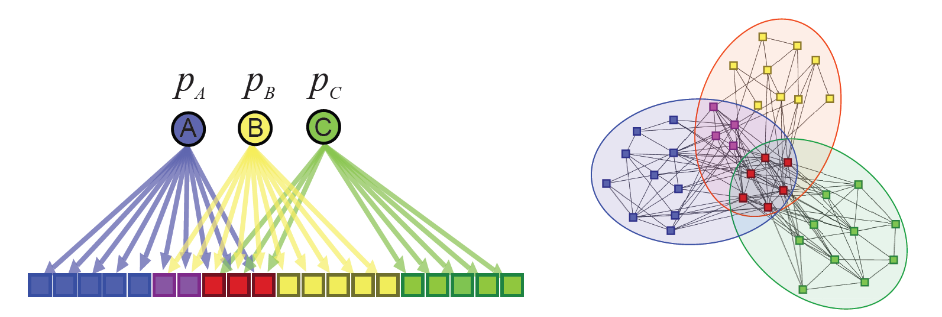
**Community-Affiliation Graph Model (AGM)**



In [1] Yang et al. motivated by the *“observation that community overlaps are more densely connected than the non-overlapping parts”* introduced the Community-Affiliation Graph Model (AGM) for overlapping network community detection. They defined the underlying model of AGM as a bipartite community affiliation, denoted as B (*V, C, M*). where ***V:***represents individuals in the community which corresponds to a set of vertices in the network’s graph representation, ***C****:* represents the communities within the network, M: represents membership/affiliation of the community.

The model assigns a parameter *pc* to each community c ∈ C, which denotes the probability for two individuals in the same community c to connect with each other. The intra-community connectivity probabilities *pc* can be computed as a non-negative vector p = (*pc*)c∈C

In an AGM, given a bipartite community affiliation model B (*V, C, M*), with p as the probability of connection between two members of the community, AGM generates a graph G (V, E) by creating edge (u, v) between a pair of nodes u, v ∈ V with probability given by:

Where C*uv* ⊂ Cis a set of communities that *u* and *v* share. Thus, this ensures that individuals belonging to multiple common communities have a higher probability to connect.

The probability equation (1) does not permit individuals with no mutual community affiliation to link with each other. In order to create this possibility of link between these individuals, AGM uses an assumed additional community called ɛ-community with a very small probability of connection given by:

Given a graph G (V, E), how does AGM detect the network communities?

AGM does community detection by “fitting”. In this process, the AGM tries to find affiliation graph B and parameters {*pc*} to G using maximum likelihood estimation L (B, {*pc*}) = P (G|B, {*pc*}) of G:

Where B is same as B (*V, C, M*), and E is the set of edges in the given network. *P* can be updated by reformulating the log-likelihood as a convex function, thus enabling the use of gradient ascent or Newton’s method to obtain a global optimum for *p*. B is updated using Metropolis-Hasting [2] algorithm.

**Cluster Affiliation Graph Model for Big Networks (BigClam)**

In [3] Yang and Leskovec present the BigClam “*an overlapping community detection method that scales to large networks of millions of nodes and edges*” The BigClam is built on AGM [1] but instead of modeling an intra-community connection probability where communities are detected through “fitting” (which is so hard), the BigClam models a community with weights associated to each member of the community. This model assigns a nonnegative weight to each edge of a bipartite graph, the strength of this weight defines the degree of affiliation.

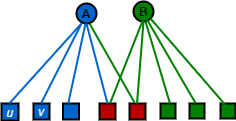
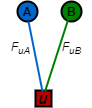
 

Figure 1(a) Community Affiliation Network Figure 1(b) Weight for affiliation

Given the graphs as shown in figure 1(a, b), Here a membership vector ***F***is assigned to each node, this membership vector tells us how strongly a node belongs to the communities. ***FuA***is the membership strength of node u to community **A.** if ***FuA*** = 0 then u and A has no membership relationship between them.

Each community in the network, for example “A” has a probability of having an edge with the nodes (*u, v*) independently. this independent edge probability of community A with node (*u, v*) is proportional to the product of strengths and is given by:

There exists also a probability of at least one common community “C” having an edge with the nodes (*u, v*) is given by:

Equation (5) holds on the assumption that when two nodes share the same community, there is an independent, non-zero chance of connection between the two nodes. Consequently, the more the number of common communities between two nodes, the higher their chances of connecting. Also, the higher the edge weight, the higher the chances of connecting.

BigClam just like the AGM also assumes an ɛ-community with a very small chance of connection ɛ for nodes that does not share a community in common.

Given a network G (*V, E*), how does BigClam detect network communities?

The BigClam does community detection by finding the most likely affiliation factor matrix to the network G by maximizing the likelihood *l*(*F*) = log *P*(G|F) of *G*:

where

To obtain the gradient for a single row *Fu* of *F*:

Where *N*(*u*) is a set of neighbours of *u* under *G.*

Computing and will take a linear time (every node in the network needs to be iterated through) of *O*(*N*) which is not scalable for a network with node greater than a million. In order to reduce the complexity to *O*(|*N*(*u*)|) we compute as in equation (9) which will significantly speedup the computation speed for large networks.

**Improve later from main text …**

**Ego-Splitting Framework**

In [4] Alessandro et al. presents the Ego-splitting framework which is based on the concept of Ego-net. Ego-networks are subnetworks that are centered on a certain node. The Ego-net of node ‘E’ (see figure 2) is defined as the subgraph of ‘E’ (i.e., the Ego-net minus ego of ‘E’)

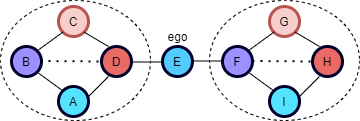


Figure 2: Example of an Ego-net of a node

The Ego-Splitting framework works as follows:

* Create the ego-net of each node
* Partition each ego-net with a non-overlapping clustering algorithm A1
* Create the persona Graph
* Partition the persona Graph with a non-overlapping clustering algorithm A2.
* Obtain the overlapping clusters of the original graph.

A more formal definition, the Ego-net framework requires two clustering algorithms A*l*(a local clustering algorithm) and A*g*(a global clustering algorithm). for a node *u* with an ego-net *Gu*. The following five steps are required for the ego-splitting algorithm to create an overlapping cluster of a graph:

* Step 1: Given a node *u,* partition the ego-net of *u* using a local clustering algorithm. such that Al(*Gu*) = {,,…,}
* Step 2: Create a set  of personas. Each node corresponds to tu personas denoted as ui i= 1, …, tu.
* Step 3: Add edges between personas. For each edge (*u,v*) ∈ *V*, find *ui,vj* such that *u* ∈ and *v* ∈  *.* Add (*ui,vj*) to persona edge *.*
* Step 4: Using the global clustering algorithm on persona graph to obtain clusters *.* (*Gu*) = .
* Step 5: Creation of communities in G. For each cluster ∈  *,*create the associated community formed by the corresponding nodes of : { ∈ *| s.t. .* Output is C { *.*

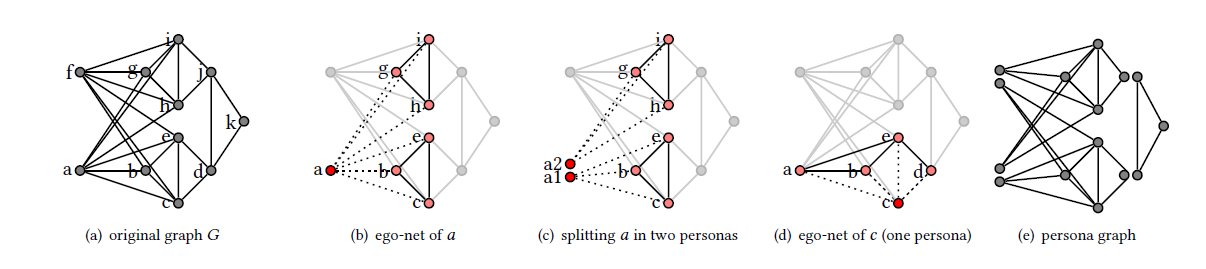


Figure 3: Clustering the ego-nets, splitting the ego and building the persona graph

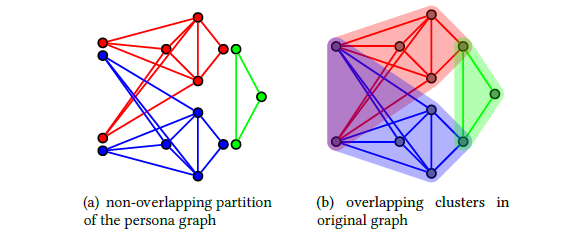


Figure 4: Clustering the persona graph

Figure 3 illustrates the execution steps of the framework using connected components as clustering algorithm. In figure 3b, the ego-net of ‘a’ is obtained and partitioned into two clusters. In figure 3c, foreach of the two clusters detected in figure 3b, a persona is created. For example, a1 is associated with the nodes b,c,e. Same is done for node c in figure 3d but this time only one cluster is detected and one persona c1 associated to nodes a,b,e,d is created. After this is done for all the nodes (in parallel), then a persona graph as shown in figure 3e is created with edges as in the original graph. In figure 4a, a global clustering algorithm is applied on the persona graph. Finally in figure 4b, the clusters defined on the personas is mapped to the overlapping clusters on the original nodes.

Calculating the computational complexity of the ego-nets could be done in the order of which is computationally expensive. An optimal algorithm can create all ego-nets in time with the upper bound depending on the number of triangles in the graph.

The beauty of the ego-splitting framework is the fact that it does not rely on any specific clustering algorithms. Any appropriate algorithm could be used depending on requirements. The framework can also handle weighted and/or directed graphs, provided that the local and global algorithm also support it.

[1] J. Yang and J. Leskovec, "Community-Affiliation Graph Model for Overlapping Network Community Detection," 2012 IEEE 12th International Conference on Data Mining, 2012, pp. 1170-1175, doi: 10.1109/ICDM.2012.139.

[2] M. Newman and G. Barkema. Monte Carlo Methods in Statistical Physics. Oxford University Press, 1999

[3] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In Proceedings of the sixth ACM international conference on Web search and data mining (WSDM '13). Association for Computing Machinery, New York, NY, USA, 587–596.

[4] Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. Ego-splitting framework: from non-overlapping to overlapping clusters. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 145–154. ACM, 2017.