.6520**(3)
	CFGC	0.5197(2/3)	0.7166(9/3)	0.0699(4/3)	0.5120(2/3)
	FuzAg	0.6776(5/4)	**0.8830**(5/4)	**0.0173**(5/4)	0.6505(5/6)
Karate	GAFCD	0.5537(4)	0.9091(4)	0.0506(4)	0.2500(4)
	FMM/H2	0.6520(2/3)	0.9567(2/3)	0.0347(2/3)	0.5000(2/3)
	MDP	0.5117(4)	0.6923(4)	0.0600(4)	0.4979(4)
	CFGC	0.6181(2/4)	0.9437(2/3)	0.0451(2/3)	0.5000(2/4)
	FuzAg	**0.6712**(2/5)	**0.9913**(2/5)	**0.0276**(2/6)	**0.7083**(2/5)
LesMis	GAFCD	0.6279(6)	0.9268(6)	0.0251(6)	0.5384(6)
	FMM/H2	0.6548(2/3)	0.9671(2/3)	0.0196(5/3)	0.7233(2/5)
	MDP	0.6389(5)	0.9744(5)	0.0160(5)	0.5786(5)
	CFGC	0.5631(3/3)	0.7939(3/3)	0.0136(4/3)	0.4742(2/3)
	FuzAg	**0.6687**(5/5)	**0.9988**(2/4)	**0.0067**(2/4)	**1.0000**(2/4)
PolBooks	GAFCD	0.7608(4)	0.9048(4)	0.0115(4)	0.7099(4)
	FMM/H2	**0.8063**(2/3)	**0.9569**(2/3)	**0.0069**(2/3)	**0.9174**(2/3)
	MDP	0.7669(4)	0.9116(4)	0.0109(4)	0.7146(4)
	CFGC	0.6527(3/4)	0.7755(3/4)	0.0312(3/4)	0.5360(3/4)
	FuzAg	0.6865(7/5)	0.8141(7/5)	0.0223(7/5)	0.4476(7/5)
Sawmill	GAFCD	0.7073(4)	0.8226(4)	0.0430(4)	0.6898(4)
	FMM/H2	0.7547(2/4)	0.7871(2/4)	**0.0210**(3/5)	0.7971(2/4)
	MDP	0.7062(4)	0.8226(4)	0.0241(4)	0.6609(4)
	CFGC	0.4927(2/3)	0.6129(2/3)	0.0716(3/3)	0.4218(2/3)
	FuzAg	**0.7954**(4/5)	**0.8387**(4/5)	0.0271(4/5)	**0.8843**(4/5)
Strike	GAFCD	0.7258(4)	0.8684(4)	0.0446(4)	0.7487(4)
	FMM/H2	0.7645(3/4)	**0.9211**(3/4)	0.0168(3/4)	0.8543(3/4)
	MDP	0.7645(3)	**0.9211**(3)	0.0168(3)	0.8543(3)
	CFGC	0.7368(2/4)	0.8947(2/4)	0.0296(2/4)	0.7857(2/3)
	FuzAg	**0.7666**(3/6)	**0.9211**(3/6)	**0.0167**(3/6)	**0.8552**(3/6)
Words	GAFCD	0.4511(5)	0.5318(5)	0.0400(6)	0.3494(5)
	FMM/H2	0.5890(2/4)	0.7153(2/4)	0.0386(2/4)	**0.5530**(2/4)
	MDP	0.4450(6)	0.5365(6)	0.0453(6)	0.2629(6)
	CFGC	0.4038(2/3)	0.5224(2/3)	0.0649(4/3)	0.3049(2/3)
	FuzAg	**0.7759**(8/5)	**0.9788**(8/5)	**0.0120**(8/5)	0.3068(5/6)

(iter) over 100 independently initialized runs. Bold entries indicate that the results are statistically significant as the best performing algorithm(s). Higher values of Modularity, Coverage and ExtD are statistically more significant, while lower values of ExtD indicate better performance. Results indicate FuzAg is best performer in terms of all quality metrics. FuzAg shows best Modularity, Coverage, ExtD and AVI values on most of the networks. Although quality metrics' values are high, CFGC seems to be worst performer on all the networks. The performance of GAFCD, FMM/H2 and MDP is almost same, mostly they result similar quality metrics' values. However, FMM/H2 performs best on PolBooks network, while MDP results best Modularity and AVI values on Jazz network. Ironically, all of the FuzAg, FMM/H2 and MDP result equal Coverage on Strike networks. GAFCD is also a strong competitor since most metrics' values are quite good.

The performance of proposed FuzAg algorithm is also analyzed in terms of mean and standard deviation of four quality metrics for the predicted disjoint communities. Table 4.5 presents the results of the community detection algorithms on Dolphin, Email, Football, Jazz, and Karate in terms of mean and standard deviation of Modularity, Coverage, ExtD, and AVI over 100 independently initialized runs. Table 4.6 presents the results of the community detection algorithms on LesMis, PolBooks, Sawmill, Strike, and Words. MDP always identifies same communities since it is independent of initialization and follows same sequence of updating partition matrix U. Therefore, mean and standard deviation are not presented for MDP. Results in the Table 4.5 indicate that proposed FuzAg outperforms on almost all of the five networks. FuzAg results best Modularity and Coverage values on all the five networks. FuzAg also results best ExtD and AVI values on Dolphin, Email, Jazz, and Karate, while GAFCD and FMM/H2 result best ExtD and AVI values respectively in Football network. Clearly, CFGC is worst performer on all of the five networks. Performance of GAFCD and FMM/H2 is similar in most of the networks.

Table 4.5: Mean (top) and standard deviation (bottom) of quality metrics' values for disjoint communities predicted over 100 runs of each algorithm-part I

Network	**Algorithms**	**Quality Metrics**			
		Modularity	**Coverage**	**ExtD**	**AVI**
Dolphin	GAFCD	0.5559	0.6623	0.0258	0.5646
		0.0043	0.0054	0.0035	0.0392
	FMM/H2	0.5963	0.6950	0.0368	0.5011
		0.1077	0.1250	0.0209	0.1194
	CFGC	0.3450	0.4035	0.0662	0.2321
		0.1627	0.1916	0.0118	0.1597
	FuzAg	**0.6054**	**0.7082**	**0.0207**	**0.6651**
		0.0424	0.0506	0.0032	0.0373
Email	GAFCD	0.6619	0.6814	0.0046	0.4718
		0.0123	0.0127	0.0013	0.0292
	FMM/H2	0.4851	0.4997	0.0051	0.3323
		0.2018	0.2073	0.0024	0.1536
	CFGC	0.3020	0.3138	0.0071	0.2204
		0.2659	0.2758	0.0013	0.1879
	FuzAg	**0.6722**	**0.6987**	**0.0043**	**0.4811**
		0.0153	0.0326	0.0028	0.0315
Football	GAFCD	0.6441	0.7111	**0.0300**	0.5486
		0.0083	0.0091	0.0006	0.0097
	FMM/H2	0.6488	0.7163	0.0347	**0.5491**
		0.0799	0.0882	0.0178	0.0890
	CFGC	0.3298	0.3644	0.0775	0.2131
		0.1156	0.1277	0.0072	0.1096
	FuzAg	**0.6593**	**0.7378**	0.0395	0.4117
		0.0397	0.0438	0.0044	0.0506
Jazz	GAFCD	0.5536	0.7294	0.0524	0.5452
		0.0000	0.0000	0.0000	0.0000
	FMM/H2	0.5554	0.7352	0.0529	0.5201
		0.0397	0.0517	0.0144	0.0791
	CFGC	0.3833	0.5088	0.0921	0.3302
		0.0896	0.1292	0.0108	0.1241
	FuzAg	**0.5681**	**0.7456**	**0.0427**	**0.5805**
		0.0718	0.0995	0.0142	0.0839
Karate	GAFCD	0.5537	0.9091	0.0506	0.2500
		0.0000	0.0000	0.0000	0.0000
	FMM/H2	0.5388	0.9019	0.0594	0.2567
		0.0899	0.0390	0.0251	0.0941
	CFGC	0.3307	0.8134	0.1037	0.1645
		0.1222	0.0554	0.0234	0.1458
	FuzAg	**0.6118**	**0.9500**	**0.0442**	**0.5392**
		0.0519	0.0285	0.0178	0.1489

Table 4.6: Mean (top) and standard deviation (bottom) of quality metrics' values for disjoint communities predicted over 100 runs of each algorithm-part II

Network	Algorithms	Quality Metrics			
		Modularity	**Coverage**	**ExtD**	**AVI**
LesMis	GAFCD	0.6279	0.9268	**0.0251**	0.5384
		0.0000	0.0000	0.0000	0.0000
	FMM/H2	0.5662	0.9120	0.0358	0.4491
		0.1209	0.0395	0.0209	0.1111
	CFGC	0.5568	0.7397	0.0311	0.2231
		0.1100	0.0545	0.0152	0.1170
	FuzAg	**0.6436**	**0.9274**	0.0341	**0.6278**
		0.0751	0.0297	0.0098	0.1499
PolBooks	GAFCD	**0.7444**	**0.8865**	**0.0132**	**0.6175**
		0.0119	0.0146	0.0014	0.0320
	FMM/H2	0.6410	0.7570	0.0285	0.5548
		0.1203	0.1431	0.0201	0.1421
	CFGC	0.3281	0.3912	0.0638	0.2306
		0.1448	0.1712	0.0130	0.1255
	FuzAg	0.5414	0.6463	0.0342	0.3289
		0.0714	0.0860	0.0062	0.0485
Sawmill	GAFCD	0.7073	0.8226	0.0430	0.6898
		0.0000	0.0000	0.0000	0.0000
	FMM/H2	0.5986	0.6984	0.0410	0.5216
		0.1251	0.1412	0.0238	0.1428
	CFGC	0.2584	0.3158	0.0929	0.1573
		0.1076	0.1312	0.0119	0.0945
	FuzAg	**0.7776**	**0.8237**	**0.0398**	**0.7499**
		0.0540	0.0741	0.0065	0.0682
Strike	GAFCD	0.7258	0.8684	0.0446	0.6487
		0.0000	0.0000	0.0000	0.0000
	FMM/H2	0.6482	0.7732	**0.0436**	0.6177
		0.1117	0.1340	0.0314	0.1536
	CFGC	0.2688	0.3205	0.1198	0.2046
		0.1813	0.2166	0.0273	0.1838
	FuzAg	**0.7542**	**0.8868**	0.0400	**0.7016**
		0.0469	0.0578	0.0078	0.0704
Words	GAFCD	**0.4160**	**0.4844**	0.0520	**0.3127**
		0.0160	0.0216	0.0007	0.0194
	FMM/H2	0.3445	0.4120	0.0537	0.2560
		0.1015	0.1148	0.0123	0.0888
	CFGC	0.1619	0.2042	0.0703	0.1076
		0.0841	0.1046	0.0026	0.0679
	FuzAg	0.2697	0.3455	**0.0455**	0.1354
		0.0980	0.1239	0.0070	0.0401

Table 4.7: Mean, Standard deviation, best with $(k/iter)$ fuzzy modularity for overlapping crisp communities

Network	**Vals**	**Algorithms**				
		GAFCD	**FMM/H2**	**MDP**	**CFGC**	**FuzAg**
Dolphin	Mean	0.5285	0.4151	–	0.3236	**0.6642**
	SDev	0.0004	0.1884	–	0.2077	0.0624
	Best	0.5287(5)	0.6352(5/5)	0.5672(4)	0.4912(8/4)	**0.7441**(7/5)
Email	Mean	0.5623	0.3428	–	0.2943	**0.5681**
	SDev	0.0016	0.2524	–	0.1125	0.1252
	Best	0.5714(14)	0.5535(15/12)	0.5267(17)	0.3877(25/3)	**0.6213**(14/4)
Football	Mean	0.6041	0.5120	–	0.5084	**0.6915**
	SDev	0.0012	0.1655	–	0.2920	0.0478
	Best	0.6046(10)	0.6034(10/5)	0.6586(7)	0.5463(9/3)	**0.8159**(14/4)
Jazz	Mean	0.4452	0.4232	–	0.2438	**0.4624**
	SDev	0.0000	0.0787	–	0.1753	0.1323
	Best	0.4452(4)	0.4478(5/6)	0.0009(3)	0.3687(9/3)	**0.7219**(6/4)
Karate	Mean	**0.4449**	0.3810	–	0.1585	0.4349
	SDev	0.0000	0.1348	–	0.1332	0.0937
	Best	0.4449(4)	0.4449(4/4)	0.3461(4)	0.3658(2/3)	**0.4847**(3/5)
LesMis	Mean	0.5667	0.4477	–	0.0002	**0.5878**
	SDev	0.0000	0.2052	–	0.0109	0.1751
	Best	0.5667(6)	0.6380(8/8)	0.2098(5)	0.0411(8/3)	**0.6739**(4/5)
PolBooks	Mean	0.5268	0.4371	–	0.3269	**0.5301**
	SDev	0.0004	0.1687	–	0.1424	0.0483
	Best	0.5271(5)	0.5290(7/4)	**0.7695(4)**	0.4668(9/3)	0.6417(12/5)
Sawmill	Mean	**0.5501**	0.4240	–	0.1711	0.4350
	SDev	0.0000	0.1862	–	0.2174	0.0820
	Best	0.5501(4)	0.5501(4/4)	0.5451(4)	0.4281(6/3)	**0.6082**(6/4)
Strike	Mean	0.5620	0.4866	–	0.0106	**0.6005**
	SDev	0.0000	0.1635	–	0.2993	0.1052
	Best	0.5620(4)	0.5736(5/10)	0.3703(3)	0.4870(9/4)	**0.6963**(6/5)
Words	Mean	**0.3068**	0.1656	–	-0.8780	0.0643
	SDev	0.0032	0.1421	–	0.0754	0.0301
	Best	0.3122(6)	0.3047(7/12)	0.1866(6)	0.0000(3/3)	**0.3627**(6/5)

Table 4.8: Mean and standard deviation and best with $(k/iter)$ of accuracy metrics' values for disjoint communities

Networks	Algorithms	Values	Accuracy Metrics			
			NMI	ARI	F-measure	Entropy
Dolphin	GAFCD	Mean	0.4578	0.2814	**0.2664**	0.0578
		SDev	0.0098	0.0099	0.0071	0.0059
		Best	0.6536(4)	0.4689(4)	0.3229(4)	**0.0000**(4)
	FMM/H2	Mean	0.4241	0.2832	0.2339	0.2625
		SDev	0.1926	0.1392	0.1489	0.3074
		Best	0.7405(3/13)	0.5813(3/13)	**0.6686**(2/1)	**0.0000**(8/6)
	MDP	Best	0.6756(4)	0.5223(4)	0.5000(4)	**0.0000**(4)
	CFGC	Mean	0.2458	0.0893	0.1594	0.5307
		SDev	0.0857	0.0789	0.0629	0.1959
		Best	0.4637 (9/3)	0.2681(9/3)	0.3034(2/3)	0.0444(9/3)
	FuzAg	Mean	**0.5384**	**0.3525**	0.1472	**0.0316**
		SDev	0.0671	0.1360	0.0658	0.0407
		Best	**0.8028**(4/5)	**0.8735**(4/5)	0.4412 (4/5)	**0.0000**(6/5)
Football	GAFCD	Mean	0.8732	0.7565	**0.4396**	**0.1596**
		SDev	0.0170	0.0431	0.0696	0.0278
		Best	0.8980(11)	0.8302(11)	0.5276(10)	0.1172(11)
	FMM/H2	Mean	0.6980	0.4973	0.2256	0.3829
		SDev	0.2148	0.1945	0.1311	0.2055
		Best	0.8729(10/5)	0.7670(10/5)	0.5793(9/3)	0.1605(10/5)
	MDP	Best	0.7473(7)	0.5547 (7)	0.1948(7)	0.3325(7)
	CFGC	Mean	0.2584	0.0881	0.0716	0.7742
		SDev	0.0798	0.0364	0.0388	0.0816
		Best	0.4302(6/4)	0.2022(6/4)	0.1662(8/3)	0.6310(6/4)
	FuzAg	Mean	**0.8815**	**0.7689**	0.3860	0.1769
		SDev	0.0379	0.0938	0.1486	0.0474
		Best	**0.8982**(12/4)	**0.8418**(11/4)	**0.7045**(11/4)	**0.1023**(12/4)
Karate	GAFCD	Mean	**0.6873**	0.5414	0.2037	**0.1584**
		SDev	0.0000	0.0000	0.0000	0.0000
		Best	0.6873 (4/7)	0.5414(4/7)	0.2037(4/7)	0.1584(4/7)
	FMM/H2	Mean	0.6479	0.5517	0.3055	0.4142
		SDev	0.2262	0.2263	0.2288	0.2936
		Best	0.7465(2/3)	0.7138(2/3)	0.5512(2/3)	0.1733(2/4)
	MDP	Best	0.4854(4)	0.4663(4)	0.2444(4)	0.3085(4)
	CFGC	Mean	0.4847	0.3957	0.2289	0.2951
		SDev	0.1141	0.1821	0.2473	0.1404
		Best	0.6766(2/3)	0.7717(2/3)	0.9410(2/3)	**0.0954**(9/3)
	FuzAg	Mean	0.5335	**0.5631**	**0.5636**	0.4439
		SDev	0.2719	0.3310	0.3504	0.2926
		Best	**0.8372**(2/4)	**0.8823**(2/5)	**0.9706**(2/4)	0.1422(3/7)
Strike	GAFCD	Mean	**0.8641**	**0.7978**	**0.7105**	**0.0000**
		SDev	0.0000	0.0000	0.0000	0.0000
		Best	0.8641(4)	0.7978(4)	0.7105	**0.0000**(4)
	FMM/H2	Mean	0.7204	0.5988	0.4254	0.1469
		SDev	0.2326	0.2628	0.3074	0.2474
		Best	0.8032(3/4)	0.6325(3/4)	0.8815(3/4)	**0.0000**(3/4)
	MDP	Best	0.8331(3)	0.8114(3)	0.7725(3)	**0.0000**(3)
	CFGC	Mean	0.3935	0.2182	0.1787	0.5005
		SDev	0.0891	0.1378	0.1512	0.0919
		Best	0.5652(5/3)	0.5248(2/3)	0.6213(3/3)	0.2000(9/3)
	FuzAg	Mean	0.6575	0.4504	0.1996	0.2148
		SDev	0.1066	0.1837	0.2054	0.1371
		Best	**0.8660**(3/6)	**0.8486**(3/6)	**0.9666**(3/6)	**0.0000**(6/5)

Table 4.9: Comparison of execution time (seconds) required for 100 runs

Network	Dolphin	Email	Football	Jazz	Karate	LesMis	PolBooks	Sawmill	Strike	Words
GAFCD	3526	732345	7281	11252	1316	3089	5455	1523	1453	18859
FMM/H2	38	8955	89	213	26	131	64	23	28	173
MDP	4596	91204387	784365	54233	2718	6935	562188	776	246	44707
CFGC	1845	95207	13704	7385	1663	1447	2901	918	1074	12968
FuzAg	323	28123	715	3381	52	701	2394	113	58	1516

Results in the Table 4.6 show FuzAg performs better in LesMis, Sawmill ans Strike networks, while GAFCD performs better in PolBooks and Words networks. In this case also CFGC performs worst. FMM/H2 shows better quality metrics' values than CFGC. Although GAFCD shows better quality metric values but it seems GAFCD suffers sticking in local optima as standard deviation is zero in some cases.

Furthermore, the performance of the proposed FuzAg algorithm is compared with GAFCD, FMM/H2, MDP, and CFGC, in terms of fuzzy modularity for the predicted crisp overlapping communities on ten networks: Dolphin, Email, Football, Jazz, Karate, LesMis, PolBooks, Sawmill, Strike, and Words. Table 4.7 presents mean, standard deviation and best fuzzy modularity values obtained over 100 independently initialized runs. Results indicate FuzAg outperforms in most of the networks. Both mean and best fuzzy modularity values of FuzAg are mostly better than GAFCD, FMM/H2, MDP and CFGC. Although GAFCD results higher mean fuzzy modularity values on Karate, Sawmill and Words networks, it lefts far behind in terms of best values resulted by FuzAg. Both MDP and CFGC are not performing well at all. MDP results better fuzzy modularity only on PolBooks network.

Accuracy Comparison: Since explicit ground truth communities of most of the networks are unavailable, the performance of FuzAg is analyzed in terms of four accuracy metrics only on four networks: Dolphin, Football, Karate and Strike. Table 4.8 presents mean, standard deviation and best values of NMI, ARI, F-measure, and Entropy for the disjoint communities predicted over 100 independently initialized runs. Higher values of

NMI, ARI and F-measure statistically more significant, while lower values of Entropy indicate better performance. FuzAg results best NMI, ARI, F-measure and Entropy values on most of the networks. FuzAg also results better mean values of all the four accuracy metrics. Although, in some cases GAFCD seems to result higher mean values than FuzAg, but the best values of GAFCD are lagging behind FuzAg. The performance of FMM/H2 and MDP is similar to GAFCD, whereas CFGC shows comparatively poor performance.

Execution Time Comparison: Table 4.9 reports the execution time for 100 runs of the community detection algorithms. These results support practical application of FuzAg as both quality and accuracy of communities are much better than other algorithms. Clearly, FuzAg is much faster than GAFCD, MDP and CFGC. Although, time complexity of FMM/H2, GAFCD and FuzAg is same i.e. $O(n^2)$, FMM/H2 is faster than both GAFCD and FuzAg. FMM/H2 gives fast results because of the early stopping criterion used [186]. Nevertheless, communities identified by FuzAg are much better than FMM//H2 as indicated by various metrics.

MCDM Ranking

Setup for MCDM Ranking: The analysis is focused in two directions. Firstly, MCDM based ranking is done by accumulating all the accuracy metrics and quality metrics in one single score. Secondly, the effectiveness of each algorithm with different accuracy levels is analyzed, formally referred as RITA analysis (see section 7.3 for more detail). TOPSIS method explained by Kou et al. [103] is considered for the ranking. TOPSIS method has the privilege to assign different weights to each criterion. Summation of all weights assigned to different metrics has to be 1.

For MCDM raning, 75% of weights are assigned to accuracy and rest of the 25% weights are assigned to quality metrics to gain more accuracy in communities. Weights assigned

Table 4.10: MCDM ranking score obtained with 75% accuracy and 25% quality.

Algorithms	**MCDM Scores**			
	Dolphin	**Football**	**Karate**	**Strike**
GAFCD	0.4609	0.7894	0.3636	0.6065
FMM/H2	0.5737	0.7659	0.5348	0.5614
MDP	0.5377	0.5419	0.2873	0.5495
CFGC	0.3138	0.2049	0.7115	0.3569
FuzAg	**0.6183**	**0.8033**	**0.7182**	**0.6257**

Note: Higher score indicates more inclination of algorithm towards accuracy. Algorithms acquiring highest rank are indicted in bold.

to both accuracy and quality are equally distributed over respective metrics. There are four accuracy metrics and 75% of total weight 1 is 0.75, which is equally distributed over all four accuracy metrics. Hence, each metrics will get weight $\frac{0.75}{4} = 0.1875$ to contribute in ranking. Similarly, four quality metrics will get weight $\frac{0.25}{4} = 0.0625$ each to contribute in ranking. To analyze effectiveness of communities generated by different algorithms including ours with respect to accuracy, the variation in the ranking score is observed with increment in percentage of accuracy. Accuracy weights are varied from 25% to 75% and apparently quality weights varied from 75% to 25%.

Result Analysis: Table 4.10 presents MCDM ranking obtained for the communities predicted by different algorithms on Dolphin, Football, Karate and Strike networks. Since, more weights are allocated to accuracy metrics so scores obtained will indicate algorithm's inclination towards accuracy. FuzAg shows highest MCDM scores for all the four networks, which indicates FuzAg produces highly inclined communities towards accuracy. GAFCD shows comparatively higher scores than FMM/H2, MDP and CFGC on both Football and Strike networks. CFGC shows least scores on Dolphin, Football and Strike networks but surprisingly, CFGC shows quite high score on Karate network that is

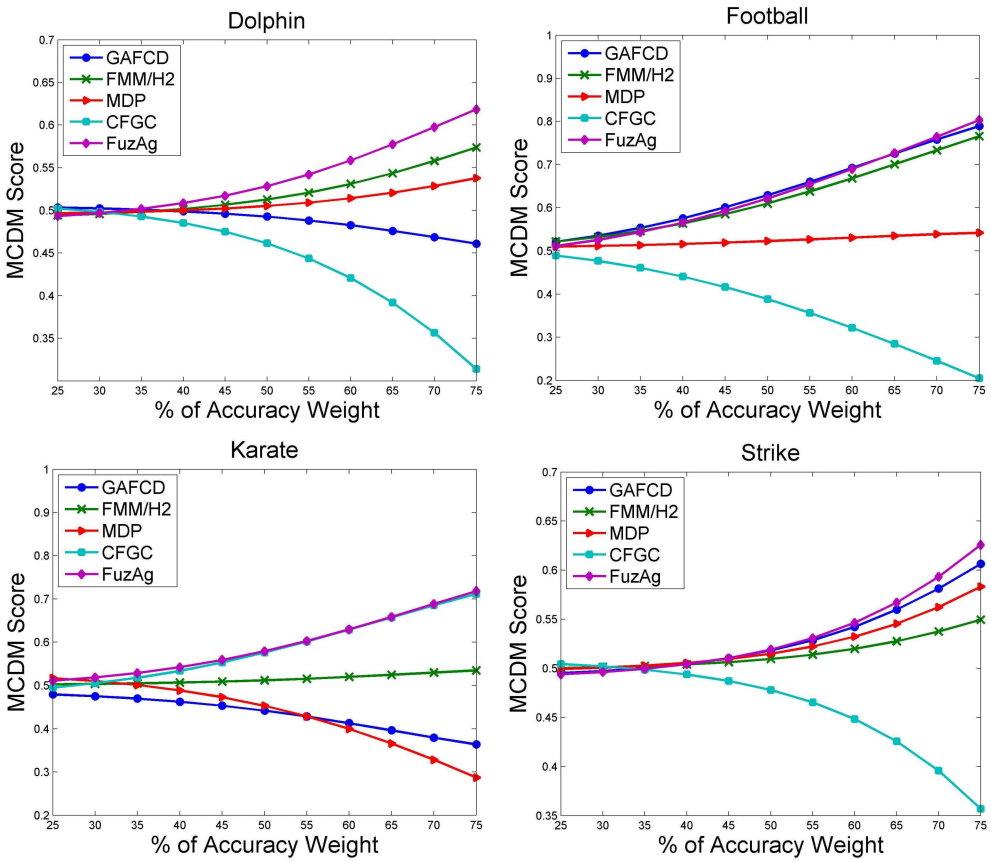
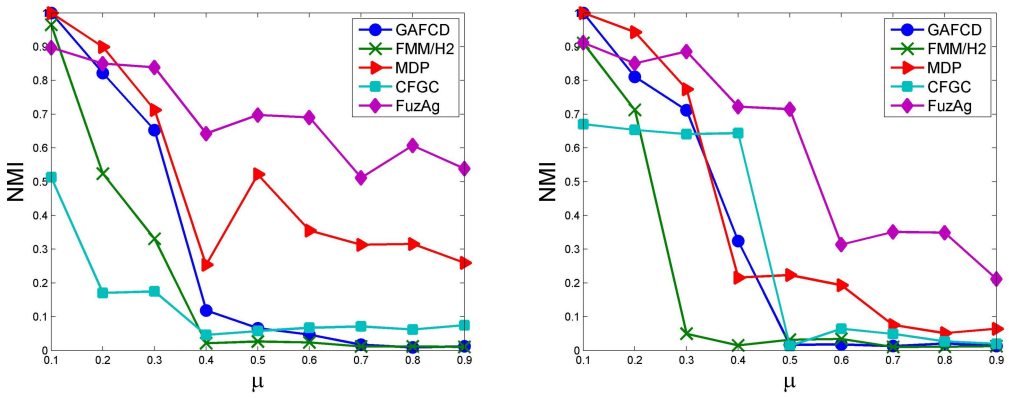


Figure 4.2: MCDM ranking acquired by each algorithm in Dolphin, Football, Karate and Strike networks with variation of accuracy contribution. Higher scores indicate tendency of algorithm's inclination towards accuracy.

comparable to FuzAg. MDP shows least score on Karate networks, while it results better scores than CFGC on other three networks.

MCDM ranking with variation of percentage of accuracy contribution in the MCDM score is presented in Figure 7.9. Initially, when accuracy contribution is given weightage 25%, all the algorithms show almost same scores. When accuracy given 25% weightage for generating MCDM scores, apparently quality contribution becomes 75% so quality metrics will have more impact on the score. As gradually percentage of accuracy increases FuzAg acquires higher scores. FMM/H2 show almost similar characteristics for all the



(a) $n = 1000$, average degree= 5 (b) $n = 5000$, average degree= 10

Figure 4.3: Normalized mutual information (NMI) values obtained on LFR benchmark graph disjoint community.

data sets as FuzAg, but it is unable to show higher scores than FuzAg when the accuracy weightage is high. FuzAg mostly shows higher quality and accuracy metrics' values on all of the four networks so non of the algorithms can overtake FuzAg when accuracy contribution increases up to 75%. However, FuzAg shows lower scores on Dolphin, Football and Strike networks, when accuracy contribution is less. Although, GAFCD being the strongest competitor for FuzAg, GAFCD is preceding FuzAg on all networks except Karate, but at the end when accuracy contribution is high, GAFCD lags behind FuzAg. CFGC is nowhere in competition with the FuzAg on Dolphin, Football and Strike networks, but on Karate network it consistently follows FuzAg. MDP shows comparatively low scores in all of the four networks.

Analysis on LFR Benchmark Graphs

The graphs are generated according to various parameters such as number of nodes (n), average degree, maximum degree, exponent degree distribution and community size distribution, range of community sizes and mixing parameter $\mu \in (0,1)$ for controlling the neighbors in other communities. Two sets of LFR benchmark graphs are generated with

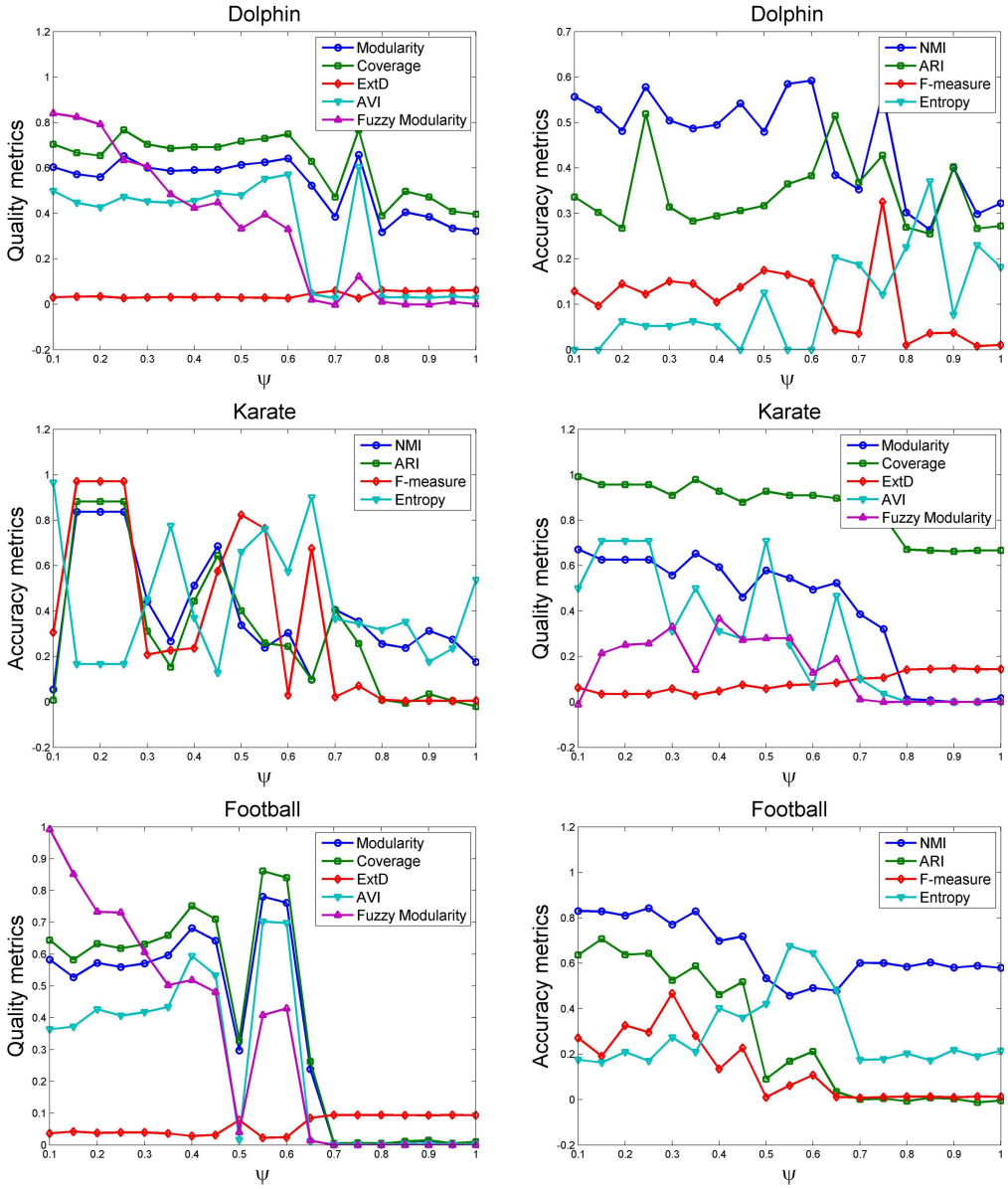


Figure 4.4: Characteristics of five quality metrics and four accuracy metrics with variation of parameter ψ on Dolphin, Football and Karate networks.

1000 nodes and 5000 nodes; each contains nine graphs with respect to the various values considered for the parameter μ. The value of μ is varied from 0.1 to 0.9 with an interval of 0.1 for both the sets. Average degree 5 and 10 are considered respectively for $n = 1000$ and $n = 5000$. Other parameters for the graphs are: maximum degree 30, exponent degree distribution 2, community size distribution 1, range of community sizes [50, 150]. Since LFR graphs have ground truth disjoint communities NMI is considered for evaluating the accuracy of predicted communities.

The performance of FuzAg is compared with other four algorithms on two sets of LFR benchmark graphs of 1000 nodes and 5000 nodes. Each of the two sets contains nine graphs. Figure 4.3 presents the results obtained on both the sets of LFR benchmark graphs. NMI values are presented against the variations of mixing parameter μ. Results indicate that proposed FuzAg algorithm is capable of identifying accurate communities in synthetic networks. Although, GAFCD, FMM/H2 and MDP show more accurate communities than FuzAg for smaller values of μ, NMI values significantly degrade for larger μ values. Smaller μ values result in smaller number of neighbors of any node to be in different communities. The problem complexity becomes comparatively higher for larger μ values. Therefore, if an algorithm shows better results for larger μ that would be more significant. Thus, slightly better performance of GAFCD, FMM/H2 and MDP for smaller values of μ is not so surprising. Consistent better results of FuzAg for larger μ values are clearly noticeable. Although, MDP is also resulting more accurate communities than both GAFCD and FMM/H2, the values of NMI are lower than FuzAg. CFGC seems to be resulting comparatively most inaccurate communities for smaller values of μ, but for large μ values CFGC shows better results than both GAFCD and FMM/H2. Nevertheless, NMI values of CFGC is quite low in comparison to both MDP and FuzAg.

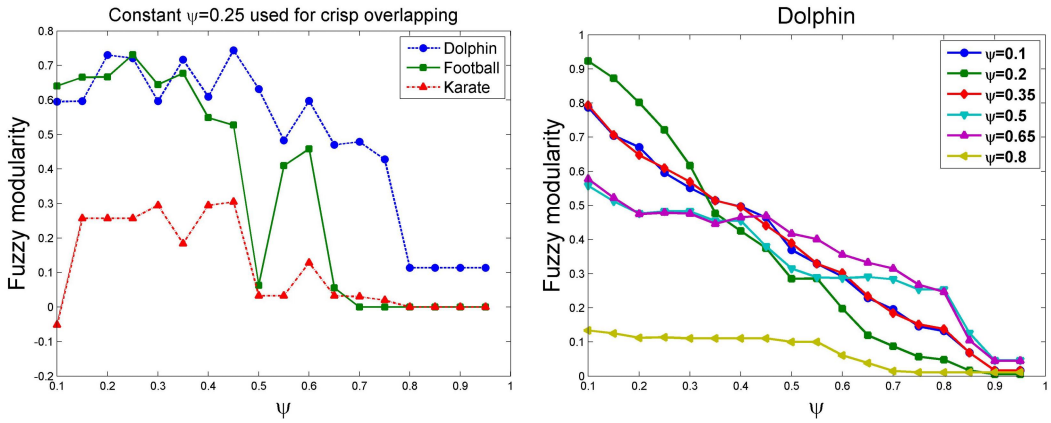


Figure 4.5: Fuzzy modularity resulted (a) by considering constant $\psi = 0.25$ for hardening membership degree to form crisp overlapping communities, and (b) by considering various constant ψ values for hardening membership degree on Dolphin network.

Analysis of Parameter Characteristics

Setup for Parameter Tuning: Proposed FuzAg algorithm has one parameter ψ to tune. To determine appropriate value for ψ, the characteristics of all the quality metrics and accuracy metrics are analyzed on three networks: Dolphin, Football and Karate. To analyze characteristics of metrics, the value ψ is varied from 0.1 to 0.95 with 0.05 interval.

Figure 4.4 presents all of the nine metrics' values with respect to the variations of only parameter ψ on Dolphin, Football and Karate networks. Most of the metrics show almost similar characteristics. For the ψ value range 0.1-0.6, mostly all metrics show better results. However, fuzzy modularity seems to be showing completely different characteristics. Most of the metrics shows comparatively better values in the range 0.1-0.6, while fuzzy modularity continuously degrades. Considering this fact, the ψ values less than 0.3 seems to be more reasonable. Moreover, on all three networks, accuracy metrics seems to be better for ψ value less than 0.3. Although, some spikes are visible in the range 0.5-0.7 for some quality metrics, considering fuzzy modularity values one can conclude that the range is inappropriate.

The parameter ψ is also used for hardening the partitions (i.e. for strict $\alpha-$cut with $\alpha = \psi$) to form crisp overlapping communities. Therefore, specifically the fuzzy modularity is rigorously analyzed with variation of ψ for the algorithm while keeping ψ constant while hardening the partitions. As discussed above, the values of ψ less than 0.3 are better so $\psi = 0.25$ is considered as shown in Figure 4.5a. Interestingly, shows same characteristics as noticed for other metrics in the Figure 4.4. Furthermore, fuzzy modularity is analyzed by considering different constant ψ values for hardening the partitions to check whether for other constant values also show same characteristics or not. Figure 4.5b presents the results of different constant ψ values. Clearly, for constant value of $\psi = 0.2 < 0.3$ shows best fuzzy modularity values only when ψ is varied in the range 0.1-0.3. For higher constant ψ value fuzzy modularity becomes worst (eg. $\psi = 0.8$). Therefore, it is beneficial to use the value of ψ less than 0.3 for FuzAg algorithm to obtain good communities.

4.5 Conclusion

The notion of self-membership is introduced for the first time in fuzzy community detection to give equal opportunity to each node in growing community. A fuzzy agglomerative community detection algorithm called FuzAg is proposed. The algorithm iteratively updates anchors and associated communities. Advantages of FuzAg algorithm are noted below.

- The Algorithm is simple and effective. Most importantly, FuzAg does not required prior knowledge about number of communities and need not to fine tune many algorithmic parameters.
- Complexity of FuzAg algorithm is shown to be $O(n^2)$, which much smaller than MDP and CFGC.

- Results on real-world networks indicate superiority of FuzAg over existing fuzzy approaches in identifying both disjoint and crisp overlapping communities.
- Analysis on LFR graphs revealed the capability of FuzAg in identifying accurate communities even when the numbers of inter-community connections are high.
- FuzAg has been highest ranked in MCDM ranking and shown higher inclination towards accuracy.