Data-Driven Modeling of Heterogeneous Multilayer Networks And Their Community-Based Analysis Using Bipartite Graphs

by

KANTHI KOMAR

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2019

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Sharma Chakravarthy for providing me an opportunity to work on this project and constantly motivating me and guiding me throughout this thesis work. Without his constant support and guidance this research work would not have been successful. I would like to thank Dr. Ramez Elmasri and Mr. David Levine taking time to serve in my thesis committee.

I would like to extend my gratitude towards Abhishek Santra for constant guidance, support and kind words. Also, I would like to thank my family and friends for their constant support and encouragement throughout my academic career.

July 26, 2019

ABSTRACT

Data-Driven Modeling of Heterogeneous Multilayer Networks And Their Community-Based Analysis Using Bipartite Graphs

Kanthi Komar, M.S.

The University of Texas at Arlington, 2019

Supervising Professor: Dr. Sharma Chakravarthy

Today, more than ever, data modeling and analysis play a vital role for enterprises in terms of finding actionable business intelligence. Data is being collected on a large scale from multiple sources hoping they can be leveraged using big data analysis techniques. However, challenges associated with the analysis of such data are numerous and depends on the characteristics of the data being collected. In many real-world applications, data sets are becoming complex as they are characterised by multiple entity types and multiple features (termed relationships) between entities. There is a need for an elegant approach to not only model such data but also their efficient analysis with respect to a given set of analysis objectives.

Traditionally, graphs have been used for modeling data that has structure in terms of relationships. Single graph models (both simple and attributed) have been widely used as there are a number of packages for their analysis. However, with the increased number of entity types and features, it becomes quite cumbersome to model and difficult (also inefficient) to analyze these complex data sets. Multilayer networks (or MLNs) have been proposed as an alternative. This thesis addresses

elegant modeling and efficient analysis of one type of MLNs called Heterogeneous Multilayer Networks (or HeMLNs).

This thesis addresses modeling of complex data sets using HeMLNs for a given set of analysis objectives of a data set using the popular entity-relationship (or ER) model to meet the analysis objectives. Then it proposes a community-based approach for analyzing and computing the objectives. For this analysis, a new community definition is used for HeMLNs as it currently available only for single graphs. A decomposition approach is proposed for efficiently computing communities in a HeMLN. Since a bipartite graph is part of the community computation of HeMLNs, the role of bipartite graph and algorithms for their use are proposed and elaborated. As the use of bipartite graphs becomes a matching problem, different types of weight metrics are proposed for HeMLN community detection.

This thesis has also conducted extensive experimental analysis for the proposed community computation of HeMLNs using two widely-used data sets: IMDb, an international movie database and DBLP, a computer science bibliography database. Experimental analysis show the efficacy of our modeling and efficient computation of a HeMLN community for analysis.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Data-driven analysis entails modeling of data sets based on analysis objectives and computation of each objective using appropriate functions in an efficient manner. This is especially important when the data sets are complex with multiple entity types and relationships (or features) and a large number of analysis objectives need to be computed. Many approaches and techniques have been used both for modeling and computations. For transactional data, relational model has been widely used from a querying perspective as well as for managing data over a period of time. A number of traditional mining approaches (e.g., decision trees to association rules) have been used based on the type of analysis performed.

Relational databases are used for extracting exact results for a query and for generating reports. In this model, it is difficult to capture the relationships between entities in an elegant manner. If graphs are mapped to relations, some structured information is likely to be lost. Also, the computation of a query will be inefficient due to large number of joins (due to relational representation and normalization.) Operations offered by relational databases are not enough to perform in-exact (or approximate) aggregate analysis such as community and hub detection to discern trends in the underlying data. Hence, Graphs have been used for modeling data that has embedded relationships among entities. This is exemplified by the social network and web data. As the data collected today includes a number of relationships between entities, these relationships need to be taken into account while analyzing the data for extracting information.

Graph or network data structure has been used for modeling and analyzing complex systems in different fields such as social, biological, physical, and information and engineering sciences. A graph or a network can be described as collection of nodes (or vertices) representing some entity or agent, and edge (or link) representing a relationship or interaction between a pair of nodes. This edge can be weighted or unweighted. In network structure, not only entities but also the connections or interactions between the entities are given equal importance. Just by adding a new node and edges, without affecting older ones we can define more elements and relationships between them to discover new insights. This shows the flexibility of network structures.

Traditional approach to modeling structured data is through single/simple graph (also called a Monoplex). Only a single relationship or feature can be modeled through single layer graphs effectively. Even though there are many well known methods for analysing single graphs, capturing multiple features in a single graph is non-trivial. For example, consider the International Movie Database(IMDb) in which two actors can exhibit various types of relationships. Two actors may be co-actors in movies, they might be related because of the genre of their movies, they might have worked with same directors, they might have won Oscars in similar categories, or their movies might have earned similar ratings and reviews. If we want to analyze these type of data taking into account different kinds of relationships, one has to move beyond simple graphs. To capture multiple features elegantly, an attribute graph can be used as an alternative to capture node types and multiple relationships using multiple edges between nodes. In order to analyze attribute graphs, one has to convert them into multiple simple graphs as algorithms widely available to analyze simple graphs cannot handle attribute graphs.

A multilayer network (or MLN) is a network which has multiple layers of simple graphs where each simple graph represents a different relationship. This can be used to model and analyze multi-featured data. For example, if the first layer captures co-actor relationship between two actors, a second layer may capture two actor's relationship based on the similarity of movie genres they act in, then a more informative structure can be modeled. A single graph can not handle multiple features effectively and attribute graphs need to be converted to single graphs for analysis purpose. Hence, it is more desirable to model multi-featured data using multi-layer network.

Multilayer Networks can be of different types. Homogeneous multilayer networks are used to model the distinct relationships that exist among the same type of entities and inter-layer edge sets are implicit as the same set of nodes are present in every layer. Relationships among different types of entities are modeled through heterogeneous multilayer network. The inter-layer edges are explicitly represented to show the relationship across layers between different types of entities. In addition, for modeling multi-featured data that capture multiple relationships within and across different types of entity sets, a combination of homogeneous and heterogeneous multilayer networks can be used, called hybrid multilayer network.

Most of the graphs exhibit many interesting properties that can be used for aggregate analysis. For example, a highly connected group of nodes (termed a community) indicates strong similarity among the nodes for that relationship. Similarly, hubs (or centrality) indicate strong interaction of a node with neighboring nodes. We can analyze these properties further to understand the data better and infer useful aggregate information. We focus on the use of communities for analysis in this thesis. A community is a set of nodes which are densely connected to each other relatively from other nodes. Community structures are quite common in real networks. For

example communities in gene correlation networks indicate genes that have similar functionality [1]. Similarly, citation networks form communities by research topic [2]. Finding an underlying community structure in a network is important as communities allow us to create a large scale map of a network since individual communities act like meta-nodes in the network which makes its study easier. [3] Hence analyzing these communities becomes important. Many algorithms, such as Minimum-cut method, Girvan Newman algorithm, Modularity maximization are developed which uses different strategies to detect communities [4]. There are widely-used packages to community detection such as Infomap [5] and Louvain [6] in a single layered network. Modeling real time data into a multilayer network and detecting communities for analysis of that network, without losing any powerful information is a non trivial task.

The definition of a community is well established for a single graph and community definition is also extended to multilayer homogeneous MLNs to some extent, but work on heterogeneous MLN community detection is lacking. Currently, in heterogeneous multilayer networks, communities are detected by aggregating all individual layer information into a single layer which might lead to loss of information.There is need to extend the definition of the communities to heterogeneous multilayer graphs. Hence, This thesis focuses on significant aspects of modeling and computing communities in **heterogeneous multilayer networks** also denoted as **HeMLN** for achieving analysis objectives.

.

1.1 Problem Statement

The problem being addressed in this thesis is that, *For a given complex data set with F features (explicit or derived) and T entity types along with a set of analysis*

objectives, i) model the data set as HeMLN, that is determine the HeMLN layers (with entities of a specific type as nodes and feature-based relationship as simple edges) and inter-layer edges (corresponding to relationship among entities with different types) based on the analysis requirements and ii) compute desired communities, that is, develop composition-based community detection algorithms for analysis of objectives on arbitrary combinations of layers as needed.

The main contributions of this thesis are:

- Establish the effectiveness of MLNs with respect to alternative modeling approaches, and apply ER-based modeling to derive a HeMLN for the chosen complex data set
- Use of structure- and semantics-preserving k-community for a HeMLN,
- detailed explanation of k-community detection algorithm using the decoupling approach using the maximal flow-based bipartite match.
- Use of traditional pairing bipartite graph algorithm for HeMLN community detection and a new bipartite match algorithm (termed Maximum Weighted Bipartite Coupling or MWBC) for composing layers,
- identification of useful weight metrics and their uniqueness
- Mapping of detailed analysis requirements of the data set using the k-community and weight choices,
- Experimental analysis using the IMDb and DBLP data sets to establish the validity of the proposed approach along with performance analysis.

1.2 Thesis Organization

Rest of the thesis is organized as follows -

- Chapter 1 introduces the need for novel approaches for modeling and analyzing (i.e., efficient computation) complex data sets for a given set of analysis objectives.
- chapter 2 elaborates on complex data sets used and analysis objectives to be addressed.
- Chapter 3 discusses the related research relevant to the problems addressed in this thesis
- Chapter 4 Discusses the justification for our approach – use of Multilayered Network (MLN) and provides details on building MLNs using ER modeling approach.
- Chapter 5 gives an overview on computation and analysis of complex multi-featured data
- Chapter 6 discusses the decomposition approach to analysis and its appropriateness and benefits and also discusses the lack of community definition for HeMLNs and our proposed bipartite graph-based approach.
- Chapter 7 contains the algorithms used in this thesis and its brief explanations.
- Chapter 8 discusses the need for weights for bipartite graph edges along with five distinct metrics for accommodating diverse analysis objectives.
- Chapter 9 has implementation details of the community detection algorithm for HeMLNs.
- Chapter 10 provides analysis of several data sets using the modeling and computation approach proposed along with drill-down analysis
- Chapter 11 includes Conclusions and future work

CHAPTER 2

COMPLEX DATA SETS AND ANALYSIS OBJECTIVES

Data-driven analysis requires a model for representing the complex data as well as computations appropriate for the model for analysis. In this thesis, we use the following three data sets to illustrate analysis-driven modeling and computation. *The choice of data sets is not for demonstrating scalability using large data sets, but for showcasing modeling and analysis of the model. These data sets have been chosen based on user-familiarity of the domains as well as the* ***presence of the ground truth from independent sources for validation****.*

2.1 International Movie Database (IMDb) data set

IMDb data set contains various information about movies produced around the world. It has entities such as movies, actors, directors, etc. Relationships among the entities can be formulated using average ratings, year in which the movie was released, genres of the movie etc.This data set has been chosen for its versatility in that it can be modeled using HeMLN based on analysis requirements.

In the IMDb data set, two types of data sets have been derived. First set contains a random set of 1000 movies that were released anywhere in the world in the period from 2001 to 2015 and the second set has been derived based on the top 500 actors. The movies they have worked in (7500+ movies with 4500+ directors) were extracted. The actor set was repopulated with the co-actors from these movies, giving a total of 9000+ actors.

2.1.1 IMDb, Random 1000 Movie Data Set

The analysis goals for this data set is to analyze the three important groups (entity types) in this data set: actors, directors, and movies where actors are related to each other on the basis of co-acting feature, directors are related on the basis of directing movies of similar genres and movies are related on the basis of their average ratings. Given these three entity types, directs-actor relationship, acts-in-a-movie with a specific-rating relationship, and directs-movie can be analyzed.

for IMDb Random 1000 Movie data Set (termed IMDb-random-1000M), following analysis objectives were posed:

- Find co-actor groups that have *maximum interaction* with director groups who have directed similar genres?
- Identify the *strongly connected* co-actor groups where *most of the actors have worked with most of the directors* who have directed similar genres
- Identify *versatile* director groups who work with *most sought after* actors among co-actors
- For the group of directors (who direct similar genres) having *maximum interaction* with members of co-actor groups, identify the *most popular rating* for the movies they direct?
- For the *most popular* actor groups from each movie rating class, which are the director groups with which they have *maximum interaction*?
- Find the co-actor groups with *strong movie ratings* that have *high interaction* with those director groups who also make movies with similar ratings (as that of co-actors.)

2.1.2 IMDb, Top 500 Actors Data Set

Analysis goal for this data set (termed IMDb-top-500A) is similar to the above data set but entity type Actor is related using their genre as a feature. Two actors are connected if the genres of the movies they act in is similar. Threshold is kept to be 50% for the similarity. This shows that, different features of the data set can be used, based on user's requirement and analysis objectives.

for this data set, following are the analysis objectives were posed:

- Based on similarity of genres, which are the actor groups whose members have *maximum interaction* with the director groups?
- For movie rating classes, which are the *most popular* actor and director groups that have *strong interaction* among them?
- Based on similarity of genres, for each director group which are the actor groups whose *majority of the most versatile members interact*?
- For the *most popular* actor groups, for each movie rating class, find the director groups with which they have *maximum interaction* and who also make movies with similar ratings.

2.2 **DBLP Computer Science Bibliography data set**

DBLP data set (termed DBLP-cs-bib) has many features such as papers published, authors of various papers, conference in which those papers were published and also year in which papers were published. it can be elegantly modeled as a heterogeneous multilayer network. Analysis objecives for DBLP involves entity type such as Authors, Paper and Years. Authors are related on the basis of their co-authorship. Two authors are said to be related if they have co-authored at least three papers.Two paper entities is related if two papers are published in the same conference are con-

nected with an edge. Six major conferences like SIGMOD, VLDB, KDD, ICDM, DAWAK, DASFAA are selected for experiments and analysis. Years are related to each other if they fall in the same defined year ranges. For this data set, following are the analysis objectives were posed:

- For each conference, which is the *most cohesive* group of authors who publish frequently?
- For the most popular collaborators from each conference, which are the 3-year period(s) when they were most active?

In the rest of the thesis, we will show how the above data sets are modeled for the analysis objectives proposed and also elaborate on the computation aspects of the objectives using the HeMLN community definition and the bipartite matching alternatives.

CHAPTER 3

RELATED WORK

As the focus of this thesis is community definition and its efficient detection in HeMLNs, we present relevant work on single or simple graphs (monoplexes) and MLNs (both homogeneous and heterogeneous.) The advantages of modeling using MLNs are discussed in [7–10].

Community Detection in Simple/Single Layer Network: involves identifying groups of vertices that are more connected to each other than to other vertices in the network. This objective is often translated into the problem of optimizing network parameters such as modularity [11] or conductance [12]. The combinatorial optimization problems for community detection are NP-complete [13]. A large number of competitive approximation algorithms exist (see reviews in [14–16]). Algorithms for community detection have been developed for a wide range of inputs including directed [17,18], weighted [19], and dynamic networks [20,21]. Recently there have also been algorithms for identifying overlapping communities [22, 23]. Parallel algorithms on different platforms (distributed, shared memory, GPU and MapReduce) have also been implemented [24, 25]. Only a few of these many algorithms are well known and used by researchers in application domains. However, to the best of our knowledge, there is no work on community detection that include node and edge labels, node weights as well as graphs with self-loops and multiple edges between nodes In contrast, subgraph mining [26–28], querying [29,30], and search [31,32] have used graphs with node and/or edge labels including multiple edges between nodes,

cycles, and self-loops. Even the most popular community detection packages such as Infomap [5] or Louvain [6], do not take these parameters into consideration.

Community Detection in Multilayer Networks: In case of multiplexes, most of the community detection work has been done with respect to the **homogeneous** variant, that is the one where every layer has the same set of nodes and there are no inter-layer edges. It includes algorithms based on matrix factorization [33,34], pattern mining [35,36], cluster expansion philosophy [37], Bayesian probabilistic models [38] and regression [39]. Techniques for determining communities in temporal homogeneous multilayer networks using spectral optimization of the modularity function based on the supra-adjacency representation have also been proposed [40, 41]. Moreover, to study the combinations of different layers in the network some efficient approaches have been proposed where communities obtained from individual layers are combined to obtain communities in any Boolean operation based combined layer multilayer network [42, 43]. All these techniques analyze by either aggregating all the layers of the multiplex or by considering the entire multiplex as a whole. However, in order to holistically study entities and relationships of multilayer networks, we also have to study the combinations of different layers in the network. Although, techniques based on information theory have been proposed for multilayer protein-protein interactions [44], this is only for reducing the number of redundant layers through aggregation, but not as a generalized approach for composing different layers that is possible using our proposed network composition approach. Here we propose novel approaches by which communities or central entities obtained from individual layers can be easily combined to obtain communities or central entities present in any composed multilayer network. To the best of our knowledge, this technique of inferring the communities or central entities of the combined network from communities or central entities of individual layers has not been studied before.

Community Detection in Heterogeneous Multilayer Network: Majority of the work on analyzing HeMLN (reviewed in [45, 46]) focuses on developing meta-path based techniques for determining the similarity of objects [47], classification of objects [48], predicting the missing links [49], ranking/co-ranking [50] and recommendations [51]. An important aspect to be noted here is that most of them do not consider the intra-layer relationships and concentrate mainly on the bipartite graph formed by the inter-layer edges and concentrate more on the bipartite-type of graph formed by the inter-layer edges while proposing the techniques. Moreover, we believe most of these operations will benefit from some version of community detection, centrality computation or subgraph mining. Further, they either consider the multilayer network as a whole or remove the type information of nodes or project one layer onto another while proposing the approaches, thus neglecting the effect of different combinations. Companies such as Google, Amazon and LinkedIn have built very large knowledge graphs that connect different types of entities (persons, places, etc.) via multiple edges that describe the relationship through the edge labels. Such graphs can be easily modeled as heterogeneous multilayer networks. However, the analysis of these knowledge graphs is limited to developing search-oriented techniques ([52], [53]). Further, the main challenges in achieving high level information fusion [54,55] is to link information over different collections so that user queries can be answered and patterns can be identified based on information extracted from images, videos, and correlated with other sources to understand frauds, money-laundering, terrorist activities etc.

The type-independent [44] and projection-based [46, 56] approaches used for HeMLNs do not preserve the structure or types and labels (of nodes and edges) of the community. The type independent approach collapses all layers into a single graph keeping *all* nodes and edges (including inter-layer edges) sans their types and

labels. Similarly, as the name suggests, the projection-based approach projects the nodes of one layer onto another layer and uses the layer neighbor and inter-layer edges to collapse the two layers into a single graph with a single entity type instead of two. The presence of different sets of entities in each layer and the presence of intra-layer edges makes structure-preserving definition more challenging for HeMLNs. A few existing works have proposed techniques for generating clusters of entities [57–59], but they have only considered the inter-layer links and not the networks themselves. However, there **does not exist a consensus on the definition of communities for heterogeneous multilayer networks**. The presence of different sets of entities, inter- and intra-layer edges makes these tasks more challenging. Few existing works have proposed techniques for generating clusters of entities [57–59], but they have only considered the inter-layer links and not the networks themselves. Thus, the combined effect of intra and inter-layer relationships have not been utilized for determining the multilayer communities. Thus, the *combined effect of layer communities, entity types, intra- and inter-layer relationships (types) have not been included in defining a community in a HeMLN*. This thesis hopes to fill that gap.

Further, to define communities in HeMLN, an already existing matching algorithm has been used [60]. This method is based on the blossom method and the primal-dual method for finding a matching of maximum weight. This algorithm provides one to one matching. The problem of one node equally being be a best match to multiple nodes, has not been addressed. Hence, another algorithm which gives one to many matching based on weights has been introduced. Experimental analysis has been shown using both algorithms.

CHAPTER 4

ER APPROACH TO MODELING MULTI-FEATURED DATA

Many real world data sets are have multiple entity types and multiple features spanning the entities. The first step is to model the complex data set as a MLN (in our case HeMLN) based on the objectives posed. Instead of doing it in an *ad hoc* manner, we use the widely-popular Entity-Relationship (or ER) approach for modeling. This makes it easier to go from an English description to a formal model which can be used for mapping objectives. Note that ER model is not unique for a given data set and analysis objectives.

Before we explain our ER-based approach to modeling a HeMLN, we discuss the modeling alternatives and why the use of MLNs is better from both the modeling and analysis perspectives.

4.1 Modeling Complex Data Sets Using Graphs

The **traditional approach** (termed single graph or simple graph or a monoplex) models entities as nodes (including labels) and features as edges (including labels and/or weights, if present) of the graph. This model gives rise to graphs with single node types (entity types are not distinguished even when multiple node types are modeled using this approach) and single edges between nodes (either for a single feature or for a combination of features).

This approach is the most popular representation as large number of computations exist f or simple graphs. There are several algorithms for analyzing simple graphs, such as detecting cliques, communities, hubs, mining subgraphs, triangles etc.

Most of these are applied on the entire graph and typically ignore labels and weights in the graph. Although combining multiple features for creating simple graphs is possible, it is not straightforward. If features are of different types (e.g., numerical and categorical), combining them in a meaningful way may not be correct or feasible. In addition, assigning importance to different features could be daunting as well. Finally, if one wants to analyze a subset of entities and associated feature types, separate simple graphs have to be created (or separated) for each such combination or analysis.

The **second alternative** is an *attribute graph*. Additional aspects of the data set can be captured by including node types in terms of labels (even multiple labels) and multiple edges corresponding to relationships for different features again with or without labels and/or weights. Subgraph mining [28], by default, have used attribute graphs including multiple edges between nodes, cycles, and self-loops. Also, most of the other algorithms available for simple graphs do not handle attribute graphs. In order to use these graphs on combinations of features, one needs to create separate graphs for each such analysis. Given F features, this may entail creating 2^F simple/attribute graphs for analysis. Although attribute graphs retain more semantic information than simple graphs, aggregation algorithms (e.g.,community and hub detection) are not available for processing general attribute graphs. In order to use extant graph algorithms, a single graph (i.e., single edges between nodes, and only one type of nodes) may have to be created even for data sets with multiple entity and features types.

The **Third alternative** is the use of MLNs as discussed in this thesis. In this thesis, it is argued that a ***M**ulti**L**ayer **N**etwork (or **MLN**) is better-suited for modeling if the associated analysis can be done on MLNs without transforming them.* Although MLNs have been used in the literature for modeling [7, 10], the current

analysis approaches map them to an equivalent single graph form. Due to this process many of the information in the multilayer graphs are lost. they are reduced to the traditional representation for analysis losing the effectiveness of modeling. Further, three types of multiplexes – homogeneous, heterogeneous, and hybrid can be used – that are useful for modeling data sets based on analysis needs [8]. **Overview of MLN and Its Major Types:** Briefly, a MLN is a collection of layers (each layer is a simple graph) whose nodes are connected within each layer (intra-layer edges) and in addition, nodes from different layers can also be connected explicitly (as inter-layer edges) if they are of different types. The choice and semantics of nodes for modeling layers and intra-layer edges as well as inter-layer edges are analysis-driven. Layers are formed based on the analysis objectives with respect to the number of entities and features needed. Once the layers are formed, they can be analyzed for feature-wise and aggregate (combinations of feature and/or entity types) analysis. It is easy to visualize that in some MLNs, every layer has the same node set.

Airlines provide flights between cities can be represented as a graph. Capturing all the airlines in the same graph requires an attributed graph (to show flights from different airlines between the same cities!), makes it complicated and difficult to analyze individually or as combinations. On the other hand, it can be elegantly modeled as a MLN as nodes in each layer are the same (cities) and each layer can represent an individual airline. Within a layer, two nodes (cities) are connected if there is a direct flight between them. This type of MLN where each layer has the same set and type of nodes is termed a **Ho**mogeneous **M**ulti**L**ayer **N**etwork (or **HoMLN**.) The inter-layer edges are implicit as the node sets of all layers are identical and are not drawn. Figure 4.1 (a) shows the HoMLN example for the Airline data set.

Another option using a MLN is to model a data set using layers except that each layer has a different entity type as its nodes (e.g., actors, directors, and movies).

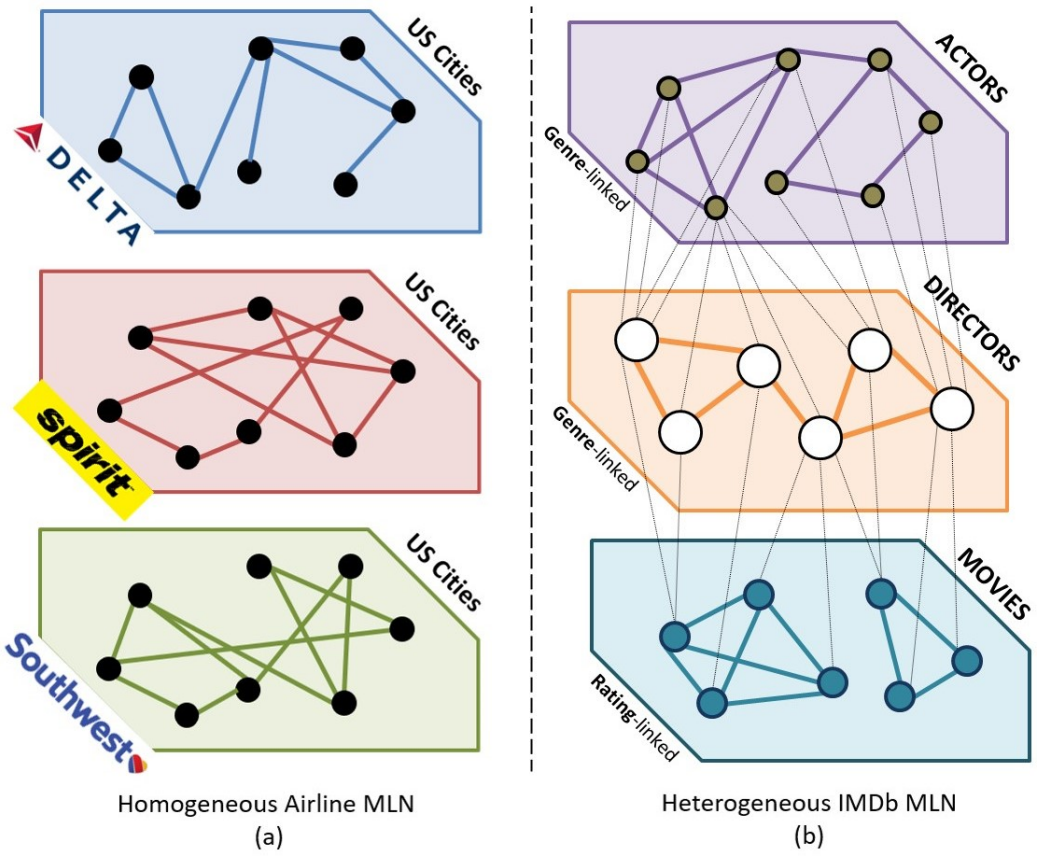


Figure 4.1: MLN Modeling Alternatives

The graph of a layer is defined with respect to chosen features and entity types (e.g., co-acting.) Additional aspects of the data set can be captured using this model by including node types in terms of labels (and even multiple labels) and multiple edges corresponding to relationships corresponding to different features again with or without labels and/or weights (termed *attribute graphs* in the literature.) directing similar genres, similar rating range). In this case, the inter-layer links are not implicit, but are defined explicitly based on feature semantics that corresponds to an edge (e.g., directs-actor, directs-movie, acts-in-a-movie). This type of MLN where each layer has a different set and type of nodes is termed a **He**terogeneous **M**ulti**L**ayer **N**etwork (or **HeMLN**.) Figure 4.1 (b) shows an example from the IMDb data set. A data set may need to be modeled in both ways based on analysis requirements. Again the

choice of HoMLN or HeMLN is driven by the data set and analysis objectives. It is possible to model the *same data set* using both of the above alternatives depending on the analysis needs. It is also possible to have a combination of these two types of multiplexes for a given data set and is termed a **Hy**brid **M**ulti**L**ayer **N**etwork or **HyMLN**. For example, combining the two data sets indicating the city in which actors and directors live results in a hybrid MLN. Interesting analysis, such as the "Number of airlines needed to reach a shooting location using a direct flight for actors and directors", can be formulated.

4.1.1 Advantages of MLNs for Modeling

From a modeling perspective, especially for a data set with multiple entity, feature, and relationship types, a MLN is a more natural and elegant choice. MLNs allow features to be separately modeled as a layer for understanding and preserving the semantics of the data set with respect to analysis objectives. This is not the case for the other representations. It is flexible for combining features for aggregate analysis without having to collapse or aggregate them as discussed below. Incremental changes for each layer can be accommodated without extensive re-modeling.

.

4.2 Modeling Data Sets into Multilayer Networks

Any analysis objective which is achieved by data involving entities, relationships can potentially benefit from a data model. The three real world data sets used in the thesis as described in chapter 2 (*Data sets and analysis objectives*) are multi-featured and have heterogeneous entities. The analysis objectives to be achieved can greatly benefit from multilayer model. Hence, it is shown how a data set having multiple features and complex relationships can be modeled. Entity Relation diagrams are

well-established and have been used to model and design relational databases for generating schema. An entity-relationship (ER) diagram is crucial to creating a good database design. It is used as a high-level logical data model, which is useful in developing a conceptual design for databases. Below, we show how an ER model can be used to derive the ER diagram for multi featured IMDb and DBLP data sets.

4.2.1 Modeling IMDb Dataset

IMDb data set contains, movie information, actors and directors in those movies, movie ratings, year in which it was released and genres of the movie. Gathering the information present in the IMDb dataset and our set of analysis questions, one can build an Entity Relationship diagram as described below:

- Actor can be considered as an entity type with attributes, actor id, name, where actor id acts as primary key.
- Director can be considered as the second entity with attributes, director id and name. Here, director id acts primary key.
- Movie can be considered to be another entity type with attributes such as Movie ID, name, genres, rating and year with movie ID acting as primary key.
- In Actor entity type, the entity type participates in a relationship with itself. In such cases it becomes essential to show recursive relationships. Hence a relationship *Acts with* is established. If two actors are co-actors in a movie they are said to be related.
- Similarly, it is necessary to establish recursive relationship among directors. If two directors have directed movies of similar genres, then a *works in similar genre* is created.
- One has to connect similar movies together based on the movie's rating to achieve above mentioned objectives. This can be achieved by establishing a

recursive relationship in movies. Two movies will be related if both movies fall in the same range of ratings. It can be named as *falls in same rating range*

- A relationship type can be established between actor entity and director entity. If a director has directed that actor, a relationship *directs actor*, is established between them.
- To establish relationship between actor and movie entities, *Acts in movie* relationship is created to denote which actors work in a movie
- Similarly relationship between director and movie entities is established to show director's work in a movie and relationship called *Directs movie*
- *Acts with* recursive relationship has cardinality ratio as M:N, meaning that each actor can be related to any number of actors. *Works in similar genre* recursive relationship has cardinality M:N, showing that one director can work with multiple directors. Cardinality of relationship *falls in similar ratings* is also M:N which shows that, a movie can be related to one or more movies.
- Binary relationship *directs actors* between actor and director entity has M:N relationship which means that, a director can direct one or more actors and also actors can work under one or more directors. Similarly other two binary relationships *acts in movie* and *directs movie* also has M:N cardinality.

A complete entity relationship (ER) diagram is as show in figure 4.1

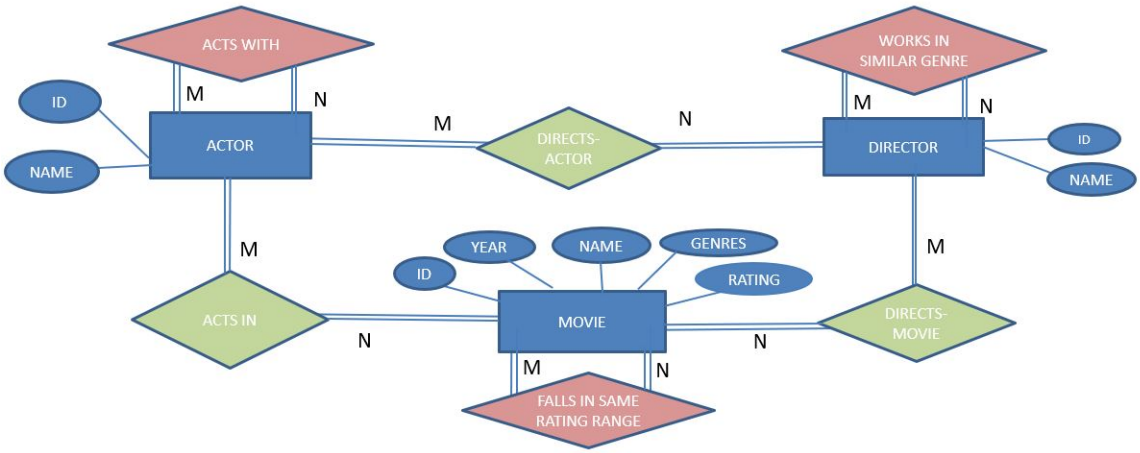


Figure 4.2: IMDb ER Diagram

We can verify whether all the analysis objectives are covered once the ER diagram of a HeMLN is developed. Essentially, this corresponds to making sure all relationships – both recursive and non-recursive have been modeled to cover the analysis objectives. if new analysis objectives need to be taken into consideration, it amounts to extending the current Er diagram with possibly new relationships and/or entities.

Once the ER diagram is modeled, mapping now is to generate MLNs instead of relations as in the traditional sense. Below are the detailed steps to convert a ER diagram into a MLN.

- Each entity in the entity relationship diagram is modeled as a separate layer in multilayer network (MLN). Hence, actor, director, and movie entities will become distinct layers. Entity set of an entity becomes nodes of the layer. For example, actor entity set will be denoted as actor nodes in the actor layer. Similarly, directors nodes and movie nodes will be generated in director and movie layer respectively. Attributes of entities typically becomes labels of nodes

and it cab be chosen, again, based on the specifications in the analysis objectives. In the case of IMDb data set, three layers will be generated.

- Each layer in a multilayer network also has intra-layer edges in between those nodes. Recursive relationship of each entity will be made as intra-layer edges in MLN model. If there are multiple recursive relationships for an entity, they become separate layers with the entity se or nodes being the same. we do not discuss this further as it corresponds to a homogeneous MLN.

  Acts with relationship will act as edges between actors. If two actors have co-acted, there will be an edge between them. If two directors *works in similar genres*, they will have an edge between them in the director layer. Similarly, in movie layer if two movies fall in the same range of ratings, that is, *has similar ratings*, the movies will have an edge between them
- Binary relationships in the ER diagram correspond to the inter-layer edges between two layers. *directs actor* relationship between director and actor will form an inter layer edges between director and actor. If a directer has directed an actor, those two actor and director nodes will have an edge between them. *acts in movie* relationship between actor and movie helps us in building inter-layer edges between actor and movie. If an actor has acted in a movie, the actor and the movie node will have an edge between them. The same logic can be applied to the relationship *directs movie* between director and movie. If a director directs that movie, that director and movie will have an edge between them.
- Both recursive and non-recursive relationships can be one-to-one, one-to-many, or many-to-many depending upon the semantics of the relationship.

Using the above steps, a multilayer network can be built. It has three layers namely, actor, director and movie and three sets of inter layer edges, as show in figure 4.3

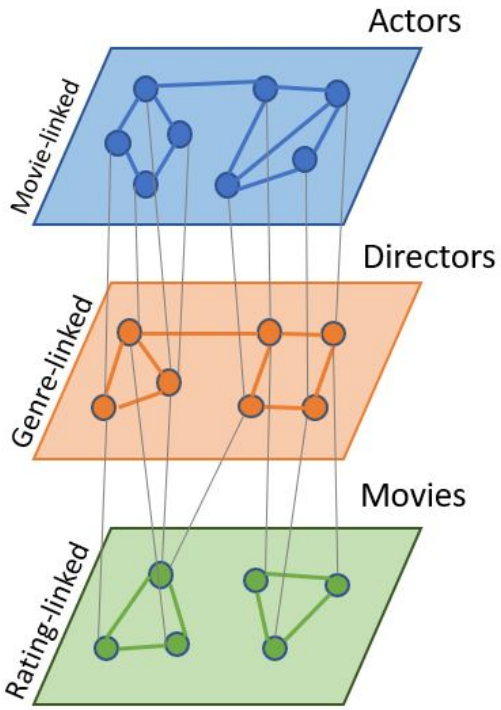


Figure 4.3: IMDb Multi Layer Network

4.2.2 DBLP dataset Modeling

DBLP dataset also exhibits heterogeneous characteristics and can be modeled as a multilayer network.

- Author can be an entity type with attributes such as Author ID, name, with Author ID acting as primary key.
- Paper can be considered to be another entity type with attributes such as Paper ID, name with paper ID acting as primary key.
- Year is also considered as an entity which has an year ID as primary key.

- In Author entity type, the inter-layer relationship can be exhibited as recursive relationship, where author collaboration is described. Hence a relationship *Collaborates with* is established.
- Similarly, it is necessary to establish recursive relationship among papers. If two papers are published in same conference, then a *published in same conference* is created. The intra-layer edges in year can be represented by creating recursive relationship *falls in same range* among year if they fall in the same range. (for example 2001-2003 can be defined as one range).
- To connect Authors and Papers, a relationship between these two entities can be established called *writes* if an author has published the paper.
- To establish relationship between author and year entities, *active in* relationship is created to denotes if an author was active in that year.
- Similarly relationship between paper and year entities is established to show in which year the papers were published and it is called *published in*.
- *Collaborates with* recursive relationship has cardinality ratio as 0:N, meaning that each author can work individually or with any number of authors.*published in same conference* recursive relationship has cardinality M:N, showing that one paper can be related to one more paper. Cardinality of relationship *falls in same range range* is also M:N which shows that, a year can be related to as many number of years depending on the range defined.
- Binary relationship *writes* between author and paper entity has M:N relationship which depicts that, a author can publish one or more papers and also paper can have one or more authors. Binary relationships *active in* has cardinality as 0:N as author may not be active in any years or can be active in many years. and *published in* binary relationship also has 1:N cardinality as paper can published only in one year but many papers can be published in a year.

completed extended entity relationship(ER) diagram is as show in figure 4.1

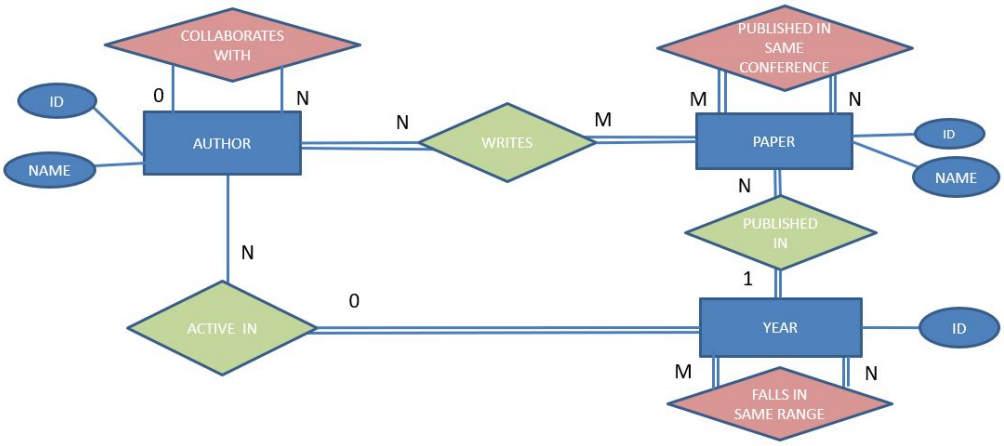


Figure 4.4: DBLP ER Diagram

As described in the IMDb example, DBLP ER diagram can also be converted into a multilayer network model. Following are the steps to be followed.

- Author entity can be converted into a layer in which authors act as nodes and recursive relationships *co-authors* acts as edges. Two author nodes can be connected if they have worked together as co-authors.
- Paper entity can be added as a second layer where paper acts as nodes and edges represents the recursive relationship *published in similar conference*.
- Similarly, year forms another layer which has years as nodes and edges between the year denotes the recursive relationship *published in same year*.
- Binary relationships such as *writes*, *published in*, *active in* forms intra edges layers between author-paper, paper-year, author-year respectively.

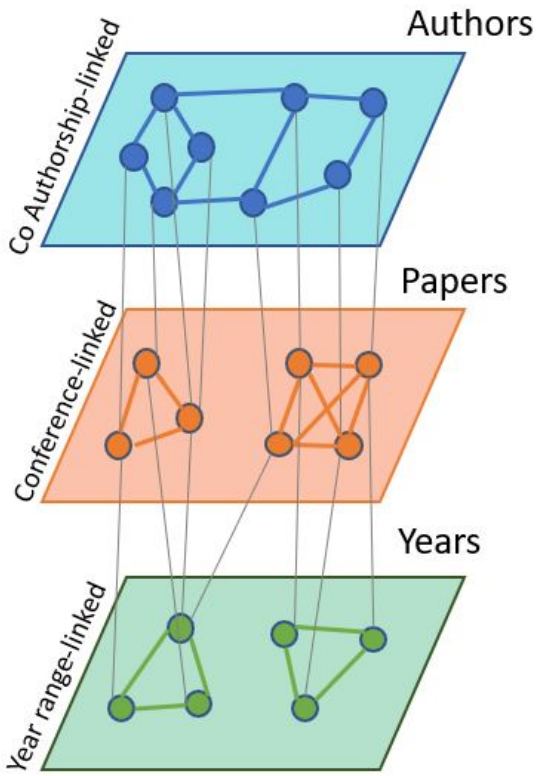


Figure 4.5: DBLP Multi Layer Network

CHAPTER 5

EFFICIENT COMPUTATION AND ANALYSIS OF MULTI-FEATURED DATA

Once the data set has been modeled using one of the MLN alternatives, the goal is to use efficient algorithms for computing communities on individual layers and compose them as needed for meeting analysis objectives. This thesis, concentrates on *communities* in HeMLNs. Other types of analysis, such as centrality, subgraph mining, even querying and search are applicable.. If the *model is a simple graph* (**first alternative**), a number of algorithms for community and hub detection are available (e.g., Infomap [5], Louvain [6] being prominent ones for community detection). Computationally, the options for a single graph approach for complex data sets are: i) combine features in some way and use extant simple graph algorithms, ii) aggregate or collapse the MLN into a simple graph (type independent [44], projection-based [46, 56]) and apply extant algorithms, Both approaches do not preserve either the structure or semantics of MLNs as they aggregate layers into a simple graph in different ways. Importantly, aggregation approaches are likely to result in some information loss [10] or distortion of properties [10] or hide the effect of different entity types and/or different intra- or inter-layer relationship combinations as elaborated in [61].

If the *model is an attributed graph* (**second alternative**), the choice of aggregate computations (e.g., community, hub) is limited or even not available. Some approaches use the multilayer network as a whole [62] and use inter-layer edges, but do not preserve the layer semantics completely.

It is possible to model a complex data set as a MLN and analyze it as a MLN without aggregating or collapsing. The decoupling approach used in this thesis (in **third alternative**) accomplishes this. It also preserves the structure and semantics of the results. This is important for further drill-down analysis of results as we shall elaborate later.

Current approaches, such as using the multilayer network as a whole [62], Current approaches for aggregation, such as type-independent and projection-based, do not accomplish this as they aggregate layers into a single graph in different ways. Importantly, the above aggregation approaches are likely to result in some information loss [10] or distortion of properties [10] or hide the effect of different entity types and/or different intra- or inter-layer relationship combinations as elaborated in [61]. Modeling a data set as MLNs and analyzing using structure-preserving. Preserving the structure of a community of a MLN entails preserving not only their multilayer network structure but also node/edge types, labels, and importantly inter-layer relationships. In other words, for example, each community should be a MLN in its own right. computations avoids these and further facilitates drill-down of each community for detailed analysis.

Briefly, for using the **decoupling approach** [42, 63], each layer is analyzed independently and the results are composed using functions to generate highly accurate (and in many cases loss-less) results. This approach has also the advantage of leveraging single graph algorithms on each layer independently. The decoupling approach has been shown to be more efficient than traditional approaches for analyzing MLNs.

If this is to be leveraged for aggregate analysis, techniques need to be developed to combine partial results from each layer. This has to be done either in a loss-less manner or with high or bounded accuracy. This is the primary challenge for using this modeling alternative. Modeling in terms of layers provides a natural "divide and

conquer" capability which is amenable to parallel processing as well. With the availability of layers, combinations of features can be analyzed without having to re-create a new graph for each such analysis which is exponential in the number of features to be handled. Also, combining partial results is likely to be computationally small as compared to layer analysis. Hence, efficiency can also be significantly improved using this modeling if efficient concomitant composing functions are developed.

In summary, modeling alternatives give rise to new computational challenges as discussed above. In spite of that, we argue that overcoming these challenges will result in solutions to big data analysis that are flexible, efficient, and scalable. Once the layers have been identified and their analysis is available, computing "what-if" analysis is quite straightforward. Even if modeling results in multiple sets of MLNs, analysis can be performed across layers from different MLNs equally easily. Furthermore, updating the data set results in updating corresponding layers and recomputing only updated layers, or just updating the partial results. Even here incremental techniques (existing and new ones) can be applied. Temporal analysis is possible by creating MLNs for desired periods and they can be cross analyzed without incurring expensive computations again. In summary, the proposed approach has significant benefits as compared to the alternatives.

more importantly each layer can be analyzed using extant algorithms. As we shall see later, MLN modeling is also amenable to efficient processing. There are other advantages as well for this modeling. if one wants to find out the collaboration between actors and directors we can represent such information in heterogeneous multilayer graphs. One layer captures the interaction between actor entities, while the second later captures the interaction between director entities. The inter layer links captures the interaction between two different entities like actor and director entities.

Similarly there are many other datasets which might require this representations for better seizure of information.

CHAPTER 6

COMMUNITY DEFINITION FOR A HeMLN

A **graph** G is an ordered pair (V, E), where V is a set of vertices and E is a set of edges. An edge (u, v) is a 2-element subset of the set V. The two vertices that form an edge are said to be adjacent or neighbors. In this thesis we only consider graphs that are undirected. A **multilayer network**, $MLN(G, X)$, is defined by two sets of graphs: i) The set $G = \{G_1, G_2, \ldots, G_N\}$ contains graphs of N individual layers as defined above, where $G_i(V_i, E_i)$ is defined by a set of vertices, V_i and a set of edges, E_i. An edge $e(v, u) \in E_i$, connects vertices v and u, where $v, u \in V_i$ and ii) A set $X = \{X_{1,2}, X_{1,3}, \ldots, X_{N-1,N}\}$ consists of bipartite graphs. Each graph $X_{i,j}(V_i, V_j, L_{i,j})$ is defined by two sets of vertices V_i and V_j, and a set of edges (also called links or inter-layer edges) $L_{i,j}$, such that for every link $l(a, b) \in L_{i,j}$, $a \in V_i$ and $b \in V_j$, where V_i (V_j) is the vertex set of graph G_i (G_j.) For a HeMLN, X is explicitly specified.

Without loss of generality, we assume unique numbers for nodes across layers and disjoint sets of nodes across layers Heterogeneous MLNs can also be defined with overlapping nodes across layers (see [10]) which is not considered in this thesis. We propose a **decoupling approach for HeMLN community detection**. Our algorithm is defined for combining communities from two layers of a HeMLN using a composition function and is extended to k layers (by applying pair-wise composition repeatedly.) We define a *serial k-community* to be a multilayer community where communities from k distinct connected layers of a HeMLN are combined in a specified order Technically, this should be expressed as $((\Psi(G_2)\ \Theta_{2,1}\ \Psi(G_1))\ \Theta_{2,3}\ \Psi(G_3)$.)

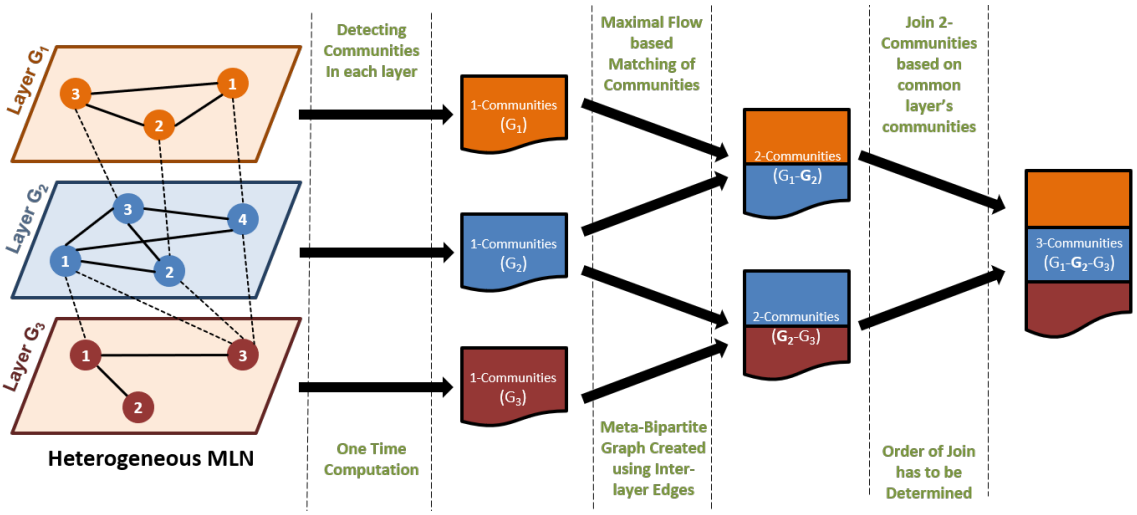


Figure 6.1: Illustration of decoupling approach for computing a 3-community $((G_2 \; \Theta_{2,1} \; G_1) \; \Theta_{2,3} \; G_3)$

However, we drop Ψ for simplicity. In fact, Θ with its subscripts is sufficient for our purpose due to pre-defined precedence (left-to-right) of Θ. We retain G for clarity of the expression. ω_e is a weight metric discussed in chapter weight metrics Our proposed decoupling approach for finding HeMLN communities is as follows;

(i) First use the function Ψ (here community detection) to find communities in each of the layers individually,

(ii) for any two chosen layers, construct a bipartite graph using their communities as meta nodes and create meta edges that connect the meta nodes (using an appropriate element of X) and assign weight (ω). ω reflects the number of edges constituting a meta edge as well as properties of participating communities as discussed in Section 8, and

(iii) compose the partial results from each layer by representing each community as a meta node of the bipartite graph and using a function Θ which computes 2-community as pairs using the *weight information of edges in the bipartite graph.*)

Figure 6.1 illustrates the decoupling approach for specifying and computing a community of a larger size from partial results. It illustrates how a set of distinct

communities from a layer is used for computing a 2-community (for 2 layers) and further a 3-community (for 3 layers) using partial results. 1-community is the set of communities generated for a layer (simple graph.)

6.1 Community Definition for a HeMLN

We first motivate the need for defining a structure- and semantics-preserving communities. For the IMDb data set, consider the HeMLM shown in Figure 6.1 and the analysis *"Find groups of actors for every director group such that the most versatile members interact?* Note that the actor and director layers can only compute groups of actors and directors, who act in or direct similar genre, respectively. The connection (or coupling) between directors and actors only come from inter-layer edges. It is only by preserving the structure of both the communities in actor and director as well as the inter-layer edges, can we compute the answer that indicates the semantics of which actor groups are paired with the director groups. The inter-layer edges preserve the relationships of individual actors and directors as well.

Clearly, multiple relationships can exist in such a collection of layers, such as co-acting, similar genres and who-directs-whom. An analysis requirement may also want to use *preferences* for community interactions. As an example, one may be interested in groups (or communities) where the *most important* actors and directors interact.

Our *definition of community for a HeMLN uses coupling of communities based on the connection strength (expressed as a weight) and is consistent with the simple graph definition of a community. Further, it also preserves the structure and semantics due to composition which is also shown to be efficient!.* Table 6.1 lists all notations used in the thesis and their meaning for quick reference.

$G_i(V_i, E_i)$	Simple Graph for layer i
$X_{i,j}(V_i, V_j, L_{i,j})$	Bipartite graph of layers i and j
$MLN(G, X)$	Multilayer Network of layer graphs (set G) and Bipartite graphs (set X)
Ψ	Analysis function for G_i (community)
$\Theta_{i,j}$	Maximum Weighted Bipartite Coupling (MWBC) function
$CBG_{i,j}$	Community bipartite graph for G_i and G_j
U_i	Meta nodes for layer i 1-community
$L'_{i,j}$	Meta edges between U_i and U_j
c_i^m	m^{th} community of G_i
$v_i^{c^m}$, $e_i^{c^m}$	Vertices and Edges in community c_i^m
$x_{i,j}$	{Expanded(meta edge $< c_i^m$, $c_j^n >$)}
0 and ϕ	null community id and empty $x_{i,j}$
ω_e, ω_d, ω_h	Weight metrics for meta edges

Table 6.1: Notations used in this thesis

6.1.1 Formal Definition of Community in a HeMLN

A ***1-community*** is a set of communities of the simple graph corresponding to a layer.

A **community bipartite graph** $\boldsymbol{CBG_{i,j}(U_i, U_j, L'_{i,j})}$ is defined between two disjoint and independent sets U_i and U_j. An element of U_i (U_j) is a 1-community id from G_i (G_j) that is represented as a single meta node. $L'_{i,j}$ is the set of meta edges between the nodes of U_i and U_j (or bipartite graph edges.) For any two meta nodes, a *single edge* is included in $L'_{i,j}$, if there is *an inter-layer edge* between any pair of

nodes from the corresponding communities (acting as meta nodes in CBG) in layers G_i and G_j. Note that there may be many inter-layer edges between the communities from the two layers. Also note that U_i (U_j) need not include all community ids of G_i (G_j.) This is a different bipartite graph between two sets of nodes (termed meta nodes) from two distinct layers that correspond to communities in each layer. A single bipartite edge (termed meta edge) is drawn between distinct meta node pairs as defined. $\boldsymbol{CBG_{i,j}(U_i, U_j, L'_{i,j})}$ is defined between two disjoint and independent sets U_i and U_j. An element of U_i (U_j) is a 1-community id from G_i (G_j) that is represented as a single meta node. $L'_{i,j}$ is the set of meta edges between the nodes of U_i and U_j (or bipartite graph edges.) For any two meta nodes, a *single edge* is included in $L'_{i,j}$, if there is *an inter-layer edge* between any pair of nodes from the corresponding communities (acting as meta nodes in CBG) in layers G_i and G_j. Note that there may be many inter-layer edges between the communities from the two layers. Also note that U_i (U_j) need not include all community ids of G_i (G_j.) The strength (or weight) component of the meta edges is elaborated in Section 8.

A ***serial 2-community*** is defined on the community bipartite graph $CBG_{i,j}(U_i, U_j, L'_{i,j})$ corresponding to layers G_i and G_j A 2-community is a set of tuples each with a pair of elements $< c_i^m, c_j^n >$, where $c_i^m \in U_i$ and $c_j^n \in U_j$, that satisfy the *Maximum Weighted Bipartite Coupling (MWBC)* or traditional approach (composition function Θ explained in 7) for the bipartite graph of U_i and U_j, along with the set of inter-layer edges between them. The pairing is done from left-to-right (hence it is *not commutative*) and a single community from the left layer can pair with *zero or several communities* from the right layer. That is, *one-to-many or many-to-one pairings* are possible. The lower bound on the number of 2-community is $|U_i| -$ *number of* U_i *nodes that have no outgoing edges.*

A ***serial k-community***k represents the number of layers used for computing the community, not the number of compositions. The **"serial"** prefix used for defining a k-community indicates the order used (but can be arbitrary) in its specification. A k-community corresponds to a connected subgraph of k layers. Our definition assumes left-to-right precedence for the composition function Θ. It is possible to define a k-community with explicit precedence specification for Θ. Also, other definitions are possible that may be order agnostic. For k layers of a HeMLN is defined as the application of *serial 2-community* definition recursively to compose a k-community. The base case corresponds to applying the definition of 2-community for any two layers. The recursive case corresponds to applying 2-community composition for a t-community with another G_j.

For each recursive step, there are two cases for the 2-community under consideration: i) the U_i is from a layer G_i *already in the t-community* and the U_j is from a *new layer* G_j. This bipartite graph match is said to **extend** a t-community (t < k) to a *(t+1)-community*, or ii) **both** U_i (U_j) from layers G_i (G_J) are *already in the t-community*. This bipartite graph match is said to **update** a t-community (t < k), **not extend** it.

In both cases i) and ii) above, a number of outcomes are possible. Either a meta node from U_i: a) matches one or more meta nodes in U_j resulting in a (or many) **consistent match**, or b) does not match a meta node in U_j resulting in a **no match**, or c) matches a node in U_j that is not consistent with a previous match termed **inconsistent match**.

Structure preservation is accomplished by retaining, for each tuple of t-community, either a matching community id (or 0 if no match) and $x_{i,j}$ (or ϕ for empty set) representing inter-layer edges corresponding to the meta edge between the meta nodes (termed **expanded(meta edge).)**

Space of Analysis Alternatives: Given a HeMLN with k layers, the number of possible k-community (or analysis space) is quite large. For a HeMLN-graph, the number of potential k-community is a function of the number of unique connected subgraphs of different sizes and the number of possible orderings for each such connected subgraph. With the inclusion of 3 weight metrics (see chapter 8), it gets even larger. It is important to understand that each subgraph of a given size (equal to the number of edges in the connected subgraph) along with the ordering represents a *different* analysis of the data set and provides a different perspective thereby supporting a large space of analysis alternatives. Finally, the composition function Θ defined above is not commutative (due to left-to-right pairing) and also not associative Due to the use of a subset of meta nodes rather than the entire 1-community during any recursive step.. Hence, for each k-community, the *order in which a k-community* is defined has a bearing on the result (semantics) obtained.

CHAPTER 7

ALGORITHMS

This chapter contains algorithms used to implement community detection in HeMLN. First algorithm, makes use of traditional approach for bipartite matching. The second algorithm, uses the proposed *Maximum Weight Based Coupling* algorithm for bipartite matching, which has property of one to many matching.

7.1 k-Community detection using traditional bipartite matching

Algorithm 1 is an iterative algorithm that accepts a linearized specification of a k-community and computes the result as described earlier. The input is an *ordering of layers*, *composition function indicating the community bipartite graphs to be used* and the type of weight to be used (elaborated below in 8) The output is a *set* whose *elements are tuples corresponding to distinct, single elements of k-community* for that specification. The size (i.e., number of tuples) of this set is bound by the base case. The layers for any 2-community bipartite graph composition are identifiable from the input specification.

The algorithm iterates until there are no more compositions to be applied. The number of iterations is equal to the number of Θ in the input (including Θ for the base case.) For each layer, we assume that its 1-community has been computed.

The bipartite graph for the base case and for each iteration is constructed for the participating layers (either one is new or both are from the t-community for some t) and maximal network flow algorithm is applied. The result is used to either extend or update the tuples of the t-community for all the cases as described in Table 7.2.

Note that the k-community size **k is incremented** only when a *new layer is composed (case i).)* For case ii) (cyclic k-community) **k is not incremented** when *both layers are part of the t-community.* When the algorithm terminates, we will have the set of tuples each corresponding to a single, distinct element of k-community for the given specification.

Algorithm 1 k-community Detection Algorithm (using traditional approach)

Require: -

1: **INPUT:** HeMLN, (G_{n1} $\Theta_{n1,n2}$ G_{n2} ... $\Theta_{ni,nk}$ G_{nk}), and a weight metric (wm).

2: **OUTPUT:** Set of tuples with two components

3: `input processed from left to right`

4: **Initialize:** k = 2, $U_i = \phi$, $U_j = \phi$, $L = \phi$

5: *result* $\leftarrow$ initialize(2-community(G_{n1},G_{n2}),HeMLN,wm)

6: *left* $\leftarrow$ next_left_subscript (Θ)

7: *right* $\leftarrow$ next_right_subscript(Θ)

8: **while** *left* $\neq$ null && *right* $\neq$ null **do**

9: $U_i \leftarrow$ subset of 1-community(G_{left})

10: $U_j \leftarrow$ subset of 1-community(G_{right}) `subsets (3,4) if layer has been processed`

11: $MP \leftarrow$ max_flow_pairs(U_i, U_j, HeMLN, wm)

12: `a set of pairs` $< c^p_{left}$, $c^q_{right} >$

13: **for each** tuple $t \in$ *result* **do**

14: **if** *both* c^x_{left} *and* c^y_{right} are part of t and $\in$ MP `[case ii (processed layer): consistent match]` **then**

15: Update t with ($x_{left,right}$)

---

else if c^x_{left} is part of t and $\in$ MP and G_{right} layer has been processed `[case ii (processed layer): no and inconsistent match]` **then**

  Update t with ϕ

else if c^x_{left} is part of t and $\in$ MP `[case i (new layer): consistent match]` **then**

  Extend t with paired $c^y_{right} \in$ MP and $x_{left,right}$

  k = k + 1

else if c^x_{left} is part of t and $\notin$ MP `[case i (new layer): no match]` **then**

  Extend t with 0 (community id) and ϕ

  k = k + 1

end if

end for

left $\leftarrow$ next_left_subscript (Θ) or null

right $\leftarrow$ next_right_subscript(Θ) or null

end while

(G_{left}, G_{right}) **outcome**	**Effect on tuple *t***
case (i) - one processed and one new layer	
a) consistent match	**Extend** t with paired community id **and** $x_{i,j}$
b) no match	**Extend** t with 0 and ϕ
case (ii) - both are processed layers	
a) consistent match	**Update** t **only with** x
b) no match	**Update** t **only with** ϕ
c) inconsistent match	**Update** t **only with** ϕ

Table 7.1: Cases and outcomes for a maximal network flow match (Algorithm 1)

Need for a new pairing algorithm: In a traditional bipartite graph (used for dating, hiring etc.), each node is a simple node. The goal is to find maximum number of matches (bipartite edges) such that no two matches share the same node. Hence, a node from one set is paired with at most one node from the other set. This has been extended to include weights for the edges without changing the pairing semantics [64]. On the other hand, for maximal network flow algorithms [65], a source and a sink is assumed and weights have to be given from source to each node which is impractical in our case.

In contrast, each meta node in our case is a community representing a group of entities with additional characteristics. For a k-community to be meaningful, we need to associate weights with edges to capture not only the number of edges but also characteristics of the participating communities as well. To capture this, we discuss

a number of alternatives for weights (termed weight metrics ω) in Section 8, derived from real-world scenarios.

For pairing nodes of the bipartite graph, since traditional approaches are not suited for our coupling, we propose a edge weight-based coupling which reflects the semantics of the community. Each node from the first set is paired with *zero, one or more nodes* from the second set solely based on the outgoing edge weights of that node. This is repeated for each node from the first set. *Most importantly, unlike current alternatives in the literature for community of a MLN, there is no information loss or distortion or hiding the effect of different entity types or relationships in our definition.*

7.2 k-Community detection using maximum weight based coupling approach

Algorithm 2 accepts a linearized specification of a k-community and computes the result as described earlier. The input is an *ordering of layers*, *composition function indicating the community bipartite graphs to be used* and the type of weight to be used. The output is a *set* whose *elements are tuples corresponding to distinct, single HeMLN k-community* for that specification. The size (i.e., number of tuples) of this set is determined by the pairs obtained during computation. The layers for any 2-community bipartite graph composition are identifiable from the input specification.

The algorithm iterates until there are no more compositions to be applied. The number of iterations is equal to the number of Θ in the input (corresponds to the number of inter-layer connections.) For each layer, we assume that its 1-community has been computed.

The bipartite graph for the base case and for each iteration is constructed for the participating layers (either one is new or both are from the t-community for some

t) and MWBC algorithm is applied. The result obtained is used to either extend or update the tuples of the t-community and add new tuples as well. All cases are described in Table 7.2. Note that the k-community size **k is incremented** only when a *new layer is composed (case i).)* For case ii) (cyclic k-community) **k is not incremented** when *both layers are part of the t-community.* When the algorithm terminates, we will have the set of tuples each corresponding to a single, distinct k-community for the given specification.

Algorithm 2 HeMLN k-community Detection Algorithm

Require: -

1: **INPUT:** HeMLN, (G_{n1} $\Theta_{n1,n2}$ G_{n2} ... $\Theta_{ni,nk}$ G_{nk}), and a weight metric (wm).

2: **OUTPUT:** Set of distinct k-community tuples

3: **Initialize:** k=2, $U_i = \phi$, $U_j = \phi$, result$' = \emptyset$

4: *result* $\leftarrow$ MWBC(G_{n1},G_{n2}, HeMLN, wm)

5: *left*, *right* $\leftarrow$ left and right subscripts of second Θ

6: **while** *left* $\neq$ null && *right* $\neq$ null **do**

7: $U_i \leftarrow$ subset of 1-community($G_{left}, result$)

8: $U_j \leftarrow$ subset of 1-community($G_{right}, result$)

9: $MP \leftarrow$ MWBC(U_i, U_j, HeMLN, wm)

10: `a set of comm pairs` $< c^p_{left}, c^q_{right} >$

11: **for each** tuple $t \in$ *result* **do**

12: kflag = false

13: **if** *both* c^x_{left} *and* c^y_{right} are part of t and $\in$ MP `[case ii (processed layer): consistent match]` **then**

14: Update *a copy of* t with ($x_{left,\ right}$) and append to result$'$

15: **else if** c^x_{left} is part of t and $\in$ MP and G_{right} layer has been processed `[case ii (processed layer): no and inconsistent match]` **then**

16: Update *a copy of* t with ϕ and append to result$'$

17: **else if** c^x_{left} is part of t and for each $c^x_{left} \in MP$ `[case i (new layer): consistent match]` **then**

18: copy and Extend t with paired $c^y_{right} \in$ MP and $x_{left,\ right}$ and append to result$'$; kflag = true

19: **else if** c^x_{left} is part of t and $\notin$ MP `[case i (new layer): no match]` **then**

20: copy and Extend t with 0 (community id) and ϕ and append to result$'$; kflag = true

21: **end if**

22: **end for**

23: *left*, *right* = next left, right subscripts of Θ or null

24: if kflag k = k + 1; result = result$'$; result$' = \emptyset$

25: **end while**

(G_{left}, G_{right}) **outcome**	**Effect on tuple *t***
case (i) - one processed and one new layer	
a) consistent match	**Copy & Extend** t with paired community id **and** $x_{i,j}$
b) no match	**Copy & Extend** t with 0 and ϕ
case (ii) - both are processed layers	
a) consistent match	**Copy & Update** t **only with** x
b) no match	**Copy & Update** t **only with** ϕ
c) inconsistent match	**Copy & Update** t **only with** ϕ

Table 7.2: Cases and outcomes for MWBC (Algorithm 2)

7.3 Weight Based Coupling Algorithm

The proposed algorithm, relaxes restrictions of traditional matching approach used in algorithm 1 and computes one to many matching. The Algorithm takes bipartite layer as an input. Bipartite layer is in the form of $< C_{mi}, C_{nj} >$, weight. Coupled Communities is the result set which is initialised to NULL at the beginning and community matchings are being added to it during the algorithm execution. For each edge in the bipartite layer, community of the left set of the layer is checked if it is present in coupled communities result set. If it is not present it is directly added to the result set. If it is already present, weight of previous coupling is compared with weight of the present coupling. As the algorithm can give out multiple coupling for left set of communities if the weights are equal, present community couple are added to result set. If the previous weight is less than the present weight, the previous coupling is replaced with present coupling.

Algorithm 3 Weight Based Coupling Algorithm

Require: - **INPUT:** U_i,U_j, L_{ij}, HeMLN, wm. **OUTPUT:** $< C_{mi}, C_{nj} >$ where C_{mi} belongs to U_i, C_{nj} belongs to U_j.

1: **INITIALISATION:** Coupled Communities = NULL
2: **for** each line in Lij $< Cmi, x >$ **do**
3: **if** C_{mi} belongs to Coupled Communities **then**
4: Previous Weight = get the weight of already coupled communities
5: **if** previous weight = weight of $< C_{mi}, C_{nj} >$ **then**
6: Add $< C_{mi}, C_{nj} >$ to Coupled Communities
7: **else if** previous weight $<$ weight of $< C_{mi}, C_{nj} >$ **then**
8: Replace previous C_{mi} couple with $< C_{mi}, C_{nj} >$
9: **end if**
10: **else** Add $< C_{mi}, x >$ to Coupled Communities
11: **end if**
12: **end for**

CHAPTER 8

HeMLN Community Detection using Customized WEIGHT METRICS

Algorithms defined uses a bipartite graph match with a given weight metric. As we indicated earlier, there is an important difference between simple and **meta nodes/edges** that represent a **community of nodes/set of edges**. Without including the characteristics of meta nodes and edges for the match, we cannot argue that the pairing obtained represents analysis based on participating community characteristics. Hence, it is important to identify how qualitative community characteristics can be mapped quantitatively to a weight metric (that is, weight of the meta edge in a community bipartite graph) to influence the bipartite matching.

The strength or weights can have different meaning in different situations. For example, in IMDb dataset, if one needs to know what group of actors interact the most with what groups of directors, weights can be determined by simply identify the number edges between every group of actors and directors. If the analysis objective changes, now one wants to also consider internal strength of each community of every layer(or meta nodes of bipartite layer), weight metric has to take into account meta node's density information. To address various considerations of weights during the analysis, we have define 5 weight matrices. Each weight matrices can be used to fetch the answer for different analysis question. Once the analysis is defined the by the user, he/she can decide which weight matrices fits perfectly to achieve the goal and proceed with the analysis.

8.1 **Simple Edge Weighted Metrics** (ω_e)

This weight metrics is determined by counting the number of edges between two meta nodes of a community bipartite layer. The edges between two communities are the edges between internal vertices of two communities. Lets consider IMDb data set, community bipartite layer is formed by community of actors and community of directors. The weight for this community bipartite layer is measured by counting the edges between each actor node of a actor community and each director node of director community. These are the inter layer edges between actor and director layer. In the figure 4.1. Actor communities and director communities make two sets of nodes in the bipartite layer. It consists of Inter layer edges between actor and director vertices are shown. Actor community A1 and director community D1 has two edges between them, hence weight is 2. Similarly there is one edge between A2 and D1, one edge between A1 and D2. We can represent simple edge weighted community bipartite graph as follows: If one wants to find for each actor community a director

Actor Community	**Director Community**	**Weight**
A1	D1	2
A1	D2	1
A2	D1	1
A2	D2	1
A3	D2	2

Table 8.1: Simple Edge Weights

community with which it most interacts we can use simple edge weighted metrics. Here interactions are determined by the edge count.

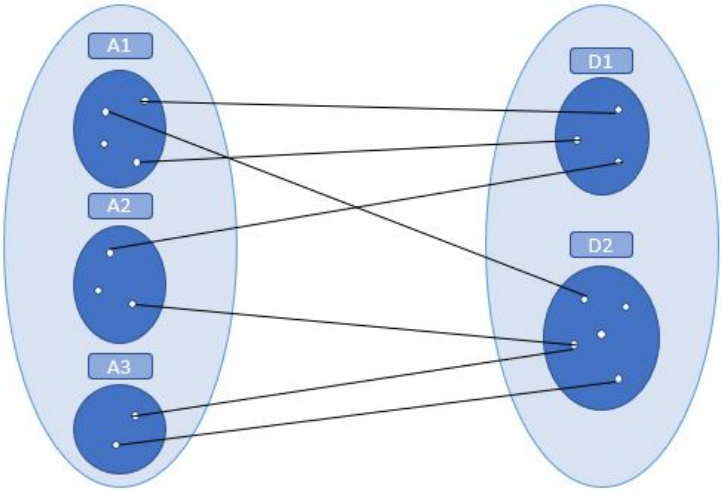


Figure 8.1: Actor Director Bipartite Graph

8.2 Edge Fraction Weighted Metrics(ω_f)

Edge fraction metrics not only considers number of edges between two meta nodes but also the number of vertices present in each meta nodes. By taking into account number of vertices in each meta nodes, one can calculate total number of possible edges between those meta nodes. Having more number of vertices in the meta nodes and less edges between them decreases the strength of the coupling between meta nodes. Hence, The fraction shows the actual strength of the coupling between two meta nodes. Formula for calculating edge fraction weight is as follows:

$$Edge\ fraction\ weight = \frac{Number\ of\ edges\ between\ meta\ nodes}{V_1\ *(V_2)} \tag{8.1}$$

Where,

V1 is the number of vertices in meta node of layer 1

V2 is the number of vertices in meta node of layer 2

For the figure 8.1, edge fraction weights would be as follows:

Actor Community	**Director Community**	**Weight**
A1	D1	0.166
A1	D2	0.05
A2	D1	0.11
A2	D2	0.066
A3	D2	0.2

Table 8.2: Edge Fraction Weights

8.3 Participation Weighted Metrics(ω_p)

participation weight captures number of participating nodes between the meta nodes. This is to differentiate situation where a single node is participating to form multiple edges from, multiple nodes are participating to form multiple edges. Participation weight takes into node participation and edge fraction to derive the strength of the connectivity. Hence this weight gives more weightage to such meta nodes where more nodes are participating in the interactions. Formula used to derive participation weight is:

$$participation\ weight = \frac{P_1}{V_1} \ * \ Edge\ fraction * \frac{P_2}{V_2} \tag{8.2}$$

Where,

P1 is the number of participating vertices in meta node of layer 1

V1 is the number of vertices in meta node of layer 1

P2 is the number of participating vertices in meta node of layer 2

V2 is the number of vertices in meta node of layer 2

Actor Community	Director Community	Weight
A1	D1	0.0053
A1	D2	0.0025
A2	D1	0.0012
A2	D2	0.0044
A3	D2	0.02

Table 8.3: Edge Fraction Weights

8.4 **Density Weighted Metrics** (ω_d)

The intuition behind this metric is to not only bring the edge contribution as a fraction (instead of the total number of edges as in ω_e.), but also participating community characteristics. Density which captures internal structure of a community is used. Clearly, *higher the densities and larger the edge fraction, the stronger is the interaction between two meta nodes (or communities.)* Since each of these three components (each being a fraction) increases the strength of the inter-layer coupling, they are multiplied to generate the weight of the meta edge. The domain of this weight will be $(0, 1]$. Formally, using the density formula,

$$density\ weight = Community\ Node1\ density * edge\ fraction * Community\ Node2\ density \tag{8.3}$$

$$density\ weight = \frac{E_1}{V_1\ *(V_1-1)}\ *\ Edge\ fraction *\ \frac{E_2}{V_2\ *(V_2-1)} \tag{8.4}$$

Where,

E1 is the edges in meta node of layer 1

V1 is the number of vertices in meta node of layer 1

E2 is the number of edges in meta node of layer 2

V2 is the number of vertices in meta node of layer 2

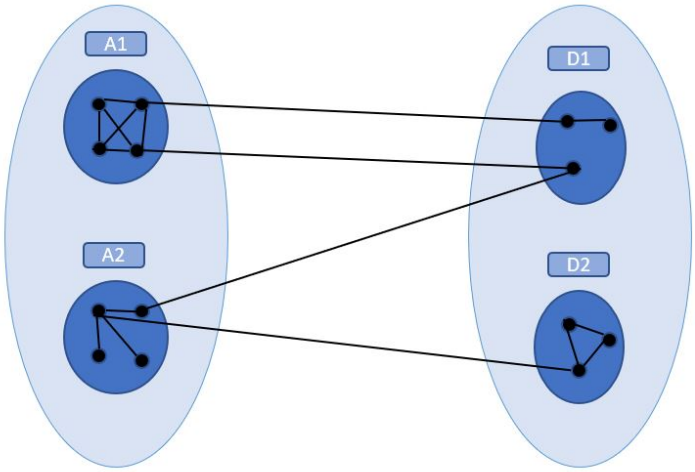


Figure 8.2: Actor-Director Bipartite Graph

For the figure 8.2 following are density weights for actor-director bipartite graph.

Actor Community	**Director Community**	**Weight**
A1	D1	0.0277
A2	D1	0.0034
A2	D2	0.04166

Table 8.4: Density Weights

8.5 Hub Participation Weighted Metrics (ω_h)

The ω_e metric captures only interaction between two communities and the metric ω_d includes the effect of community structure, but not the characteristics of the nodes that interact. (ω_p) captures number of participating nodes, but doesn't show how important those participating nodes are. Typically, we are interested in knowing whether highly influential nodes within a community also interact across the commu-

nity. This can be translated to the *participation of influential nodes within and across each participating community* for analysis. This can be modeled by using the notion of hubHigh centrality nodes (or hubs) have been defined based on different metrics, such as degree centrality (vertex degree), closeness centrality (mean distance of the vertex from other vertices), betweenness centrality (fraction of shortest paths passing through the vertex), and eigenvector centrality. **participation** within a community and their interaction across layers. In this paper, we have used degree centrality for this metric to connote higher influence. Again, ratio of participating hubs from each community and the edge fraction are multiplied to compute ω_h.

$$Hub\ participation\ weight = H_1\ \ *\ Edge\ fraction *\ H_2 \tag{8.5}$$

Where,

H1 is hub participation in meta node of layer 1

H2 is hub participation in meta node of layer 2

$$Hub\ participation\ = \frac{Participating\ Hubs\ in\ meta\ node}{Total\ Hubs\ in\ meta\ node} \tag{8.6}$$

table 8.5 shows the Participating hub weights are as follows for the diagram 8.2

Actor Community	**Director Community**	**Weight**
A1	D1	0.06925
A2	D1	0(No hubs are participating)
A2	D2	0.01388

Table 8.5: Participating Hubs Weights

CHAPTER 9

IMPLEMENTATION DETAILS

This chapter contains the implementation details of each method used to model and analyse the heterogeneous multilayer graphs. First step before executing this program is to build the layers of graphs. Any dataset has to be cleaned and preprocessed into layers of graphs. For example, IMDb dataset is in the form of CSV files which contains various information. Only relevant information and feature has to be picked which can satisfy the business requirements or analysis objectives. For IMDb dataset, actor, genres the actors worked in, director, genres the directors worked in, movie, rating of the movie, has been picked to build the layers. These individual layers can be used to compute communities using Infomap [5] or Louvain [6]. Decoupling approach is applied to find communities across layers.

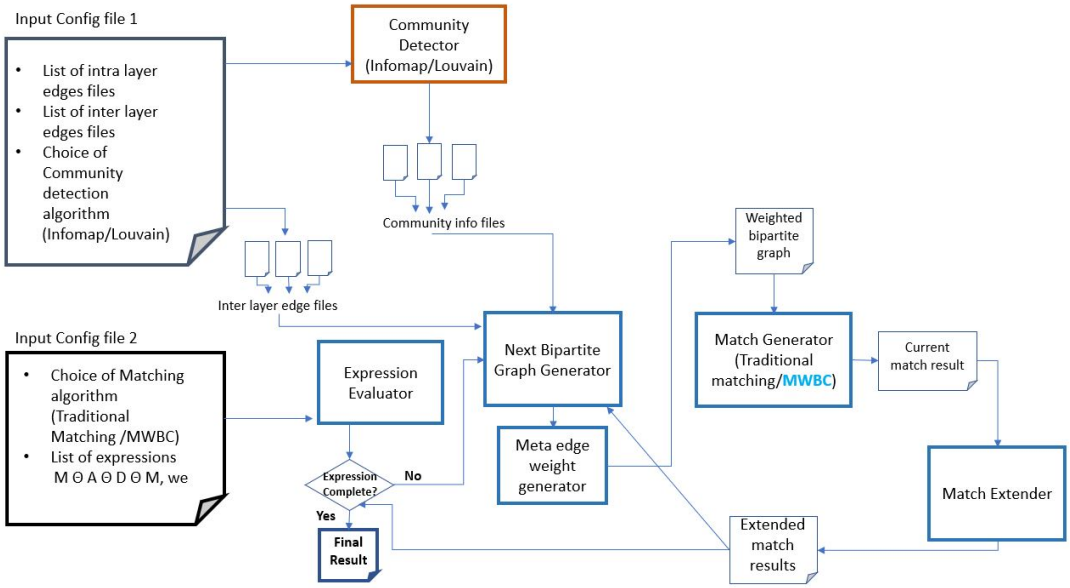


Figure 9.1: HeMLN Community Computation Architecture and Flow

The above architecture shows the flow of the implementation and process involved in community computation of HeLMN. The driver function takes two input configuration file as input. First configuration file contains multi layer network in the form of inter layer edges and intra layer edges. User can mention his/her choice of community detection algorithm. Based upon user's choice, appropriate community detection algorithm(Infomap/Louvain) will be invoked. **Expression Evaluator** process the second configuration file contains equations to be evaluated which represents the analysis objective. Expression contains weight metrics to be applied. This file also contains the choice of matching algorithm to be applied(Traditional Maximal Matching/Maximal Weighted Bipartite Coupling). **Next Bipartite Graph Generator** creates a skeleton for the bipartite graph and **Meta edge weight generator** adds weight mentioned in the configuration file. **Match Generator** process the weighted bipartite graph and applies the matching algorithm. Matching algorithm, traditional maximal matching or maximal weight bipartite coupling is applied. If matching algorithm has to be applied for more than 2 layers, **Match Extendor** takes into account the previous matches obtained to extend it to the new matches obtained in the new layers.

These components are implemented using Python 3 programming. Below are the methods implemented for each above described components.

9.1 Building Edge Weighted Bipartite Graph

Edge weights are easiest weight metrics to compute. Edge weights denote the number edges between two nodes present in a bipartite set as explained in the weight metrics section. As edge weight do not include any characteristics of a inner community structure, intra-layer edges have no significance in this metrics. Only inter-layer edges between two layers has to be considered. For example, to compute edge met-

rics between actor and director layer, actor-director inter-layer edges are considered. As the calculation of edge weights are on meta node, that is, a community becomes a meta node in the bipartite graph, a node's community information is considered. Hence for this method, inter-layer edges file and community info file are given as input. A result file containing community number bipartite set 1, community number bipartite set2 and number of edges between the communities is produced. In this program, three methods are implemented and are as follows:

9.1.1 Method: mapNodesToCommunityNumber

This method as name given, maps each node to the community it belongs to. It takes community info file as input. It reads the file and creates a dictionary called *communityInfo*. Dictionary key is node number and value is community number it belongs to. This method takes O(|V|) time as each vertices in the community info file has to be mapped to it community number.

9.1.2 Method: getCommunityInfo

Given a community dictionary and node number, returns the community number the node belongs to.

9.1.3 Method: computeEdgeWeights

This method, takes intra-layer info file as input. This file contains node number of layer 1, node number of layer 2 which denotes that there exists an edge between between those two nodes. Method reads the file line by line and calls getCommunityInfo, and gets community number for each node. It creates a dictionary called *edgesBetweenCommunities* which contains a tupple (community number 1, community number 2) where community number 1 is the community number of layer 1

similarly community number 2 is the community number of layer 2 as key and the edges between them as value. As a result it creates a edge weighted bipartite layer file using the *edgesBetweenCommunities*. This method takes O(|E|) time where E denotes the intra-layer edges. getCommunityInfo takes O(1) as it is simply retrival of value from dictonary.

9.2 Building Edge Fraction Weighted Bipartite Graph

Edge fraction metrics not only considers edges between two meta nodes but also number of vertices in each meta node. This method needs to compute number of vertices in each community using community info file. Most of the implementation for metrics edge fraction is same as edge weighted metrics. Implementation details is as follows:

9.2.1 Method: mapCommunityInfo

This method takes community info file as input and creates two dictionaries. Dictionary *communityNumber* contains vertices as key and community it belongs to as value. Dictionary *numberofVerticesInCommunity* has community number as key and number of vertices as value.

9.2.2 Method: getCommunityInfo

This method has two use cases. If *communityInfo* dictionary and node number is passed, it gives out community number the node belongs to. If *numberofVertices* dictionary and community number is sent, it returns the number of vertices present in that particular community.

9.2.3 Method: computeEdgeFractionWeights

This method is similar to the method computeEdgeWeights. It creates dictionary called *edgesBetweenCommunities*. While creating the result file, it computes edge fraction by getting number of vertices of each meta node and number of edges between the two meta nodes.

$$edge\ fraction\ weight = \frac{Number\ of\ edges\ between\ metanode1\ and\ metanode2}{number\ of\ vertices\ in\ metanode1\ *\ \ number\ of\ vertices\ in\ metanode2} \tag{9.1}$$

The above formula is applied. The resultant edge fraction bipartite file contains meta node1, meta node 2, edge fraction weight.

9.3 Building Density Weighted Bipartite Graph

In density weighted metrics, both intra layer edge characteristics and individual meta node's characteristics is considered. These functionalities' implementation is similar to edge fraction metrics computation. **Method: mapCommunityInfo**, **Method: getCommunityInfo**, **Method: computeEdgeFractionWeights** are adopted from above.

9.3.1 Method: computeDensityWeights

This method computes density of each meta nodes in the layer. It takes layer info file as input and counts the edges in each meta node. Density is calaculated using the formula:

$$Density = \frac{sum\ of\ edges\ present\ in\ meta\ node}{all\ possible\ edges\ in\ meta\ node} \tag{9.2}$$

$$\begin{aligned} all\ possible\ edges\ in\ meta\ node = sum\ of\ nodes\ in\ meta\ node* \\ (1\ - sum\ of\ nodes\ in\ meta\ node) \end{aligned} \tag{9.3}$$

Calculated density is stored in a dictionary called *densityWeights* where meta node number is a key and its density is value. The output file can be created by applying the following formula using density.

$$D\,W = Density\ of\ meta\ node\ 1\ *\ edge\ fraction\ * Density\ of\ meta\ node\ 2 \quad (9.4)$$

The result file density weighted bipartite layer is created, which contains, meta node 1, meta node 2, density weight.

9.4 Building Participation Weighted Bipartite Graph

In participation weighted metrics, edge fraction is considered similar to the above to metrics and it also considers number of participating nodes between the two meta nodes. Methods such as **computeEdgeFractionWeights**, **mapCommunityInfo** and **getCommunityInfo** remains the same.

9.4.1 fMethod: computeParticipationWeights

This method takes intra-layer edges file as input. It creates two dictionaries, *layer1ParticipationWithLayer2* and *layer2ParticipationWithLayer1*. These dictionaries contains a tupple of (meta node of layer1, meta node of layer 2) as key and list of nodes from (meta node of layer1) participating with (meta node of layer2). Number of participating nodes from meta node of layer 1 to meta node of layer 2 is nothing but the length of the list the dictionary holds as value. To compute this, the method has to scan entire intra-layer edges file hence it takes O(| E |) time where E is the intra layer edges. Once the information is scanned and stored in the form of dictionary, the participation weight formula is applied to compute participation weight(PW).

$$P\,W = \frac{participating\ nodes\ in\ meta\ node\ 1}{total\ number\ of\ nodes\ in\ meta\ node\ 1} * \ edge\ fraction * \frac{participating\ nodes\ meta\ node\ 2}{total\ number\ of\ nodes\ in\ meta\ node\ 2} \quad (9.5)$$

The resultant file called participation weighted bipartite graph is generated which will have results in the format: meta node 1, meta node 2, participation weight

9.5 Building Participating Hubs Weighted Bipartite Graph

In participating hubs metrics, participation of a node is considered only if it qualifies to be a hub. A hub node has greater degree than the average degree of its meta node. Hence, we need to calculate degree of each node in a meta node and average degree of the meta node. Other computations to achieve participating hubs weight can be adopted from previous methods. Methods **computeEdgeFractionWeights**, **mapCommunityInfo** , **getCommunityInfo** and **Method: computeParticipationWeights** are as is explained above.

9.5.1 Method: computeDegree

This method takes layer info file as input. It scans the input file's each line to creates a dictionary called *degreeOfEachNode*. In this dictionary, node number acts as key and its degree acts the value. For each edge encountered its vertices' degree is incremented. Each meta node's average degree is computed by computing

$$average\ degree\ =\ \frac{sum\ of\ degree\ of\ nodes\ of\ a\ meta\ node}{number\ of\ node\ in\ a\ meta\ node} \tag{9.6}$$

and is stored in dictionary *communityAverageDegree* where meta node number is the key and its average degree is the value. To check if a node in a meta node is a participating hub, dictionary *degreeOfEachNode* is iterated and degree is compared with it's meta node's average degree. It's meta node's number and avergae degree is fetched from **getCommunityInfo** by passing appropriate dictionary. If the node qualifies as a hub, the hub count of that meta node is incremented and stored in dictionary called *communityHubCount*. The qualifying hub is also checked if it is

participating with other layer's meta node. Hubs participation is registered in another dictionary called *participatingHubs* where a tupple (meta node1, meta node2) acts as key and its count acts as value. The result weight is calculated by applying the formula:

$$HW = \frac{participating\ hubs\ in\ meta\ node\ 1}{total\ number\ of\ hubs\ in\ meta\ node\ 1} * edge\ fraction * \frac{participating\ hubs\ in\ meta\ node\ 2}{total\ number\ of\ hubs\ in\ meta\ node\ 2} \tag{9.7}$$

The resultant file is created which contains result in the format meta node 1, meta node2, participating hubs weight.

9.6 **Fining community structure in multi-layer bipartite graphs**

9.6.1 Traditional Matching

Once the desired weighted bipartite graph is created, Maximum Weight Matching can be applied to get the best matching of meta nodes. maximum weight matching is the problem of finding, in a weighted graph, a matching in which the sum of weights is maximized. The algorithm has many implementations and is being extensively. Hence, an already implemented algorithm has been used in this thesis. A package called Networkx provides an efficient implementation and it also has many functionalities that is most appropriate for the input. In Networkx, maximum weighted matching is based on the blossom method for finding augmenting paths and the primal-dual method for finding a matching of maximum weight, both methods invented by Jack Edmonds. [60]

This method takes bipartite file in the form, meta node1, meta node2, weight as input. The first step is to create a bipartite graph from the input file. Inbuilt method, Graph() present in the Networkx creates a graph and graph.add_node() can be used to add vertices/nodes can be added to graph. graph.add_weighted_edges_from(E)

method can be used to add edges to the graph where E is list of edges acts as input. While adding the edge, bipartite set number is also has to be mentioned. max_weight_matching(G) inbuilt method takes Graph G of type Networkx as input and the matching is returned as a dictionary such that mate[v] == w if node v is matched to node w. Unmatched nodes do not occur as a key in mate. Networkx package also offers plot method to represent the matches in the form of figures.

9.6.2 Maximum Weight Based Coupling

This section includes the implementation detail of the proposed algorithm *Maximum Weight Based Coupling* for finding best matches across the bipartite layer. The algorithm, is simple and has been implemented efficiently. Program is called by inputting layers of graphs and a weight metrics to be considered. From the provided weight metrics, one of the above described methods are called to construct the bipartite graph. For example if Actor layers, Director layers and density was given as weight metric, density weighted actor-director bipartite layer will be generated.

Bipartite layer consists information in the format: *actor meta node, director meta node, density weight*. This file is processed line by line. A dictionary called *layerMatchingTable* is created where the layer 1 meta node is the key and value is a list containing weight in its first index following by all the matched layer two meta nodes. Layer 1 meta node is matched with more than layer 2 meta nodes if there the weight between those nodes are equal. Input might consist of more than two layers of graphs. Hence, another list called *matchedCommunities* is maintained which contains previously matched communities of already processed layers. For example, if actor layer, director layer and movie layer is considered and density is given as a metrics, density weighted Actor-Director layer is processed. As this is the first iteration, all the actor meta nodes are considered as *matchedCommunities* for actors is empty. After

processing, matched director meta nodes are stored in the *matchedCommunities* list which is taken into account while processing Density weighted Director-movie bipartite graph. At this time only director meta nodes present in the *matchedCommnuties* list is considered for processing the weight based coupling algorithm. Once all the given input layer is processed, output is produced which represents the heterogeneous communities across the layers.

This method scans the entire generated bipartite layer, which contains all the edges between two layers, hence take O(| E |) time.

As it has to process every layer given in the input time taken would be

O(N-1)(| E |)

where N is the number of input layer and N-1 is number of generated bipartite layer.

CHAPTER 10

EXPERIMENTAL ANALYSIS

10.1 IMDb, Random 1000 Movie Data Set

Actors(A) forms the first layer multilayer network where two actors are connected if they have acted in the same movies. Similarly the second layer is formed by the directors(D) and two directors are connected if the genres of movies they directed are similar above the threshold. Third layer is built using movies(M). Two movies are connected if they fall in the same range of ratings. The range of ratings are defined as 0-2, 2-4, 4-6, 6-8, 8-10. Inter layer edges are also present in the multilayer graphs. An actor is connected to the director, if they have worked together. A director is connected to a movie if he/she directs that movie. An actor is connected to the movie, if he/she have worked in that movie. A strongly connected group of actors are formed in the actor layer if there is many overlaps in the genres the actors have worked in. Same can be seen in the director layer. In the movie layer, all movies of a range are connected to each other forming cliques of movies. For the purposes of this analysis, we generated disjoint 1-community for each individual layer. with a setting which assigns any node to at most one community (i.e. computes disjoint communities). Infomap works in a hierarchical fashion while optimizing the *map equation*, which exploits the *information-theoretic duality* between the problem of compressing data, and the problem of detecting and extracting significant patterns or structures (communities) within those data based on *information flow*. The statistics for each layer are shown in Table 10.1.

A number of insights can be gleaned from layer analysis. The average actor community size are 5.3. There is only 1 large group (51) of co-actors. Further, 790 (92%) clique communities indicate that actors who co-act do it with the same group. In contrast, the number of communities is proportionately less (with respect to the number of nodes) in the director layer indicating that directors do not limit themselves to directing a small set of genres. The lower number of communities in the Movie (rating) layer is expected as it is a layer of 5 non-overlapping rating ranges. Also, 35% of movies (major chunk) have ratings in the range [6-8), A small number of movies (community M5) has the lowest range of rating [0, 2). Also, we know that all of them are cliques.

	Actor	**Director**	**Movie**
#Nodes	4588	1091	1000
#Edges	10255	91991	87492
#Communities (Size > 1)	862	18	5
Avg. Community Size	5.3	60.61	132.5

Table 10.1: 1-community Statistics for IMDb, random 1000 movies data set

10.1.1 **Analysis objectives result:**

For all the IMDb, random 1000 movie data set analysis objectives *Infomap community detection algorithm has been used. To map the communities between two layers,* maximum weight matching algorithm has been applied which is, one to one mapping of communities.

A(1) **Find co-actor groups that have *maximum interaction* with director groups who have directed similar genres?**

2-community: A $\Theta_{A,D}$ D; ω_e; order does not matter.

A(2) **Identify the *strongly connected* co-actor groups where *most of the actors have worked with most of the directors* who have directed similar genres**

2-community: A $\Theta_{A,D}$ D; ω_d; order does not matter.

A(3) **Identify *versatile* director groups who work with *most sought after* actors among co-actors**

2-community: A $\Theta_{A,D}$ D; ω_h; order does not matter.

A1, A2 and A3 analysis results are as follows:

A(1), ω_e	
c_A	c_D
A6	D1
A252	D2
A577:c	**D3**
A2	D4
A555:c	**D5**
A100:c	D6
A374:c	**D7**:c
A484:c	**D8**
A10	D9
A683:c	D10
A83:c	**D11**
A89:c	**D12**:c
A46:c	**D13**:c
A161:c	D14:c
A220:c	**D15**
A85:c	D16:c
A188	D17:c
A53:c	D18:c

(a)

A(2), ω_d	
c_A	c_D
A511:c	D1
A204:c	D2
A577:c	**D3**
A483:c	D4
A555:c	**D5**
A332:c	D6
A374:c	**D7**:c
A484:c	**D8**
A158:c	D9
A683:c	D10
A83:c	**D11**
A89:c	**D12**:c
A46:c	**D13**:c
A824:c	D14:c
A220:c	**D15**
A310:c	D16:c
A330:c	D17:c
A793:c	D18:c

(b)

A(3), ω_h	
c_A	c_D
A511:c	D1
A204:c	D2
A577:c	**D3**
A483:c	D4
A555:c	**D5**
A332:c	D6
A374:c	**D7**:c
A484:c	**D8**
A381:c	D9
A828:c	D10
A83:c	**D11**
A89:c	**D12**:c
A46:c	**D13**:c
A824:c	D14:c
A220:c	**D15**
A310:c	D16:c
A330:c	D17:c
A793:c	D18:c

(c)

All Communities, c indicates a clique, 44.4% common pairings (862 A Communities, 18 D Communities)

A(1), ω_e	
c_A	c_D
A374	**D7**
A89	**D12**
A46	**D13**
A161	D14
A85	D16
A330	**D17**
A53	D18

(d)

A(2), ω_d	
c_A	c_D
A374	**D7**
A89	**D12**
A46	**D13**
A824	D14
A310	D16
A330	**D17**
A793	D18

(e)

A(3), ω_h	
c_A	c_D
A374	**D7**
A89	**D12**
A46	**D13**
A824	D14
A310	D16
A330	**D17**
A793	D18

(f)

Clique Communities only, 57.14% common pairings (790 A Communities, 7 D Communities)

With 3 layers and 3 weight metrics, there are a total of 9 possible 2-community specifications. As the movie-rating layer has very few (and only clique) communities, we have chosen the A and D layers (with A as the left set of the bipartite match) for analysis using different metrics as shown in A(1) to A(3). As each metric is based on a different intuition for inter-layer coupling, they are expected to give different results.

Table 10.2a, 10.2b, and 10.2c show the results of 2-community as community pairings, respectively, for each metric. When the entire 1-community is used for each layer, **44.4% pairings (8 out of the 18) (marked in green)** are the *same across metrics*. In all the common pairings, at least one of the participating community is a clique.

To understand the analysis differences across ω, we grouped non-clique (density < 1) communities from each layer for analysis. Table 10.2a-10.2c shows the results for non-clique communities from each layer. Just **one common pairing** is found (marked in **cyan**) **validating the uniqueness of proposed metrics.**

A(1), ω_e	
c_A	c_D
A6	D1
A252	**D2**
A1	D3
A2	D4
A187	D6
A4	D8
A10	D9
A17	D11
A129	D15

(a)

A(2), ω_d	
c_A	c_D
A59	D1
A252	**D2**
A257	D3
A190	D4
A293	D6
A246	D8
A10	D9
A122	D11
A129	D15

(b)

A(3), ω_h	
c_A	c_D
A59	D1
A252	**D2**
A184	D3
A190	D4
A293	D6
A4	D8
A16	D9
A122	D11

(c)

Non-clique Communities only, 11.11% common pairings (72 A Communities, 11 D Communities)

Table 10.2: 2-community results for (A $\Theta_{A,D}$ D)

For completeness, Table 10.2d, 10.2e, and 10.2f show the community pairings when only cliques are used from each layer. Every matched pair that appears for ω_e (Table 10.2d) and

We know that the metric ω_e does not depend on the community characteristics, such as density and hub participation, unlike ω_d and ω_h. This can also be validated from the results. All the actor communities that are part of matches for ω_d (Table 10.2b) and ω_h (Table 10.2c) are cliques, marked by *'c'*, (which should be the case.) However, for the Table 10.2a which uses ω_e, it is not the case validating our intuition. Even the removal of cliques does not effect the non-clique matches that were obtained for ω_e (Table 10.2a and Table 10.2a.)

A(4) **For the group of directors (who direct similar genres) having *maximum interaction* with members of co-actor groups, identify the *most popular rating* for the movies they direct?**

Acyclic 3-community: A $\Theta_{A,D}$ D $\Theta_{D,M}$ M; ω_e

Figures 10.1 and A(5) show the 3-community analysis results diagrammatically for A(4) and A(5), respectively. For analysis A(4), 18 elements (consistent matches) are obtained after the first composition, bounded by the 18 communities in the director layer. For the second composition, although all the 18 communities from layer director are carried over (as they paired during base case), only 5 produce consistent matches with the movie layer, to get extended to become total elements (**bold blue line**), whereas, for the other 13 there was a no match (broken blue lines), thus becoming partial elements.

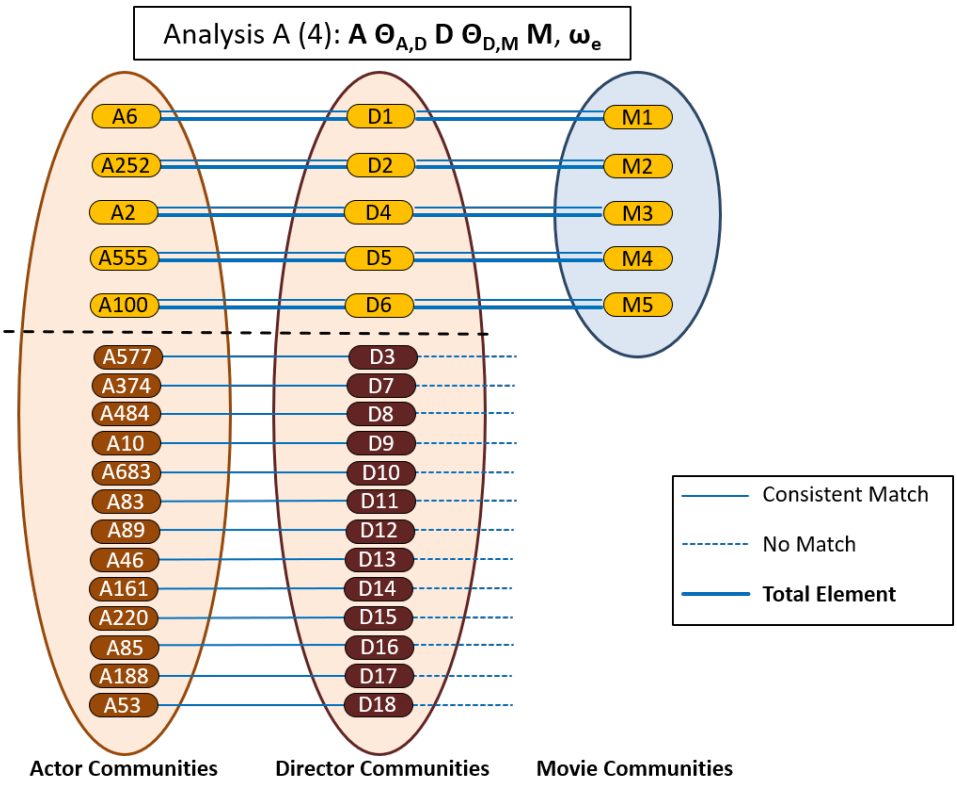


Figure 10.1: A4 Results: Acyclic 3-community (**total and partial**)

A(5) **For the *most popular* actor groups from each movie rating class, which are the director groups with which they have *maximum interaction*?**

Acyclic 3-community: M $\Theta_{M,A}$ A $\Theta_{A,D}$ D; ω_e

For A(5), every pairing from the first composition got extended in the second composition to produce 5 total elements (**bold blue lines**), bounded by movie communities. Moreover, for both the result sets, none of the total elements overlap. Thus, **the *order*** in which k layers are specified to obtain k-community determines the set of partial and total elements, providing insights corresponding to analysis specification.

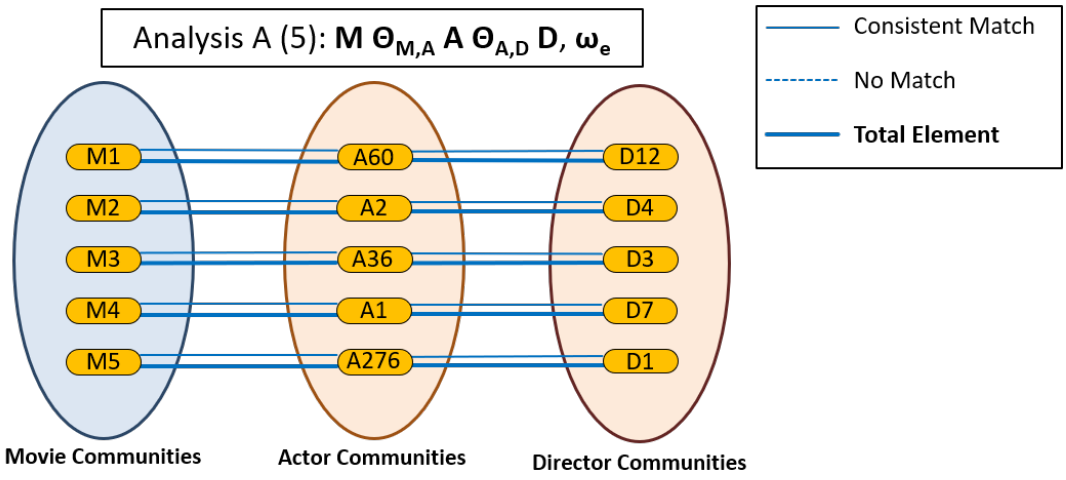


Figure 10.2: A5 Results: Acyclic 3-community (**no partial elements**)

A(6) **Find the co-actor groups with *strong movie ratings* that have *high interaction* with those director groups who also make movies with similar ratings (as that of co-actors.)**

Cyclic 3-community: M $\Theta_{M,A}$ A $\Theta_{A,D}$ D $\Theta_{D,M}$ M; ω_e

Results of each successive pairings (there are 3) are shown in Figure 10.3 using the same color notation. Coupling of movie-actor pairs (first composition) results in 5

consistent matches bounded by the movie layer. **Also, it is easy to see from the figure that only one of them continues and becomes a total element for the cyclic 3-community (bold blue triangle.) Further, the final result is an extension (M-A-D-M instead of M-A-D) of the acyclic 3-community result seen in Figure 10.2.** When the base case is extended to the director layer (second composition), we got 5 consistent matches as can be seen in Figure 10.2. The final composition to complete the cycle uses the 5 communities of director layer and 5 communities of the movie layer as left and right sets of community bipartite graph, respectively. **Only one consistent match is obtained to generate the total element (M2-A2-D4-M2) for the cyclic 3-community.** It is interesting to see 3 inconsistent matches (red broken lines) between the communities which clearly **indicate that all couplings are not satisfied by these pairs.** These result in 3 partial elements (M3-A36-D3, M4-A1-D7 and M5-A276-D1.)

The inconsistent matches also highlight the importance of order which is fundamental to our k-community definition for analysis. If a different order had been chosen (viz. director and actor layer as the base case), the result could have included the inconsistent matches. In this example, we also see *one* no match (broken blue line) in the final step, where D12 does not get matched to any movie community, thus generating the partial element, M1-A60-D12.

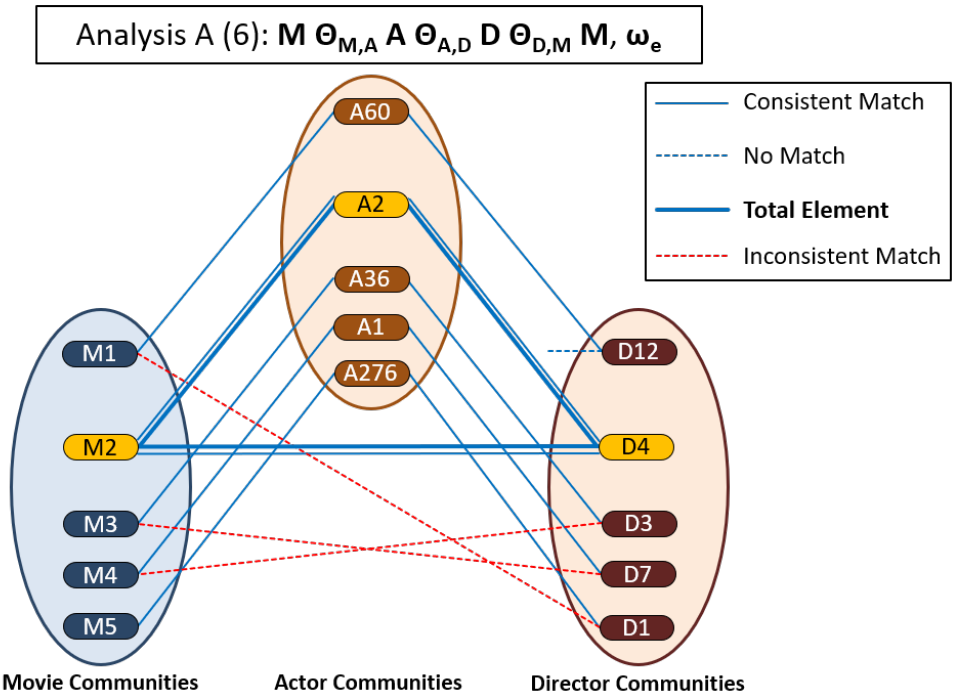


Figure 10.3: A6 Results: Progressive results of a cyclic 3-community

10.2 IMDb, Top 500 Actors Data Set

Actors(A) forms the first layer multilayer network where two actors are connected if they have acted in the movies with similar genres. There should be a match of more than or equal to 50% in list of genres. Similarly the second layer is formed by the directors(D) and two directors are connected if the genres of movies they directed are similar above the 50% threshold. Third layer is built using movies(M). Two movies are connected if they fall in the same range of ratings. The range of ratings are defined as 0-2, 2-4, 4-6, 6-8, 8-10. Inter layer edges are also present in the multilayer graphs. An actor is connected to the director, if they have worked together. A director is connected to a movie if he/she directs that movie. An actor is connected to the movie, if he/she have worked in that movie. **Individual Layer Statistics**: Table 10.3 shows the layer-wise statistics for IMDb HeMLN. 63 Actor

(A) and 61 Director (D) communities based on similar genres are generated. Out of the 10 ranges (communities) in the movie (M) layer, most of the movies were rated in the range [6-7), while least popular rating was [1-2). No movie had a rating in the range [0-1).

	Actor	**Director**	**Movie**
#Nodes	9485	4510	7951
#Edges	996,527	250,845	8,777,618
#Communities (Size > 1)	63	61	9
Avg. Community Size	148.5	73	883.4

Table 10.3: IMDB HeMLN Statistics

10.2.1 **Analysis Objectives Results**

All IMDb, top 500 actor data set, analysis objectives have been experimented with *Louvain community detection algorithm. To map the communities between two layers, traditional matching algorithm has been applied. It is an one to one mapping algorithm.*

A(7) ***Based on similarity of genres, which are the actor groups whose members have maximum interaction with the director groups?***

2-community: $A\ \Theta_{A,D}\ D;\ \omega_e$

For A(7), 49 A-D (Actor-Director) similar genre-based community pairs are obtained, where most actor-director pairs have interacted with each other at least once. Intuitively, a group of actors that prominently works in some genre (say, Drama, Action, Romance, ...) must pair up with the group of directors who primarily make movies in the same genre. Due to space constraint, in Fig. 10.4 (a) we have shown A-D community pairings for the Romance and Comedy genres. Few famous actors and directors from each community have been listed. Such pairings may help production houses

to sign up actors and directors for different movie genres. Recently, ***Vin Diesel signed up for Avatar 2 and 3 (Action movie) which is being directed by James Cameroon and this will be the first time they will be collaborating*** *[66]. Interestingly, even though they did not work together ever, we paired them together in the groups that corresponded to the Action genre on the basis of high interaction among other similar actors and directors. Thus, potential actor-director collaborations can be explored using MLN analysis.*

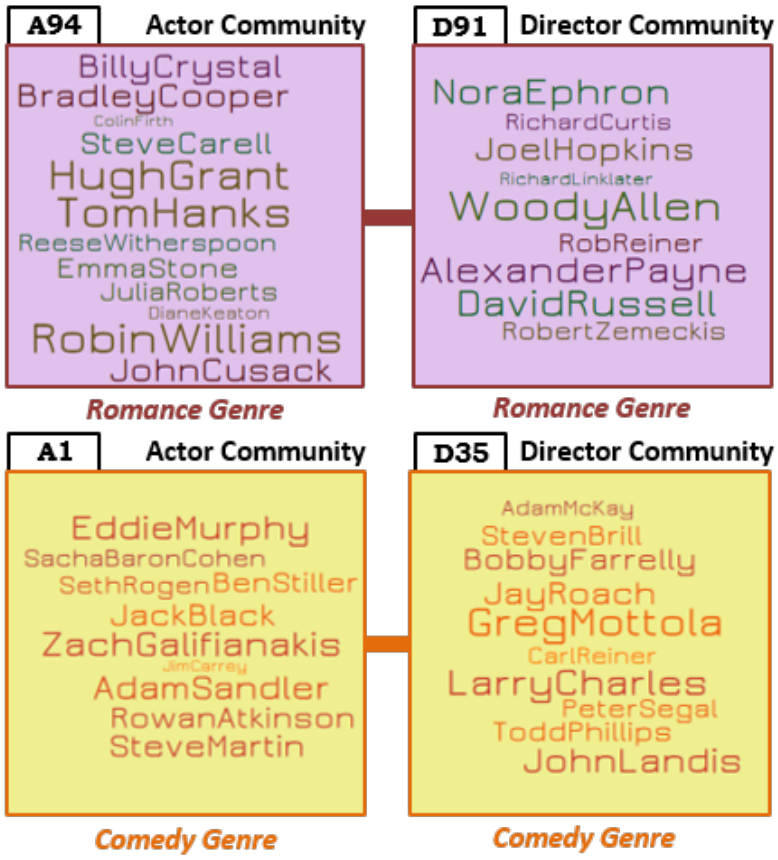


Figure 10.4: A7 results

A(8) ***For movie rating classes, which are the most popular actor and director groups that have strong interaction among them?***

Cyclic 3-community: $A\ \Theta_{A,M}\ M\ \Theta_{M,D}\ D\ \Theta_{D,A}\ A;\ \omega_e$

For A(8), we obtained the most popular actor (A) and director (D) community for each movie rating (M) community, that also have high interaction among them, through 3 iterations of MWBM. In Figure 10.5 (b) the most popular actor and director groups for [6-7) movie rating are from different genres. Even though few actor-director pairs from these two have collaborated on a few movies, it can be seen from Figure 10.5 (a) that D91 pairs (has maximum interaction) with A94. Thus, validating the absence of pairing between D91 and A144. However, in case of Figure 10.5 (c), ***both the popular groups for [7-8) rating are from Drama genre and many actor-director pairs have collaborated on many movies*** *like* ***Leonardo Di-Caprio, Kate Winslet with Sam Mendes for Revolutionary Road****,* ***Sean Penn with Gus Van Sant for Milk*** *and so on. Thus, the popular groups A175 and D106 are also paired with each other.*

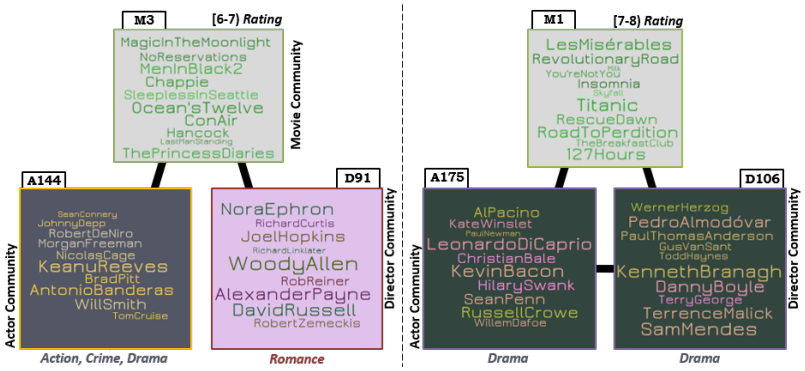


Figure 10.5: A8 results

Following two analysis A(9) and A(10) are experimented with the proposed, ***Weight Based Coupling Algorithm****.*

A(9) ***Based on similarity of genres, for each director group which are the actor groups whose majority of the most versatile members interact?***

2-community: $D\ \Theta_{A,D}\ A;\ \omega_h$

A(9) Results: 34 D-A (Director-Actor) similar genre-based community pairs were obtained, where majority of most versatile members interact. Intuitively, a group of directors that prominently makes movies in some genre (say, Drama, Action, Romance, ...) must pair up with the group(s) of actors who primarily act in similar kind of movies. Moreover, a director group may work with multiple actor groups and vice-versa. For example, in Figure 10.6, the sample result shows that the director groups, D28 and D91, with academy award winners like ***Damien Chazelle and Woody Allen, respectively, pair up with the actor group with members like Diane Keaton, Emma Stone and Hugh Grant****. Members from these groups are primarily known for movies from the* ***Romance, Comedy and Drama*** *genre.*

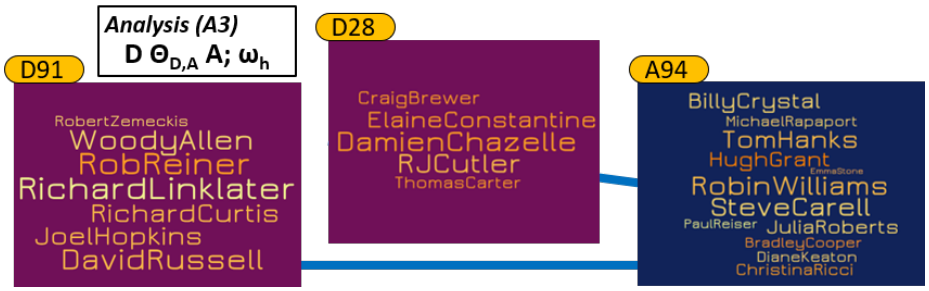
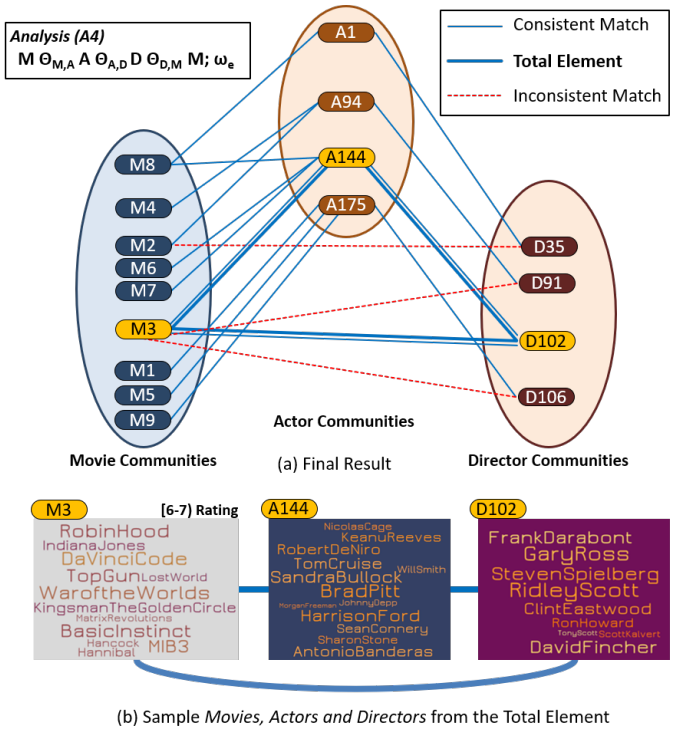


Figure 10.6: A9 Analysis results

A(10) ***For the most popular actor groups, for each movie rating class, find the director groups with which they have maximum interaction and who also make movies with similar ratings.***

Cyclic 3-community: $M\ \Theta_{M,A}\ A\ \Theta_{A,D}\ D\ \Theta_{D,M}\ M;\ \omega_e$

A(10) Results: *Here, the most popular actor groups for each movie rating class are further coupled with directors. These director groups are coupled again with movies to check whether the director groups also have similar ratings. Results of each successive pairing (there are 3) are shown in Figure 10.7 (a) using the same color notation. Coupling of movie and actor communities (first composition) results in 10 consistent matches. When the base case is extended to the director layer (second composition) using all director communities and the matched 4 actor communities, we got 4 consistent matches. The final composition to complete the cycle uses 4 director communities and 9 movie communities as left and right sets of community bipartite graph, respectively.* ***Only one consistent match is obtained to generate the total element (M3-A144-D102-M3) for the cyclic 3-community (bold blue triangle.)*** *The resulting total element is from the* ***Action, Drama genre*** *as can be seen from the sample members shown in Figure 10.7 (b). It is interesting to see 3 inconsistent matches (red broken lines) between the communities which clearly indicate that all couplings are not satisfied by these pairs. These result in 9 partial elements* ***The inconsistent matches also highlight the importance of mapping an analysis objective to a k-community specification for computation.*** *If a different order had been chosen (viz. director and actor layer as the base case), the result could have included the inconsistent matches.*



(b) Sample *Movies, Actors and Directors* from the Total Element

Figure 10.7: A10 Analysis results

10.3 ***DBLP data set***

In the first layer of MLN, Papers(P) form the nodes, where two papers which are published in the same conference are connected with an edge. Six major conferences like SIGMOD, VLDB, KDD, ICDM, DAWAK, DASFAA are selected for experiments and analysis. This brings together all the papers published in a conference and forms a clique. Second layer consists of Authors (Au) as nodes and two authors are connected with an edge if they have co-authored at least three papers. This layer brings together authors who have worked together. One can see strongly connected communities of authors who have worked together in many publications. Third layer consists of years as nodes and an edge is established between two years if it falls in the same range of 5 years. The ranges of years are 2001-2003, 2004-2006, 2007-2009, 2010-2012,

2013-2015, 2016-2018. Besides intralayer edges, there exist inter layer edges as well. Authors in the first layer are connected to papers in the second, if the author is a part of papers publication. Similarly, Paper in the second layer is connected to year in the third, if the paper is published in that year

Individual Layer Statistics*: DBLP HeMLN statistics are shown in Table 10.4. 591 Author (Au) communities are generated based on co-authorship. 6 Paper (P) communities are formed by grouping papers published in same conference. KDD (2942) and DASFAA (583) have highest and least published papers, respectively. Out of 6 ranges of years (Y) selected, the maximum and minimum papers were published in 2016-2018 (1978) and 2001-2003 (1421), respectively.*

	Author	**Paper**	**Year**
#Nodes	16,918	10,326	18
#Edges	2,483	12,044,080	18
#Communities (size >1)	591	6	6
Avg. Community Size	3.3	1721	3

Table 10.4: DBLP HeMLN Statistics

10.3.1 Analysis Objectives Results

: All DBLP data set, analysis objectives have been experimented with Louvain community detection algorithm. To map the communities between two layers, maximum weight based coupling(MWBC) algorithm has been applied. MWBC is a one to many mapping algorithm.

A(11) **For each conference, which is the *most cohesive* group of authors who publish frequently?**

2-community: P $\Theta_{P,Au}$ Au; ω_d

A(11) Analysis: On applying MWBC on the CBG created with all Paper and Author communities, we obtained 7 total elements that correspond to the *most cohesive co-authors who also publish frequently in each conference* (shown in Figure 10.8 with list of few prominent authors.) ICDM and DaWaK have **multiple author communities** that are **equally important**. Researchers **George Karypis** and Michihiro Kuramochi **are members of one of the frequently publishing co-author groups (in the last 18 years) for ICDM (4 papers)**. Significance of this result is validated from the fact that George Karypis has been a recipient of **IEEE ICDM 10-Year Highest-Impact Paper Award (2010) and IEEE ICDM Research Contributions Award (2017)**. Moreover, **multiple conferences can have same cohesive co-author groups**. For example, **co-authors Rajeev Rastogi and Minos N. Garofalakis are strongly associated with SIGMOD (7 papers) and VLDB (4 papers) in the past 18 years** Weights at the layer level are not considered in this analysis. Hence, for an author (e.g., Jiawei Han) who has authored large *number of papers*, his co-authors are distributed among different co-author communities due to lack of weight and hence does not come out. This clearly demonstrates the need for weighted communities at the layer level to increase analysis space as has been shown with meta edge weights.

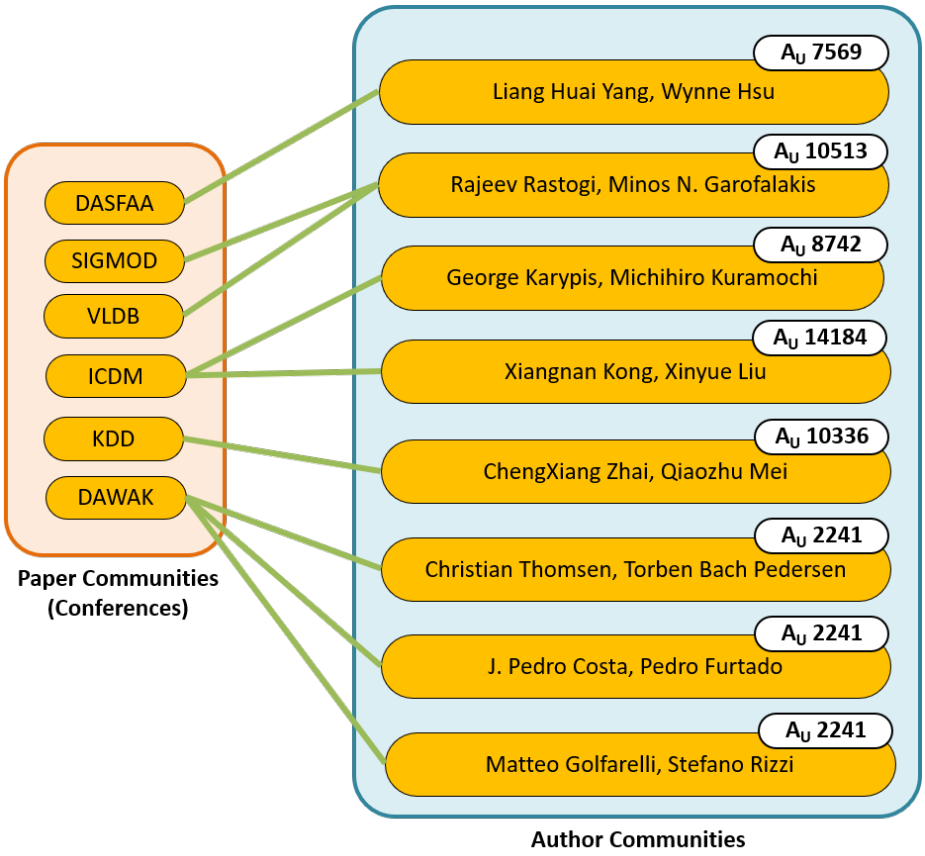


Figure 10.8: A11 Analysis results

A(12) **For the most popular collaborators from each conference, which are the 3-year period(s) when they were most active?**

3-community: P $\Theta_{P,Au}$ Au $\Theta_{Au,Y}$ Y; ω_e

A(12) Analysis: For the required *acyclic 3-community* results, the *most popular author groups* for *each conference* are obtained by MWBC (first composition). The matched 6 author communities are carried forward to find the year periods in which they were *most active* (second composition). 6 total elements are obtained (path shown by **bold blue lines** in Figure 10.9.) Few prominent names have been shown in the Figure 10.9 based on citation count (from Google Scholar profiles.) Clearly,

multiple co-author groups can be active in the same year for different conferences as seen from the results. **For SIGMOD, VLDB and ICDM the most popular researchers include Srikanth Kandula (15188 citations), Divyakant Agrawal (23727 citations) and Shuicheng Yan (52294 citations), respectively who have been active in different periods in the past 18 years**.

An interesting point to be noticed here is that none of the 6 author groups (*obtained from first composition*) had 2013-2015 and 2016-2018 as the most active periods. This is where the relevance of order comes which is derived from the analysis objectives.

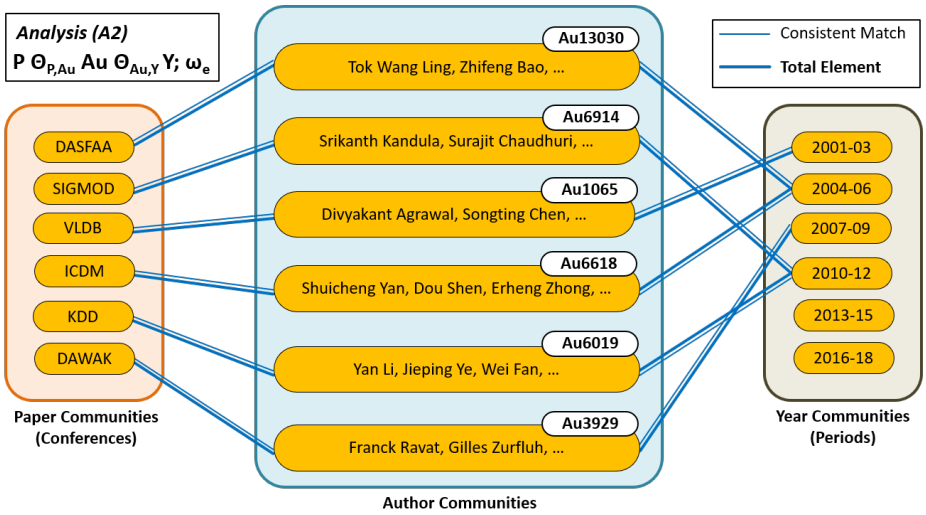


Figure 10.9: A12 Analysis results

CHAPTER 11

CONCLUSION AND FUTURE WORK

This thesis, provides a structure-preserving definition of a k-community for a MLN, efficiency of its detection and versatility. We adapted a composition function - traditional matching and also maximum weight based coupling approach with customized weight metrics for a broader analysis. Finally, we used the approach for demonstrating its analysis capability and versatility using the IMDb data set.

We have argued for using multiplexes for modeling as well as analysis. The part of the hesitation to use multiplexes for modeling comes from lack of computation algorithms as compared to other modeling alternatives. Towards that end, we have proposed and developed a community detection approach for HeMLN. We have applied it on the IMDb data set to demonstrate its applicability for flexible analysis as well as computational efficiency using the decoupling approach.

Future work includes applying this framework to Homogeneous MLNs. We are also exploring alternate definitions of a MLN community with different analysis characteristics. This decoupling approach also needs to be extended to other analysis concepts, such as centrality detection, subgraph mining, and querying of multiplexes for both types of MLNs.

REFERENCES

[1] H. A. Kathryn Dempsey, Kanimathi Duraisamy and S. Bhowmick, "A parallel graph sampling algorithm for analyzing gene correlation networks."

[2] N. M. Girvan M, "Community structure in social and biological networks."

[3] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices"."

[4] H. C. Gnce Keziban Orman, Vincent Labatut, "On accuracy of community structure discovery algorithms."

[5] L. Bohlin, D. Edler, A. Lancichinei, and M. Rosvall, "Community detection and visualization of networks with the map equation framework," 2014. [Online]. Available: http://www.mapequation.org/assets/publications/mapequationtutorial.pdf

[6] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of community hierarchies in large networks," *CoRR*, vol. abs/0803.0476, 2008. [Online]. Available: http://arxiv.org/abs/0803.0476

[7] S. Boccaletti, G. Bianconi, R. Criado, C. del Genio, J. Gmez-Gardees, M. Romance, I. Sendia-Nadal, Z. Wang, and M. Zanin, "The structure and dynamics of multilayer networks," *Physics Reports*, vol. 544, no. 1, pp. 1 – 122, 2014.

[8] A. Santra and S. Bhowmick, "Holistic analysis of multi-source, multi-feature data: Modeling and computation challenges," in *Big Data Analytics - Fifth International Conference, BDA 2017*, 2017.

[9] J. Kim and J. Lee, "Community detection in multi-layer graphs: A survey," *SIGMOD Record*, vol. 44, no. 3, pp. 37–48, 2015.

[10] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *CoRR*, vol. abs/1309.7233, 2013. [Online]. Available: http://arxiv.org/abs/1309.7233

[11] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, p. 066111, 2004.

[12] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," 2008.

[13] U. Brandes, M. Gaertler, and D. Wagner, "Experiments on graph clustering algorithms," in *In 11th Europ. Symp. Algorithms*. Springer-Verlag, 2003, pp. 568–579.

[14] M. A. Porter, J. P. Onnela, and P. J. Mucha, "Communities in networks." *Notices of the American Mathematical Society*, vol. 56, no. 9, 2009.

[15] S. Fortunato and A. Lancichinetti, "Community detection algorithms: A comparative analysis: Invited presentation, extended abstract," in *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS '09. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, pp. 27:1–27:2.

[16] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 43:1–43:35, Aug. 2013.

[17] E. A. Leicht and M. E. J. Newman, "Community structure in directed networks," *Phys. Rev. Lett.*, vol. 100, p. 118703, Mar 2008. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.100.118703

[18] T. Yang, Y. Chi, S. Zhu, Y. Gong, and R. Jin, "Directed network community detection: A popularity and productivity link model."

[19] J. W. Berry, B. Hendrickson, R. A. LaViolette, and C. A. Phillips, "Tolerating the community detection resolution limit with edge weighting," *Phys. Rev. E*, vol. 83, p. 056119, May 2011. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevE.83.056119

[20] S. Bansal, S. Bhowmick, and P. Paymal, "Fast community detection for dynamic complex networks," in *Complex Networks*. Springer, 2011, pp. 196–207.

[21] D. S. Bassett, M. A. Porter, N. F. Wymbs, S. T. Grafton, J. M. Carlson, and P. J. Mucha, "Robust detection of dynamic community structure in networks," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 23, no. 1, pp. –, 2013. [Online]. Available: http://scitation.aip.org/content/aip/journal/chaos/23/1/10.1063/1.4790830

[22] J. Yang and J. Leskovec, "Overlapping community detection at scale: A non-negative matrix factorization approach," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13. New York, NY, USA: ACM, 2013, pp. 587–596.

[23] T. Chakraborty, S. Kumar, N. Ganguly, A. Mukherjee, and S. Bhowmick, "GenPerm: An Unified Method For Finding Overlapping and Non-overlapping Communities." 2015, accepted to TKDE.

[24] C. Staudt, A. Sazonovs, and H. Meyerhenke, "Networkit: An interactive tool suite for high-performance network analysis," *CoRR*, vol. abs/1403.3005, 2014.

[25] S. Bhowmick and S. Srinivasan, "A template for parallelizing the louvain method," *Dynamics on and of Complex Networks*, vol. 2.

[26] M. Kuramochi and G. Karypis, "Finding frequent patterns in a large sparse graph*," *Data Min. Knowl. Discov.*, vol. 11, no. 3, pp. 243–271, 2005.

[27] L. B. Holder, D. J. Cook, and S. Djoko, "Substucture Discovery in the SUBDUE System," in *Knowledge Discovery and Data Mining*, 1994, pp. 169–180.

[28] S. Das and S. Chakravarthy, "Duplicate reduction in graph mining: Approaches, analysis, and evaluation," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 8, pp. 1454–1466, 2018. [Online]. Available: https://doi.org/10.1109/TKDE.2018.2795003

[29] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri, "Querying knowledge graphs by example entity tuples," in *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 10, 2015, pp. 2797–2811.

[30] S. Das, A. Goyal, and S. Chakravarthy, "Plan before you execute: A cost-based query optimizer for attributed graph databases," in *DaWaK 2016, Porto, Portugal, September 6-8, 2016*, 2016, pp. 314–328.

[31] Y. Hao, H. Cao, Y. Qi, C. Hu, S. Brahma, and J. Han, "Efficient keyword search on graphs using mapreduce," in *IEEE International Conference on Big Data, Santa Clara, CA, USA*, 2015, pp. 2871–2873.

[32] D. Shasha, J. T.-L. Wang, and R. Giugno, "Algorithmics and applications of tree and graph searching," in *PODS*, 2002, pp. 39–52.

[33] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov, "Clustering with multi-layer graphs: A spectral perspective," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 5820–5831, 2012.

[34] G.-J. Qi, C. C. Aggarwal, and T. Huang, "Community detection with edge content in social media networks," in *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 534–545.

[35] Z. Zeng, J. Wang, L. Zhou, and G. Karypis, "Coherent closed quasi-clique discovery from large dense graph databases," in *Proceedings of the 12th ACM SIGKDD*

international conference on Knowledge discovery and data mining. ACM, 2006, pp. 797–802.

[36] A. Silva, W. Meira Jr, and M. J. Zaki, "Mining attribute-structure correlated patterns in large attributed graphs," *Proceedings of the VLDB Endowment*, vol. 5, no. 5, pp. 466–477, 2012.

[37] H. Li, Z. Nie, W.-C. Lee, L. Giles, and J.-R. Wen, "Scalable community discovery on textual data with relations," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 1203–1212.

[38] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. ACM, 2012, pp. 505–516.

[39] D. Cai, Z. Shao, X. He, X. Yan, and J. Han, "Mining hidden community in heterogeneous social networks," in *Proceedings of the 3rd international workshop on Link discovery*. ACM, 2005, pp. 58–65.

[40] H. Zhang, C.-D. Wang, J.-H. Lai, and S. Y. Philip, "Modularity in complex multilayer networks with multiple aspects: a static perspective," in *Applied Informatics*, vol. 4, no. 1. Springer Berlin Heidelberg, 2017, p. 7.

[41] M. Bazzi, M. A. Porter, S. Williams, M. McDonald, D. J. Fenn, and S. D. Howison, "Community detection in temporal multilayer networks, with an application to correlation networks," *Multiscale Modeling & Simulation*, vol. 14, no. 1, pp. 1–41, 2016.

[42] A. Santra, S. Bhowmick, and S. Chakravarthy, "Efficient community re-creation in multilayer networks using boolean operations," in *International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland*, 2017, pp. 58–67. [Online]. Available: https://doi.org/10.1016/j.procs.2017.05.246

[43] ——, "Scalable holistic analysis of multi-source, data-intensive problems using multilayered networks," *arXiv preprint arXiv:1611.01546*, 2016.

[44] M. D. Domenico, V. Nicosia, A. Arenas, and V. Latora, "Layer aggregation and reducibility of multilayer interconnected networks," *CoRR*, vol. abs/1405.0425, 2014. [Online]. Available: http://arxiv.org/abs/1405.0425

[45] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2017.

[46] Y. Sun and J. Han, "Mining heterogeneous information networks: a structural analysis approach," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 20–28, 2013.

[47] C. Wang, Y. Sun, Y. Song, J. Han, Y. Song, L. Wang, and M. Zhang, "Relsim: relation similarity search in schema-rich heterogeneous information networks," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 621–629.

[48] C. Wang, Y. Song, H. Li, M. Zhang, and J. Han, "Text classification with heterogeneous information network kernels." in *AAAI*, 2016, pp. 2130–2136.

[49] J. Zhang, P. S. Yu, and Y. Lv, "Organizational chart inference," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1435–1444.

[50] C. Shi, Y. Li, S. Y. Philip, and B. Wu, "Constrained-meta-path-based ranking in heterogeneous information network," *Knowledge and Information Systems*, vol. 49, no. 2, pp. 719–747, 2016.

[51] C. Shi, Z. Zhang, P. Luo, P. S. Yu, Y. Yue, and B. Wu, "Semantic path based personalized recommendation on weighted heterogeneous information networks,"

in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 453–462.

[52] Y. Mass and Y. Sagiv, "Knowledge management for keyword search over data graphs," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014*, 2014, pp. 2051–2053. [Online]. Available: http://doi.acm.org/10.1145/2661829.2661846

[53] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum, "Automated template generation for question answering over knowledge graphs," in *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, 2017, pp. 1191–1200. [Online]. Available: http://doi.acm.org/10.1145/3038912.3052583

[54] E. Blasch, J. Llinas, D. Lambert, P. Valin, S. Das, C. Chong, M. Kokar, and E. Shahbazian, "High level information fusion developments, issues, and grand challenges: Fusion 2010 panel discussion," in *Information Fusion (FUSION), 2010 13th Conference on*. IEEE, 2010, pp. 1–8.

[55] E. P. Blasch, D. A. Lambert, P. Valin, M. M. Kokar, J. Llinas, S. Das, C. Chong, and E. Shahbazian, "High level information fusion (hlif): Survey of models, issues, and grand challenges," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 27, no. 9, pp. 4–20, 2012.

[56] A. Berenstein, M. P. Magarinos, A. Chernomoretz, and F. Aguero, "A multilayer network approach for guiding drug repositioning in neglected diseases," *PLOS*, 2016.

[57] Y. Sun, J. Han, P. Zhao, Z. Yin, H. Cheng, and T. Wu, "Rankclus: integrating clustering with ranking for heterogeneous information network analysis," in *Pro-*

ceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology. ACM, 2009, pp. 565–576.

[58] Y. Sun, Y. Yu, and J. Han, "Ranking-based clustering of heterogeneous information networks with star network schema," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009, pp. 797–806.

[59] D. Melamed, "Community structures in bipartite networks: A dual-projection approach," *PloS one*, vol. 9, no. 5, p. e97823, 2014.

[60] Z. Galil, "Efficient algorithms for finding maximum matching in graphs," ser. ACM Computing Serveys '86, 1986.

[61] M. De Domenico, A. Solé-Ribalta, S. Gómez, and A. Arenas, "Navigability of interconnected networks under random failures," *Proceedings of the National Academy of Sciences*, 2014. [Online]. Available: https://www.pnas.org/content/early/2014/05/21/1318469111

[62] J. D. Wilson, J. Palowitch, S. Bhamidi, and A. B. Nobel, "Community extraction in multilayer networks with heterogeneous community structure," *J. Mach. Learn. Res.*, vol. 18, no. 1, pp. 5458–5506, 2017. [Online]. Available: http://dl.acm.org/citation.cfm?id=3122009.3208030

[63] A. Santra, S. Bhowmick, and S. Chakravarthy, "Hubify: Efficient estimation of central entities across multiplex layer compositions," in *2017 IEEE International Conference on Data Mining Workshops, ICDM Workshops*, 2017.

[64] J. Edmonds, "Maximum matching and a polyhedron with 0, 1-vertices," *Journal of research of the National Bureau of Standards B*, vol. 69, no. 125-130, pp. 55–56, 1965.

[65] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, p. 399404, 1956.

[66] J. Stolworthy, "Avatar 2 and 3: Vin diesel joins cast of james camerons long awaited sequels," *https://www.independent.co.uk/us*, 2019.

BIOGRAPHICAL STATEMENT

Kanthi Komar was born in Shivamogga, Karnataka, India. She received her Bachelor of Engineering degree in Computer Science from Acharya Institute of Technology, Bengaluru, India in June 2016. She later went to work as a Program Analyst at Cognizant from Jan 2017 to July 2017. She started her Master studies in Computer Science at the University of Texas at Arlington in Fall 2017. She interned as a Data Engineer at Insight Data Science from Jan 2018 to April 2018. She received her Master of Science in Computer Science from The University of Texas at Arlington, in August 2019. Her research interests include Graph Mining, Big Data Engineering and Analysis and Machine Learning.