UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Dottorato di Ricerca in

Information and Communication Technologies

CICLO

XXXIII

Mining and Learning Problems in Complex Graph Data

Settore Scientifico Disciplinare ING-INF/05

Coordinatore: Prof. Felice Crupi

Supervisore/Tutor: Prof. Sergio Greco

Supervisore/Tutor: Prof. Andrea Tagarelli

Dottorando: Dott. Domenico Mandaglio

Abstract (Italian)

I grafi sono modelli matematici che rappresentano oggetti, chiamati nodi o vertici, coinvolti in relazioni a coppia, detti archi. Tali modelli vengono impiegati per descrivere sistemi interconnessi tra cui reti tecnologiche (es. il World Wide Web), reti sociali e biologiche. A partire dal modello originario dei grafi, diverse estensioni del modello sono state proposte in letteratura: grafi pesati, multidimensionali, temporali e probabilistici permettono di esprimere, rispettivamente, l'intensità associata a ogni arco, rappresentare diverse tipologie di relazioni tra vertici, includere informazioni su quando le interazioni tra nodi avvengono, e assegnare a ogni possibile peso sugli archi la probabilità di osservare quello specifico peso. Lo scopo di questa tesi è definire modelli e metodi per problemi di mining e learning di relazioni forti e nascoste tra nodi su grafi complessi. In particolare, il focus di questo progetto di ricerca è la scoperta di relazioni a coppia come associazioni di gruppo e relazioni di trust. Il primo obiettivo consiste nel partizionare l'insieme dei nodi di un grafo in gruppi (detti cluster o community) tali che i nodi appartenenti allo stesso gruppo siano collegati più fortemente tra di loro rispetto che con il resto della rete. Questo obiettivo è anche noto in letteratura come graph clustering o community detection. Le community (o cluster) sono gruppi di entità che probabilmente condividono delle proprietà e/o hanno un ruolo simile all'interno del sistema a cui appartengono. Le relazioni di trust vengono tipicamente modellate attraverso un grafo pesato, detto rete di trust, che si riferisce a un grafo di individui collegati da relazioni di coppia asimmetriche corrispondenti a espressioni soggettive di fiducia, dove il peso associato a ogni arco viene interpretato come il grado di fiducia che un utente ha nei confronti di un altro individuo. Ogni modello di rappresentazione a grafo, indipendentemente dalla sua natura, permette di descrivere in diversi modi l'intrinseca natura multiforme dei sistemi reali che deve essere tenuta in considerazione quando si intende identificare relazioni tra i nodi come quelle di gruppo o di trust. Questo implica la necessità di un processo di aggregazione di informazioni che permette di considerare simultaneamente i diversi aspetti del sistema rappresentato. Tuttavia, l'aggregazione dei diversi aspetti di un sistema pone alcune problematiche aggiuntive al task considerato poiché le diverse dimensioni dei dati potrebbero essere inconsistenti tra di loro o l'informazione relativa a un qualche aspetto potrebbe rappresentare rumore per il raggiungimento dell'obiettivo prefissato. L'abbondanza e diversità di dati rappresentabili attraverso grafi che può essere estratta da sistemi online (es. il Web) o offline (es. interazioni sociali) favorisce la necessità di nuovi modelli e metodi che siano capaci di tenere in considerazione efficacemente l'eterogeneità nella tipologia di informazioni nella scoperta di pattern nel comportamento di entità appartenenti a un sistema complesso. Più specificatamente, quattro sono i temi di ricerca che possono essere indentificati in questa tesi. Primo, è stato studiato il problema di consensus community detection su reti multidimensionali: dato un insieme di partizionamenti dei nodi di una rete, ciascuno calcolato considerando separatamente una dimensione del grafo multidimensionale, trovare un nuovo partizionamento dei nodi (detto consensus clustering) che sia rappresentativo e, allo stesso tempo, filtri l'eventuale rumore dei vari partizionamenti in input. Come seconda linea di ricerca, è stato trattato il problema di consensus community detection dinamico su grafi temporali che consiste nel calcolare, per ogni stato di evoluzione di una rete, un consensus clustering che sia rappresentativo degli stati precedentemente osservati sulla rete e, quindi, rispecchi la sequenza delle strutture a community nei vari istanti di tempo. Chiaramente, la natura temporale di questo secondo problema pone alcune sfide aggiuntive nella sua risoluzione poiché i vari partizionamenti sono disponibili e devono essere processati in modo incrementale;

inoltre, vi è il requisito che bisogna opportunamente pesare le informazioni sugli stati nella rete in modo da dare maggior rilievo agli stati più recenti piuttosto che quelli più remoti. Inoltre, la dimensione temporale delle interazioni tra utenti di una rete sociale può aiutare a inferire una rete di trust. Quest'ultimo obiettivo corrisponde alla terza linea di ricerca di questa tesi che ha come obiettivo la risoluzione del seguente problema: data una sequenza di grafi pesati corrispondenti agli stati di una rete in diversi istanti di tempo, derivare un grafo pesato e orientato, i cui nodi corrispondono alle entità del grafo temporale e gli archi rappresentano le relazioni di trust con associato grado di fiducia. Come quarta linea di ricerca è stato studiato un nuovo problema di clustering su grafi probabilistici in cui le interazioni tra i nodi sono caratterizzate da distribuzioni di probabilità e condizionate da fattori esterni ai nodi ma caratteristici dell'ambiente in cui interagiscono. Questo contesto include ogni scenario in cui una serie di azioni possono alterare le interazioni tra entità tra cui, ad esempio, i sistemi di raccomandazione su piattaforme di social media e task di team formation. In particolare, è stato considerato il caso in cui i fattori condizionanti le interazioni possono essere modellati attraverso un clustering dei nodi del grafo e l'obiettivo è trovare il clustering che massimizza l'interazione totale nel grafo. Per ciascuna linea di ricerca sono stati proposti degli algoritmi che sono stati confrontati, su dati reali e/o generati artificialmente, con lo stato dell'arte dei rispettivi problemi al fine di valutarne sia l'efficacia che l'efficienza.

Acknowledgements

First, I want to acknowledge and thank Prof. Sergio Greco for the opportunity to carry on the research of this thesis without constraining it. My sincere thanks also go to Prof. Andrea Tagarelli for his support and guidance over the past three years. None of the research I've done would have been possible without Andrea's guidance and mentoring.

I also acknowledge Alessia Amelio and Francesco Gullo as invaluable mentors and contributors that, together with Andrea, have given shape to my research activities. They helped me become a better researcher and better writer, and has always been an absolute pleasure to collaborate with each of them.

I am grateful to my wonderful roommates Nicola, Antonio, Rocco, Francesco and Ludovica for making my time in office so enjoyable. I also thank all the people working in the 44Z cube for good conversations in front of a cup of coffee (thanks Eugenio for making this possible) and all the other activities that have come along with working in the same office.

A big thank you goes to Prof. Sergio Flesca and Prof. Ester Zumpano for providing guidance over my teaching activities and making these experiences nice.

I would also like to thank my best friends and relatives, for their support and presence during this journey.

Lastly, I would like to thank my parents Giuseppe and Concetta, my brother Angelo and my sister Debora for their love and encouragement throughout the years.

Contents

List of Figures

List of Tables

To my family

Chapter 1

Introduction

A graph is a mathematical structure representing a set of objects, called nodes or vertices, which share pairwise relationships, called edges. The term 'network' is nearly synonymous, and is specifically used to refer to a complex system of interacting agents that, at its core, can be modeled and studied using a graph. Complex systems studied by network scientists include transportation networks (e.g., road networks, airline networks), technological networks (e.g., the World Wide Web, the Internet, cell phone networks), social networks (e.g., friendship networks), biological networks (e.g., neural systems, protein interaction networks) and collaboration networks (e.g. Wikipedia). Given the fact that complex networks are pervasive, they have long attracted attention from researchers in computer science, social science, biology, and other disciplines.

Clearly, to be useful, a model has to represent the features of real world phenomena in a simple way, but without loosing too many details in the process. Therefore, upon the original simple graph model, many extensions have been proposed: *weighted*, *multilayer* [26], *time-evolving* [23] and *uncertain* [71] relations are now fundamental building bricks of any study aiming to unveil novel insights about the complex interacting phenomena in the real world. Each of these complex graph models extends the information associated to relations in a graph: weighted graphs allow to express a strength associated to each relationship, the multilayer graph model enables to represent multiple modes of interaction or different types of relationships among entities of the same type (e.g. people), temporal graphs include information on when interactions between nodes happen, uncertain graphs assign to every possible weight on edges a probability of observing the corresponding weight on the edge.

On these complex graph models, countless different problems have been tackled and numerous algorithms have been proposed to solve them. Among these, some of the existing fundamental problems aim to identify *strong* and *hidden* relationships among nodes in a graph. These target relationships are hidden since they cannot trivially be induced from the graph data (e.g. just by computing predefined quantities from edges information) and they are also strong because they reflect different patterns of association between nodes.

As an example, one of the most fundamental problems in network science and graph theory is to identify *group associations* which, in the most widely accepted setting, consists of finding sets of nodes that are more closely connected to each other than to the rest of the graph [96]. This task is also known in the literature as the graph clustering problem or community detection[1]. Real networks are not random, rather nodes naturally cluster into groups and links are more likely to connect nodes within the same group. This phenomenon tells us that the organization of such complex networks is modular. Network scientists call this organization as the *community structure* of networks. Communities (or clusters) are groups of entities that

[1]Unless otherwise specified, in this thesis the term graph clustering (resp. cluster) will be used as synonymous with community detection (resp. community).

probably share common properties and/or play similar roles within the interacting phenomenon that is being represented: they may correspond to groups of pages of the World Wide Web dealing with related topics, to related genes in a biological network, to groups of related individuals in social networks and so on.

Another example of strong and hidden relationships that involve pairs of nodes in a graph are *trust relationships*. These are usually modeled as a directed and weighted graph, called *trust network*, which commonly refers to a graph of entities (i.e., individuals) that are linked through asymmetric relationships that correspond to subjective trust statements where the trust score associated to each edge can be regarded as a personalized opinion of one user (trustor) with respect to another user (trustee). Given a trust network, *trust inference* is the task of predicting a new relation between two nodes. Trust inference is an essential task in many data analysis and machine learning applications, from social influence propagation and opinion spreading to recommender systems and privacy preserving, whose impact extends also to peer-to-peer networks and mobile ad-hoc networks. Unfortunately, conventional approaches to trust inference assume trust networks are available, while in practice, with few exceptions, they must be derived from social network features.

Most of the existing problems on graph data, as well as the adopted approaches to solve them, fall under the broad field of data mining and machine learning. Although the two paradigms share some common traits, data mining is designed to extract the rules from large quantities of gathered data while machine learning trains a system (by learning some model parameters) to perform complex tasks and uses harvested data and experience to become smarter in the accomplishment of the given tasks. Since machine learning is conceived to learn from past experience/data to perform better with future data, this paradigm is more suitable for problems on dynamic graphs.

Each complex graph model, regardless of its static/dynamic nature, allows to describe, in different way, the intrinsic multifaceted nature of real world systems. The fact that reality is multifaceted should also be taken into account while discovering nodes relationships like group associations or trust links. This implies some form of aggregation process which allows to consider each aspect of the modeled reality. As an example, the community detection problem in multilayer networks aims to find a good community structure for most layers, i.e. it should seek to identify communities whose nodes are internally connected by many edges possibly of different types, and are externally connected by few edges of different types. However, simultaneously considering different aspects of a complex graph model poses additional challenges to the task since different dimensions of the data may be inconsistent to each other: in a multilayer network representing friendship and relative relationships among a set of people, edges related to friendship may be very different from links denoting pair of users which are relatives; also, in a temporal network whose edges describe interactions among a set of users on an online social platform at different time, links may vary dramatically from a timestep to the next one. Moreover, information coming from some dimensions could represent noise for the particular task at hand, thus it would be helpful to have a mechanism which can filter noise in the aggregation process. The abundance and diversity of graph data which can be extracted nowadays from online (e.g. the Web) or offline (e.g. social interactions) systems favors an increasing demand for new models and methods that are capable of effectively taking into account the heterogeneity in the type of information, discovering entities' behaviour patterns in a complex graph system.

1.1 Contributions

In this thesis we address research topics that are centered around the problem of mining and learning strong and hidden relations in complex graph data. Specifically, four main research topics can be distinguished:

Consensus community detection in multilayer networks. The consensus clustering consists in the following problem: given a set of clusterings as different groupings of the input data, a consensus (or aggregation) criterion function is optimized to induce a single, meaningful solution that is representative of the input clusterings. This paradigm has emerged as an effective tool for community detection in multilayer networks, which allows for producing consensus solutions that are designed to be more robust to the algorithmic selection and configuration bias. However, one limitation is related to the dependency on a co-association-based consensus clustering scheme, i.e., the consensus clusters are derived from a co-association matrix built to store the fraction of clusterings in which any two nodes are assigned to the same cluster. Low values in this matrix would reflect unlikely consensus memberships, i.e., noise, and hence should be removed; to this purpose, the matrix is subjected to a filtering step based on a user-specified parameter of minimum co-association, θ. Unfortunately, setting an appropriate θ for a given input network is a challenging task. The goal of this research line is to overcome this limitation with a new framework of ensemble-based multilayer community detection, which features parameter-free identification of consensus communities based on generative models of graph pruning that are able to filter out noisy co-associations. The two main findings of this research line, drawn from experimental results, are: (i) some of the model-filters are effective in simplifying an input multilayer network to support improved community detection, and (ii) the proposed framework outperforms state-of-the-art multilayer community detection methods according to some quality criteria. *The research line on consensus community detection in multilayer networks is the subject of Chapter 3.*

Dynamic consensus community detection in temporal networks. Community detection in temporal networks is a very active field of research, which can be leveraged for several strategic decisions, including enhanced group-recommendation, user behavior prediction, and evolution of user interaction patterns in relation to real-world events. This research line introduces the novel problem of dynamic consensus community detection, i.e., to compute a single community structure that is designed to encompass the whole information available in the sequence of observed temporal snapshots of a network in order to be representative of the knowledge available from community structures at the different time steps. Unlike existing approaches, the dynamic consensus community structure should be able to embed long-term changes in the community formation as well as to capture short-term effects and newly observed community structures. Also, differently from the previous research line, computing meaningful co-associations for the nodes in a temporal network and properly maintaining and updating the consensus community structure over time should be done by avoiding the recomputation of the consensus from scratch each time a new status of the network is available. In order to accomplish this task, a hybrid reinforcement learning/clustering framework is defined based on the combinatorial multi-armed bandit paradigm. The introduced framework is also instantiated in an efficient method which is conceived to cope with temporal networks having different evolution rates. *The research line on dynamic consensus community detection in temporal networks is the subject of Chapter 4.*

Trust network inference. The conventional approach to trust inference is to compute the relation between any two non-adjacent nodes in a trust network by considering the different paths from one node to the other, as well as strategies for trust propagation and for aggregating the propagated trust values through different paths. Unfortunately, all existing trust-inference approaches rely on the assumption that a trust network has been already formed, while in reality trust networks are not naturally available. Rather, trust relations must first be determined by aggregating the available information in a social environment, e.g., the history of users' activities and their interactions. Computing trust relations is however a particularly difficult task, because of a number of challenges that already arise at data source level. In fact, the amount of information representing the observed interactions and activities of users in a social network, could be limited in size and quality. More specifically, a social network may contain a significant amount of redundant or irrelevant relations as well as noise in the information that express the strength of interaction between any two users. In this research line, we face the above challenges by addressing a new problem we named Trust Network Inference (TNI). Given a sequence of timestamped interaction networks as input, the goal of TNI is to infer from this sequence a directed weighted network, whose nodes are the users in the temporal networks and links denote trust relationships with associated trust scores. We propose to solve the TNI problem based on a generalized preference learning paradigm since it provides key advantages in addressing all the aforementioned issues, i.e., limitedness, redundancy and noise of the information about the users' interactions from which a trust network is to be inferred. One further key feature of proposed approach is domain-independency, as it does not rely on platform-specific types of user interactions. Nonetheless, our approach is designed to exploit both topological information and, when available, content information relating to the user interaction dynamics. *The research line on trust network inference is the subject of Chapter 5.*

Optimizing interactions in probabilistic graphs under clustering constraints. Modeling and mining behavioral patterns of users of online as well as offline systems is central to enhance the user engagement and experience in the systems. In this regard, uncertain graph models are seen as a powerful tool to capture the inherent uncertainty in user behaviors. A common way of modeling uncertainty in a graph is to associate each pair of users with a probability value that expresses the likelihood of observing and quantifying an interaction between the two users. In this regard, one important aspect is that the modeling of user interactions should also account for exogenous conditions or events that occur within the social environment where the users belong to, which indeed can significantly affect the users' interaction behaviors. For example, delivering a post on a user's page (e.g., Facebook wall) that contains a message of friend recommendation will likely favor or not a meeting between two users, and so their interactions. Intuitively, it is of high interest to identify proper settings of a network system and relating conditions that can maximize the overall user interactions within the system. In this research line, we extend the uncertain graph modeling framework to capture the dependency of interactions on conditioning factors, in a network system. In particular, we focus on the case when the interaction behaviors depend on a clustering of the set of users in a graph, so that the probability of interaction between any two users varies depending on whether they belong to the same cluster or not. Modeling such interaction conditioning factors in terms of cluster memberships of users arises in several relevant application scenarios such as recommendations on social-media platforms and team formation tasks. The goal

of this research line is to introduce the problem of optimizing the overall interaction among a set of entities in a probabilistic graph, subject to the cluster memberships of the entities, i.e. partition a set of entities such as to maximize the overall vertex interactions or, equivalently, minimize the loss of interactions in the graph. After theoretically characterizing the problem, approximations algorithms and heuristics are introduced and their efficiency and effectiveness are evaluated. *The research line on optimizing interactions in probabilistic graphs under clustering constraints is the subject of Chapter 6.*

Chapter 2

Background

Summary. This chapter enstablishes preliminary concepts that will help the understanding of the techniques and models adopted in subsequent chapters. First, it introduces some background on graph models. Then, it reports on the major details about graph clustering problems on different graph data models: starting from community detection methods on simple graphs, the focus is then moved to more complex settings such as (consensus) community detection on multilayer networks and clustering on uncertain graphs. Finally, after a brief overview of bandit and preference-based learning problems, the chapter describes how these learning paradigms can be useful to understand graph dynamics.

2.1 Basic definitions

A graph G is a pair of sets $G = (V, E)$, where V is the set of vertices (or nodes) and E is the set of edges (or links). In an *undirected* graph each edge is an unordered pair (i, j) while in a *directed* graph the set of pairs is ordered. A graph is *weighted* if a weighting function $w : E \to \mathbb{R}$ assigns a weight $w(u, v)$ to each edge $(i, j) \in E$ which denotes the strength od the interaction between the two linked nodes. If the weigh function is not provided the graph is said *unweighted*. The links of a graph G can be described by its $|V| \times |V|$ adjacency matrix, denoted with A, such that $A_{ij} = 1$ if $(i, j) \in E$, 0 otherwise. We always assume loopless graphs, i.e. $A_{ii} = 0$ for each $i \in V$.

For an undirected graph G, the *degree* k_i of a node $i \in V$ is the number of edges incident with the node, and it is defined in terms of the adjacency matrix A as $k_i = \sum_{j \in V} A_{ij}$. If the graph is directed, the degree of the node has two components: the number of outgoing links (or out-degree) $k_i^{out} = \sum_{j \in V} A_{ij}$ and the number of ingoing edges (or in-degree) $k_i^{in} = \sum_{j \in V} A_{ji}$. The total degree is defined as $k_i = k_i^{out} + k_i^{in}$.

Given an undirected graph G, for any node $i \in V$ its *strength* $s_i = \sum_{j \in V} A_{ij} w(i, j)$ is the sum of of weights of links connected to the node. In directed graphs, similarly to degree, each node is charachterized by the in-strength as the sum of inward link weights and the out-strength as the sum of outward link weights.

A list of the node degrees of a graph is called the degree sequence. The most basic topological characterization of a graph G can be obtained in terms of the degree distribution $P(k)$, defined as the probability that a node chosen uniformly at random has degree k or, equivalently, as the fraction of nodes in the graph having degree k. Directed networks can be characterized by two distributions, $P(k^{out})$ and $P(k^{in})$.

A graph $G' = (V', E')$ is a *subgraph* of another graph $G = (V, E)$ iff $V' \subseteq V$, $E' \subseteq E$ and for each $(i, j) \in E'$ it holds that $i, j \in V'$. Given a graph $G = (V, E)$, for any subset of vertices $V' \subseteq V$ the induced subgraph $G[V']$ is the graph whose vertex set is V' and whose edge set consists of all of the edges in E that have both endpoints in V'. The same definition works for both undirected and directed graphs. Also, the induced subgraph $G[V']$ may also be called the subgraph induced in G by V'.

Uncertain (or *probabilistic*) *graphs* introduce uncertainty with respect to the existence of nodes and edges. Although existential probabilities can be assigned to the vertices of the graph as well, in this thesis we focus on edge probabilities only. The existence of an edge depends on several factors depending on the particular application under consideration. For example, in a social network where the edge corresponds to a message exchange between two users, the message will be sent with some probability (i.e., it is not sure that user u will send a message to user v).

Let $\mathcal{G} = (V, E, p)$ be an uncertain graph, where V is the set of nodes and $p : E \rightarrow (0, 1]$ is a function that assigns probabilities to the edges of the graph and p_{uv} represents the probability that edge (u, v) exists. A widely used approach to analyze uncertain graphs is the one of *possible worlds*, which considers an uncertain graph $\mathcal{G}$ as a generative model and each possible world constitutes a deterministic realization of $\mathcal{G}$. According to this view, an uncertain graph $\mathcal{G}$ is interpreted as a set $\{G = (V, E_G)\}_{E_G \subseteq E}$ of possible deterministic graphs. Let $G \sqsubseteq \mathcal{G}$ indicate that G is a possible world of $\mathcal{G}$. Given a possible world $G = (V, E_G)$, the probability of observing it, assuming independence between edges as usual in the literature on uncertain graphs [83, 75, 8, 69, 101, 17, 70, 71] , is given by the following:

$$Pr(G) = \prod_{(u,v)\in E} p_{uv} \prod_{(u,v)\in E\setminus E_G} (1 - p_{uv}) \tag{2.1}$$

A *multilayer network* graph $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$ is a tuple, where $\mathcal{V}$ is the set of entities and $\mathcal{L} = \{L_1, \ldots, L_\ell\}$ denotes the set of layers. Each layer corresponds to a given type of entity relation, or edge-label. For each pair of entity in $\mathcal{V}$ and layer in $\mathcal{L}$, let $V_{\mathcal{L}} \subseteq \mathcal{V} \times \mathcal{L}$ be the set of entity-layer pairs representing that an entity is located in a layer. The set $E_{\mathcal{L}} \subseteq V_{\mathcal{L}} \times V_{\mathcal{L}}$ contains the undirected links between such entity-layer pairs. For every layer $L_i \in \mathcal{L}$, V_i and E_i denote the set of nodes and edges, respectively. Also, the inter-layer edges connect nodes representing the same entity across different layers (monoplex assumption). A *multiplex network* is a special case of a multilayer network where $E_{\mathcal{L}}$ is restricted to intra-layer edges, i.e. no inter-layer edges connecting entities in different layers are allowed. The focus of this thesis is on multiplex network but the term multilayer is sometimes used as synonymous.

Given a set of entities $\mathcal{V}$, a *temporal network* $\mathcal{G} = (G_1, G_2, \ldots, G_t, \ldots)$ is a series of graphs over discrete time steps, where $G_t = \langle V_t, E_t \rangle$ is the graph at time t, with set of nodes V_t and set of undirected edges E_t. Each node in V_t corresponds to a specific instance from the set $\mathcal{V} \subseteq \mathcal{V}$ of entities that occur at time t.

2.2 Graph clustering

Definition 1 (Clustering solution) *Given a graph $G = (V, E)$, a clustering solution or simply a clustering $\mathcal{C} = \{C_1, \cdots, C_k\}$ is a partition of the set of nodes V into k groups or clusters, i.e. $C_i \cap C_j = \emptyset$ for every $i \neq j$ and $\cup_{i=1}^{k} C_i = V$.*

Although detecting overlapping clusters is also an important and well-studied task (cf. [124] for a survey), in this thesis we focus on graph clustering problems based on non-overlapping clusters.

In most cases, graph clustering is cast as the task of optimizing an objective function $f : \mathscr{C} \rightarrow \mathbb{R}$, where $\mathscr{C}$ denotes the set of valid clusterings. For every clustering $\mathcal{C} \in \mathscr{C}$, the function f outputs a real-valued objective score $f(\mathcal{C}) \in \mathbb{R}$ which quantifies how much the clustering $\mathcal{C}$ is a good fit for a particular graph G. The choice of the quality function f depends on the particular clustering problem and method which

typically provide a specific objective function to minimize/maximize in order to define good and bad clusterings.

Some settings may require to deal with clusterings with a specified number of clusters. As an example, consider parallel computing [96] where where nodes in a graph represent computational tasks, and edges indicate some form of data dependence. In this case, if there are k processors for accomplishing the work, it is beneficial to split up the tasks into exactly k clusters in a way that minimizes communication between processors. In order for the work to be balanced among the k processors, it is important to separate nodes into blocks of roughly the same size. This task, which is usually referred in the literature as graph partitioning, asks to find k balanced clusters which minimize the number of edges between them. Given the emphasis on minimizing inter-cluster connectivity, graph partitioning often places a higher priority on external sparsity than on internal density.

The design goals of graph partitioning occasionally conflict with the motivation behind certain applications of graph clustering. In many real-world networks (e.g., biological or social networks), it is unhelpful to assume there are a fixed number of clusters. Furthermore, it may very well be the case that natural communities in the graph vary significantly in size.

In this thesis we consider clustering problems where the number of clusters is not given as an input parameter; instead, the clustering size is determined by the specific objective function. Also, differently from graph partitioning, we focus on clustering problems whose objective f drives the discovery of clusters with specific requirements on both inter-cluster and intra-cluster connectivity. Next, we discuss some clustering problems and methods which shares some similarities with the graph clustering problems presented in this thesis.

2.2.1 Community detection

Community detection has been studied extensively across different disciplines and, although several definitions of community has been provided in the literature, there is significant general agreement about the fundamental characteristics of a graph community. A community is often thought of as a set of nodes that has more connections between its members than to the remainder of the network [80, 96]. This definition highlights two basic requirements for community detection: communities should have a high internal density and sparse external connections.

Following the standards in the literature, hereinafter we will use the terms community detection and graph clustering synonymously in this thesis. Similarly, the words cluster and community will be typically used interchangeably to refer to densely connected nodes that are only loosely connected to the rest of the graph.

Widespread agreement about the high-level definition of community detection quickly turns into a plethora of different ways to formalize and study the problem in theory and practice. The top-level distinction is between global or local methods, respectively discovering all communities in the input network or generating a single community around one or more seed nodes. Although the latter setting is of interest, the focus of this thesis is on global methods. Existing approaches to (global) community detection includes traditional methods based on data clustering like k-means [84], divisive algorithms mainly based on hierarchical clustering [59], spectral algorithms [102] and dynamic algorithms [104]. Arguably the most widely applied objective for community detection is the modularity score [96]. In the following we revise some modularity-based community detection methods, upon which the following chapters are based.

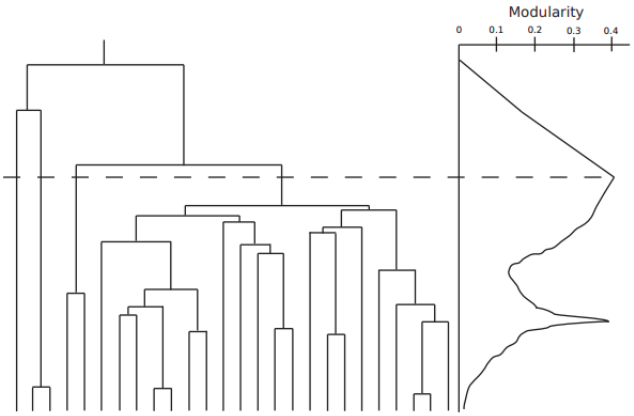


FIGURE 2.1: A dendrogram showing a possible cluster hierarchy built upon a graph with 26 nodes, with a plot of resulting modularity values given the clustering [21].

Modularity is based on the idea that a random graph is not expected to have a community structure. A clustering is said to have high modularity if clusters have higher internal edge density than would be expected at random. Randomness in this context is defined by a pre-specified null model defining the probability that an edge exists between each pair of nodes. Given a graph $G = (V, E)$, the modularity of a clustering $\mathcal{C}$ is defines as follows:

$$Q(\mathcal{C}) = \frac{1}{d(V)} \sum_{C \in \mathcal{C}} \left(d_{int}(C) - \frac{d(C)^2}{d(V)} \right)$$

where $d(V) = \sum_{v \in V} d(v)$ denotes the total degree of nodes over the entire graph, $d(C)$ is the sum of degrees of nodes within community C and $d_{int}(C)$ denotes the internal degree of C, i.e., the portion of $d(C)$ which corresponds to the number of edges that link nodes in C to other nodes in C (i.e., twice the number of links internal to C). The value of Q varies from -0.5 to 1.0. In particular, it reaches the minimum of -0.5 when all edges link nodes in different communities, and the maximum of 1.0 when all edges link nodes in the same community.

Modularity has been applied extensively and variations of the objective have been introduced for weighted graphs, bipartite graphs, multi-slice networks, and a number of other cases. Despite its widespread use, however, it is known to exhibit several limitations. First of all, even random graphs may exhibit high maximum modularity scores, making it hard to discern when a high modularity score is indicative of meaningful structure [50]. Modularity is also known to suffer from an inherent resolution limit [36], meaning that it may fail to detect communities that are smaller than a certain size, which depends on the size of the network. Finally, modularity is very challenging to optimize. Not only is the objective NP-hard, [27] showed that it is NP-hard to even approximate to within any constant factor. Thus, most of existing works are heuristic methods and they do not come with provable approximation guarantees. Next, we review some of these methods upon which the following chapters are based on.

Newman algorithm

The first algorithm devised to maximize modularity was a greedy method proposed by Newman [95]. It is an agglomerative hierarchical clustering method, where groups of vertices are successively joined to form larger communities. More specifically, given a graph with n nodes, the algorithm starts with n singleton communities. Next, the merging of each pair of communities is evaluated and the pair of communities which leads to the maximum gain of modularity or the minimum loss of modularity is selected for merging. This joining step is iteratively applied until a single community containing every node is formed. The different phases of the algorithm can be represented as a dendrogram which shows the sequence of merges, as in Fig. 2.1. Cutting the resulting dendrogram at different levels results into different community structures thus the clustering with the highest modularity can be selected from the dendrogram.

Louvain algorithm

Given a graph with n nodes and starting with a clustering with n singleton clusters, the algorithm iteratively performs two steps:

1. *Modularity maximization.* For each node i, the algorithm considers the neighbors j of i and it evaluates the gain of modularity that would take place by removing i from its community and by placing it in the community of j. Among these possible local modifications of the current clustering solution, the algorithm applies the one which leads to the maximum gain in modularity. If no positive gain is possible, node i stays in its original community. This process is applied repeatedly and sequentially for all nodes until no further improvement can be achieved thus a local maxima of the modularity is reached.

2. *Community aggregation.* Starting from the communities obtained at the end of the modularity maximization phase, a new weighted graph is built: nodes corresponds to the communities found during the first phase and the sum of the weight of the links between nodes in the corresponding two communities are the weights of the links in the new graph. Links between nodes of the same community lead to self-loops for this community in the new network. Once this second phase is completed, it is then possible to reapply the first phase of the algorithm to the resulting weighted network and to iterate.

The two phases are iterated (c.f. Figure 2.2 for an example) until there are no more changes and a maximum of modularity is attained. This process leads to a hierarchy of communities whose height is determined by the number of passes of the two steps.

Nerstrand algorithm

Nerstrand [78] is a modularity-based algorithm for community detection on undirected and weighted graphs. This method adopts the multilevel paradigm and it is composed of three phases: *coarsening*, *initial clustering*, and *uncoarsening*.

In the coarsening phase, starting from the input graph G, a series of increasingly coarser graphs $G_1, \cdots, G_s$ is generated. The coarser graph at the i-th step of this phase is obtained by grouping, through a proper aggregation scheme, nodes of the graph obtained at the $(i-1)$-th step. *First Choice Grouping* (FCG) is the default aggregation scheme: for each node v in the coarser graph, FCC groups v with the neighbor node which results in the highest modularity gain in the corresponding clustering solution.

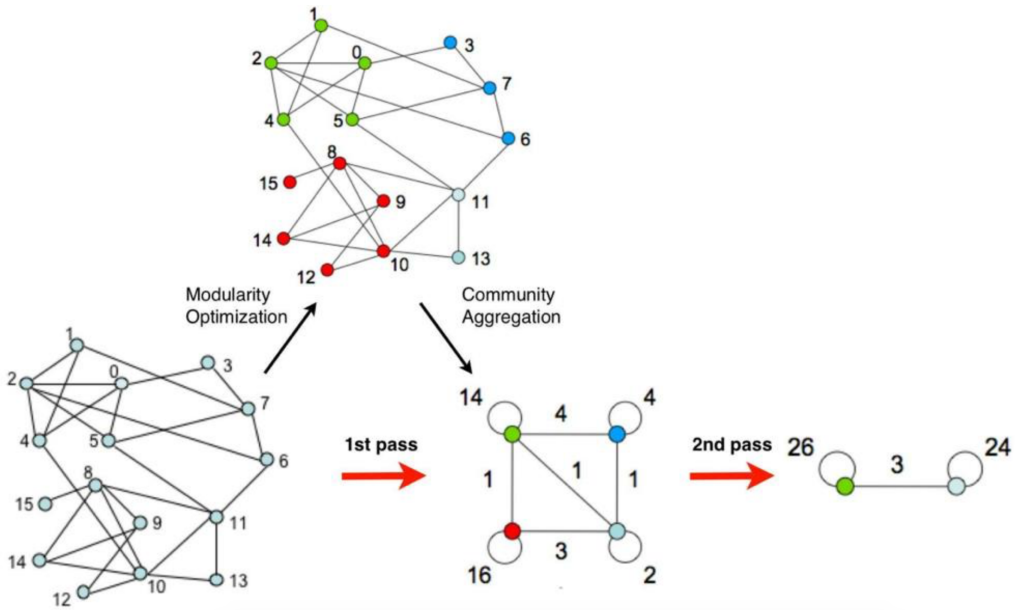


FIGURE 2.2: Visualization of the steps of Louvain algorithm [7].

Next, the initial clustering phase consists of applying a direct clustering algorithm (i.e., a non-multilevel clustering algorithm) on the coarser graph G_s, obtained in the latest coarsening phase. Finally, in the uncoarsening phase, the initial solution is used to derive solutions for the successive finer (larger) graphs. More specifically, the clustering of the coarsest graph G_s is used as an estimate for a good clustering of the finer graph G_{s-1}. This solution is then improved by finding a local maxima of modularity on G_{s-1}. This is repeated until the clustering is applied to, and improved for the initial graph G.

Extension to multilayer networks

Existing methods for community detection in multilayer networks differ in the way in which they handle the presence of multiple layers [85]: reducing them to a single layer (*flattening* methods), processing each layer independently to then aggregate the results of single-layer community detection (*aggregation* methods), or considering all the layers at the same time (*multilayer* methods).

The first approach consists in simplifying the multiplex network into a graph by merging its layers, using a so-called flattening algorithm, then applying a traditional community detection algorithm. The algorithms belonging to this class are defined by the flattening method and by the single-layer community detection algorithm applied to the flattened network. The simplest flattening method consists in creating an unweighted graph where two nodes are adjacent if their corresponding actors are adjacent on any of the input layers [5]. The advantage of this approach is that the resulting graph is easier to handle, because there are more clustering algorithms for simple graphs than for weighted graphs and weights often imply an additional level of complexity, e.g., deciding a threshold above which weighted edges should be considered. A potential disadvantage is that an unweighted flattening is more susceptible to noise. Weighted flattenings reflect some structural properties of the original multiplex network in the form of weights assigned to the output edges [5, 62]. In theory these methods are less susceptible to noise, but the resulting communities may be biased

towards edges appearing on several layers, and the results can be more difficult to interpret because of the weights.

Aggregation methods first apply traditional community detection algorithms to each layer, then merge their results. As a consequence, these methods include actors in the same community only when they are part of the same community in at least one layer. There are three types of layer-by-layer approaches in the literature [85]. The pattern mining approach exploits association rule mining methods, which are among the main data-mining tasks used to find objects that frequently co-occur together in different transactions. ABACUS [6] considers each single-layer community as a transaction, so that the final communities contain actors that are part of the same community in at least a minimum number of layers. The second way to merge the result of single-layer community detection methods is based on a notion of consensus (c.f. Sect. 2.2.2): given a set (or ensemble) of community structure solutions from the individual layers, the goal is to find a single, meaningful solution that is representative of the input ensemble, by optimizing an objective function that is designed to aggregate information from the individual solutions in the ensemble. While early approaches such as the one in [73] are limited to use a clustering ensemble method as a black-box tool for combining multiple clustering solutions from a single-layer network, the first wellprincipled formulation of the ensemble-based multilayer community detection (EMCD) problem, provided in [114], does not limit aggregation at node membership level, but rather it accounts for intra-community and inter-community connectivity. The consensus solution discovered by EMCD is the one with maximum multilayer modularity from a search space of candidates delimited by topological upper-bound and lower-bound solutions, respectively, of the input multilayer network. Finally, some methods in the literature process the layer-specific adjacency matrices, or derived matrices, and extend spectral-clustering for simple graphs by exploiting the relationship between the eigenvectors and eigenvalues in the constructed matrices and the presence of clusters in the corresponding graphs. As an example, the principal modularity maximization (PMM) method [119] extracts structural features from the various layers, then concatenates the features and performs PCA to select the top eigenvectors. Using these eigenvectors, a low-dimensional embedding is computed to capture the principal patterns across the layers, finally a simple k-means is applied to assign nodes to communities. Further details on this class of approaches can be found in [117].

The third class of algorithms operates directly on the multiplex network model. As an example, a method belonging to this class based on a random walker would allow the walker to switch from one layer to the other, which would not be possible if the layers have been flattened or if we want to separately identify communities on individual layers. Various approaches originally developed for simple graphs have been extended to the multilayer case. Density-based methods first identify dense regions of the network, then include adjacent regions in the same community. A popular method for simple graphs is clique percolation, where dense regions correspond to cliques and adjacency consists in having common nodes. The multilayer clique percolation method extends this process by looking for cliques spanning multiple layers, and redefining adjacency so that both common nodes and common layers are required [120]. Methods based on random walks consider that an entity randomly following the edges in a network would tend to get trapped inside communities, because of the higher edge density between nodes inside the same community, less frequently moving from one community to the other. LART [76] and Multiplex-Infomap [28] are both based on this consideration. LART first runs a different random walk for each layer, then a dissimilarity measure between nodes is obtained leveraging the per-layer transition probabilities. Finally, a hierarchical clustering method is used to produce a hierarchy

of communities which is eventually cut at the level corresponding to the best value of multislice modularity [93]. Multiplex-Infomap is an extension to multiplex networks of the classic Infomap algorithm [107]. Infomap is a search algorithm that minimizes the flow-based map equation model, which relies on the principle that communities are detected as groups of nodes among which the flow, based on a random walk model, persists for a long time once entered. In the multidimensional case [28], the proposed flow dynamics allows the random walker to transition both within each layer and across layers. Several of the reviewed algorithms in the multilayer class use an objective function that, given an assignment of the nodes to communities, returns a higher value when there are more edges inside communities and less edges across communities. Once the objective function has been defined, then different optimization methods can be used to identify a community assignment corresponding to a high value of the function. Generalized Louvain [93], the best-known method in this class, uses an extended version of modularity. This class also includes a method returning a different type of communities with respect to the ones generated by the other algorithms, where edges are grouped instead of actors and nodes [91]. Finally, the multilayer class includes an algorithm based on label propagation [10]. A traditional label propagation method would start assigning a different label to each node, then having each node replace its label with one that is frequent among its neighbors, until some stopping condition is satisfied. The multilayer version of this approach follows the same idea, weighting the contribution of each neighbor based on their similarity with the node on the different layers. For example, two nodes being adjacent on all layers and having the same neighbors on all layers would have a higher probability of getting the same label.

2.2.2 Consensus clustering

Consensus clustering [97], also known as clustering ensembles [110, 31], or aggregation clustering [45], deals with the following problem: given a set of clustering solutions (called ensemble), derive a consensus clustering as a (new) clustering by the optimization of a certain objective function (the consensus function) which expresses how well any candidate consensus clustering complies with the solutions in the ensemble.

Clustering ensemble methods are based on the idea that, due to algorithmic peculiarities of any specific clustering method, a single clustering solution may not be able to capture all facets of a given clustering problem. In this case, it is useful to generate an ensemble by varying one or more aspects of the clustering process, such as the clustering algorithm, the parameter setting, or the number of clusters, and eventually obtain a consensus clustering by properly "aggregating" the information in the ensemble. In Section 2.2.2, we discuss background on consensus clustering, while in Section 2.2.2 we review consensus clustering methods for community detection.

Basic consensus data clustering

Consensus clustering methods are here presented under three main categories [52]: *instance-based* methods, *cluster-based,* and *hybrid* approaches.

Instance-based methods are developed to carry out a direct comparison between data objects. Most instance-based methods operate on the *co-occurrence* or *co-association* matrix $\mathbf{M}$. For each pair of objects (i, j), this matrix stores the number of solutions of the ensemble in which i and j are placed in the same cluster, divided by the size of the ensemble. In the Majority Voting (MV) algorithm [37], $\mathbf{M}$ is "cut" at a given threshold, i.e., all objects whose pairwise entry in $\mathbf{M}$ is greater than the

threshold are joined in the same cluster. This approach has been proved to be equivalent to an agglomerative hierachical clustering algorithm with single link metric on $\mathbf{M}$, cutting the resulting dendrogram according to the given threshold [38]. Other algorithms are based on using M either as a new data matrix or as a pair-wise distance matrix involved in a specific clustering algorithm. The authors in [111] map the consensus clustering problem to a graph/hypergraph partitioning problem. They present two instance-based methods, namely the Cluster-based Similarity Partitioning Algorithm (CSPA) and the HyperGraph Partitioning Algorithm (HGPA). CSPA induces a weighted graph from M and partitions it using the well-known graph partitioning algorithm METIS [67]. HGPA builds a hypergraph whose vertices are the data objects and the hyperedges are given by the clusters of all the clustering solutions in the ensemble; the consensus clustering is then obtained by partitioning the hypergraph using HMETIS [66].

Cluster-based consensus clustering approaches are based on the principle "to cluster clusters". The key idea is to run a clustering algorithm on the set of clusters contained in all clustering solutions of the ensemble, in order to compute a set of meta-clusters. The consensus clustering is finally computed to assign each data object to the meta-cluster that maximizes some assignment criterion (e.g., majority voting) [52]. As an example, the authors in [11] proposes a two-stage clustering procedure. In the first stage, clustering solutions are obtained by multiple runs of the K-Means algorithm. Then, the output centroids from these clustering solutions are clustered by an additional run of K-Means, and the resulting meta-centroids are used as initial points for a complete run of Expectation Maximization or K-Means.

Hybrid consensus clustering methods attempt to combine ideas coming from both instance-based and cluster-based approaches. The objective is to build a hybrid bipartite graph whose vertices belong to the sets of objects and clusters [52]. The Hybrid Bipartite Graph Formulation (HBGF) algorithm [34] builds a bipartite graph where each edge (i, C) has weight equal to 1, if the object i belongs to the cluster C, 0 otherwise. The clustering ensemble result is obtained by partitioning the graph according to standard methods such as METIS, or spectral graph partitioning algorithms. The Weighted Bipartite Partitioning Algorithm (WBPA) [30] follows the same overall scheme of HBGF, although it extends the range of weight values from $\{0, 1\}$ to $[0, 1]$.

Extension to graph data

One of the first approach adopting a consensus clustering paradigm to graph clustering is the one proposed in [77]. This method is a simple approach for consensus clustering in weighted networks, whose general scheme is as follows: given a weighted graph G, a selected community detection algorithm A, a desired number of clusterings n_p, and a real-valued threshold θ, it performs the steps: (i) apply A on G n_p times to obtain n_p clusterings, (ii) build the co-association matrix $\mathbf{M}$ (without any constraint on node linkage) and threshold it using θ, (iii) apply A on $\mathbf{M}$ n_p times; (iv) if the obtained clusterings are all equal then stop, otherwise go back to step (ii). To avoid the generation of disconnected components, an additional step at the end of each iteration is to restore the entries in the co-association matrix with the highest weight among the pruned ones.

As discussed in Sect. 2.2.1, some existing aggregation methods for community detection in multilayer networks adopt the consensus clustering approach [114]. Given a multilayer network $G_{\mathcal{L}}$, an *ensemble of community structures* for $G_{\mathcal{L}}$ is a set $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\ell\}$, such that each $\mathcal{C}_h$ (with $h = 1..\ell$) is a community structure of the layer

graph G_h. This ensemble could be obtained by applying any non-overlapping community detection algorithm to each layer graph (c.f. Sect 2.2.1).

The authors in [114] address the problem of ensemble-based multilayer community detection (EMCD): given an ensemble of community structures for a multilayer network, the problem is to compute a *consensus community structure*, as a set of communities that are representative of how nodes were grouped and topologically-linked together over the layer community structures in the ensemble. In order to determine the community membership of nodes in the consensus structure, a *co-association*-based scheme is defined over the layers, to detect a clustering solution (i.e., the consensus) that conforms most to the input clusterings. Given $G_{\mathcal{L}}$, and $\mathcal{E}$ for $G_{\mathcal{L}}$, the *co-association matrix* $\mathbf{M}$ is a matrix with size $|\mathcal{V}| \times |\mathcal{V}|$, whose (i, j)-th entry is defined as $|m_{ij}|/\ell$, where m_{ij} is the set of communities shared by $v_i, v_j \in \mathcal{V}$, under the constraint that the two nodes are linked to each other [114].

Differently from other consensus methods, in [114] aggregation is not limited at node membership level, but it also accounts for intra-community and inter-community connectivity. EMCD is modeled in [114] as an optimization problem in which the *consensus community structure* solution is optimal in terms of *multilayer modularity*, and is to be discovered within a hypothetical space of consensus community structures that is delimited by a "topological-lower-bound" solution and by a "topological-upper-bound" solution, for a given *co-association threshold* θ. Intuitively, the topological-lower-bound solution may be poorly descriptive in terms of multilayer edges that characterize the internal connectivity of the communities, whereas the topological-upper-bound solution may contain superfluous multilayer edges connecting different communities. The modularity-optimization-driven consensus community structure produced by the method in [114], dubbed M-EMCD, hence produces a solution that is ensured to have higher modularity than both the topologically-bounded solutions.

2.2.3 Uncertain graph clustering

The problem of clustering probabilistic graphs has been recently studied by [75]. Classical techniques and methods that are used for partitioning graphs into clusters could, in principle, be applied to probabilistic graphs by either considering the edge probabilities as weights or by leaving out probabilities smaller than a specific threshold. However, for the first approach the main problem is that it cannot solve the clustering problem for weighted probabilistic graphs, because once the probability is considered as weight, the actual weights cannot be encoded meaningfully onto the edges. Instead, for the second approach, the problem is that the threshold value cannot be computed in a principled and reliable way. This second problem is also discussed in the context of consensus community detection in multllayer graphs in Chapter 3.

The objective of existing methods for clustering uncertain graphs is similar to community detection in deterministic graphs: they aim to maximize the intra-cluster connectivity and minimize the inter-cluster connectivity according to the possible world semantics. This translates into seeking for a clustering such that nodes in the same cluster are densely connected and nodes in different clusters are poorly linked on average in different possible worlds. A novel clustering problem on probabilistic graphs, where both intra-cluster and inter-cluster connectivity is maximized, is introduced in Chapter 6.

pKwikCluster algorithm

In [75] the problem of clustering in probabilistic graphs is formulated as an optimization problem by resorting to the notion of (expected) edit distance between graphs.

Given two deterministic graphs $G = (V, E_G)$ and $Q = (V, E_Q)$, the edit distance between G and Q, denoted as $D(G, Q)$, is the number of edges that need to be deleted or added from G in order to be transformed into Q. In other words,

$$D(G, Q) = |E_G \setminus E_Q| + |E_Q \setminus E_G|$$

Let A^G (resp. A^Q) denote the 0-1 adjacency matrix of G (resp. of Q), the edit distance between G and Q can also be defined as:

$$D(G, Q) = \sum_{u,v;u<v} |A^G_{uv} - A^Q_{uv}|$$

This definition can be extended to the case where one of the two graphs is a probabilist graph. In this case, the edit distance between a probabilistic graph $\mathcal{G} = (V, P)$ and a deterministic graph $Q = (V, E_Q)$ is the expected edit distance between every $G \sqsubseteq \mathcal{G}$ and Q. That is,

$$D(\mathcal{G}, Q) = \mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}}[D(G, Q)] = \sum_{G \sqsubseteq \mathcal{G}} Pr[G] D(G, Q) \tag{2.2}$$

We could naively compute Eq. 2.2 by generating all possible worlds of $\mathcal{G}$ and for each $G \sqsubseteq \mathcal{G}$ computing $Pr[G]$ through Eq. 2.1. However, this computation is exponential in the the size of the probabilistic graph. However, the expected edit distance in Eq. 2.2 can also be computed in polinomial time with the following [75]:

$$D(\mathcal{G}, Q) = \sum_{(u,v) \in E_Q} (1 - p_{uv}) + \sum_{(u,v) \notin E_Q} p_{uv} \tag{2.3}$$

Another central notion in [75] is the *cluster graph*, which is a particular deterministic graph that consists of vertex-disjoint disconnected cliques. More formally, a cluster graph $C = (V, E_C)$ is a deterministic graph with the following properties: (i) C defines a partition of the nodes $V = \{V_1, \ldots, V_k\}$; (ii) $\forall i \in \{1, \ldots, k\}$ and for each pair of nodes $u, v \in V_i$, $(u, v) \in E_C$; (iii) $\forall i, j \in \{1, \ldots, k\}$, $i \neq j$ and for each pair of nodes $u \in V_i$, $v \in V_j$ $(u, v) \notin E_C$.

The authors in [75] formulate the problem of clustering in probabilistic graph as the following optimization problem: given a probabilistic graph $\mathcal{G} = (V, P)$, find the cluster graph $C = (V, E_C)$ such that $D(\mathcal{G}, C)$ is minimized.

The algorithm presented in [75], dubbed *pKwikCluster*, is a randomized expected $5 - approximation$ algorithm. It starts with a random node u and creates a cluster with all neighbors of u that are connected with u with probability higher that 0.5. If no such node exists, u defines a singleton cluster. Having u and its cluster neighbors removed, the algorithms proceeds with the rest of the graph until all nodes are assigned to its proper cluster. The cost of the algorithm is linear in the size of the input graph, thus scalable to large probabilistic graphs. Note that the number of clusters in output is not part of the input. In fact, the objective function itself dictates the number of clusters that are appropriate for every input.

2.3 Learning problems for understanding graph dynamics

The following two subsections provide a brief overview of bandit and preference-based learning paradigms while the last subsection introduces their application for understand graph dynamics.

2.3.1 Reinforcement learning and combinatorial multi-armed bandit (CMAB)

Reinforcement learning (RL) aims to learn optimal actions from a finite set of available actions through continuously interacting with an unknown environment. In contrast to supervised learning techniques, reinforcement learning does not need an experienced agent to show the correct way, but adjusts its future actions based on the obtained feedback signal from the environment. There are three key elements in a RL agent, i.e., states, actions and rewards. At each instant a RL agent observes the current state, and takes an action from the set of its available actions for the current state. Once an action is performed, the RL agent changes to a new state, based on transition probabilities. Correspondingly, a feedback signal is returned to the RL agent to inform it about the quality of its performed action. RL is defined as how an agent should take actions in an environment so to maximize some notion of cumulative reward. RL acts optimally through trial-and-error interactions with an unknown environment. Actions may affect not only the immediate reward but also the next situation and all subsequent rewards. Since the main goal is maximizing the cumulative reward, the intuition underlying RL is that actions that lead to large rewards should be made more likely to recur. However, a further key aspect is to achieve a trade-off between making decisions that yield high current rewards, or *exploitation*, and making decisions that discard immediate gains in favor of better future rewards, or *exploration*.

Multi-armed bandits (MAB) are a simplified setting of RL where the agent is not involved into learning to act in more than one situation, i.e. there is only one possible state. In the traditional MAB framework, there exists a set $\mathcal{A}$ of m *arms* (or actions), associated with a set of random variables $\{X_i^t \ | 1 \leq i \leq m, t \geq 1\}$, whose values range in $[0, 1]$. X_i^t indicates the random outcome of *triggering*, or playing, the i-th arm in the t-th round. The random variables $\{X_i^t | \ t \geq 1\}$ associated to the i-th arm are independent and identically distributed; note however this holds in a stationary context, but more in general, in a *non-stationary context*, those variables may change [54]. Also, variables of different arms may not be independent. At each step t the agent selects/plays an arm $a_t \in \mathcal{A}$ and the reward $X_{a_t}^t$ is revealed. Let T denotes the number of steps, the goal of a bandit algorithm is selecting the proper actions in order to maximize the expected cumulative reward $R(T)$:

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T} X_{a_t}^t\right]$$

Combinatorial multi-armed bandit (CMAB) is an extension of MAB that introduces the concept of *superarm* as a set of (base) arms that can be triggered together [20, 41]. At each round t, a superarm $A_t \subseteq \mathcal{A}$ is chosen and the outcomes of the random variables $X_{a_t}^t$, for all $a_t \in A_t$, are revealed. Moreover, the base arms belonging to A_t may probabilistically trigger other base arms not in A_t, thus revealing their associated outcomes. Selecting a super arm at each step corresponds to solve a particular combinatorial problem [20]. Let $R(A_t)$ be a random variable denoting the reward obtained at round t by playing superarm A_t. This reward depends, linearly or non-linearly, on the base arms that constitute the superarm and other possibly triggered base arms. The objective of a CMAB method is to select at each round t the superarm A_t that maximizes the expected reward $\mathbb{E}[R(A_t)]$, in order to eventually maximize the expected cumulative reward over all rounds, i.e. $R(T) = \mathbb{E}[\sum_{t=1}^{T} R(A_t)]$ According to the exploration-exploitation trade-off, at each trial the bandit may decide to choose the superarm with the highest expected reward (given the current mean estimates

for the base arms) or to select a superarm discarding information from earlier rounds with the aim of discovering the benefit from adopting some previously unexplored arm(s) [41, 20].

2.3.2 Preference-based top-k selection

Consider a finite set of decision *alternatives*, or *options*, $\mathcal{O} = \{o_1, \ldots, o_N\}$, for which the following assumptions hold: (i) the options in this set are pairwise comparable, (ii) there exists a finite number of samples, from an unknown pairwise-preference distribution, that provide information about whether or not an option might be preferred to another one, and (iii) the samples could be "noisy" (i.e., they could significantly vary w.r.t. the unknown distribution model).

The *preference-based top-k selection* problem [15] is to choose the set of k options ($k < N$) that maximize the *preference* over all alternatives, which is formally equivalent to the following optimization problem:

$$\underset{S \subset \mathcal{O}, |S|=k}{\operatorname{argmax}} \sum_{o_i \in S} \sum_{o_j \in \mathcal{O} \wedge j \neq i} \mathbb{I}\{o_j \prec^{\mathcal{R}} o_i\}, \tag{2.4}$$

where $\prec^{\mathcal{R}}$ is a strict preference order relation according to a predefined ranking model $\mathcal{R}$, such that $o_j \prec^{\mathcal{R}} o_i$ means the option o_i is preferred to o_j, and $\mathbb{I}\{\cdot\}$ is the indicator function which is equal to 1 if the argument is true, 0 otherwise. Note also that, given that the outcomes of the pairwise comparisons could be noisy and the available number of samplings are limited, the optimality of the solution to Eq. 2.4 should be guaranteed with probability at least $1 - \delta$, for any predefined probability δ; typically, $\delta = 0.1$.

To quantify the pairwise preferences, the outcome of a comparison between o_i and o_j is modeled as a random variable $Y_{i,j}$, which assumes value 0 (resp. 1) if $o_i \prec o_j$ (resp. $o_i \succ o_j$), and a "neutral" 1/2 otherwise. Given a pair o_i, o_j and a set of $n_{i,j}$ realizations of their comparison $\{y_{i,j}^{(1)}, ..., y_{i,j}^{(n_{i,j})}\}$ of $Y_{i,j}$, assumed to be independent, the expected value $y_{i,j} := \mathbb{E}[Y_{i,j}]$ can be estimated as:

$$\bar{y}_{i,j} = \frac{1}{n_{i,j}} \sum_{l=1}^{n_{i,j}} y_{i,j}^{(l)}. \tag{2.5}$$

Ranking models

A ranking model $\mathcal{R}$ produces a complete order of the options in $\mathcal{O}$ upon the preference relation matrix $\mathbf{Y} = [y_{i,j}]_{N \times N} \in [0,1]^{N \times N}$. Example of ranking models are: (i) *Copeland's ranking* (CO) [92], (ii) weighted voting, or *sum of expectations* (SE), and (iii) *random walk* (RW) ranking.

Copeland's ranking determines that option o_i is preferred to option o_j ($o_j \prec^{CO} o_i$) if and only if $b_j < b_i$, where $b_i = |\{o_h \in \mathcal{O} \mid y_{i,h} > \frac{1}{2}\}|$, i.e., whenever o_i beats more options that o_j does. According to sum of expectations ranking, $o_j \prec^{SE} o_i$ holds if and only if $\sum_{h \neq j} y_{j,h} < \sum_{h \neq i} y_{i,h}$. Random walk ranking first requires a left-stochastic version $\mathbf{S} = [s_{ij}]_{N \times N}$ of the matrix $\mathbf{Y}$, such that $s_{i,j} = \frac{y_{i,j}}{\sum_{l=1}^{N} y_{l,j}}$. Then, the ranking of options is determined as the stationary probability distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ of the Markov chain underlying $\mathbf{S}$. Finally, the options are ranked according to the computed probabilities, i.e., $o_j \prec^{RW} o_i$ iff $\pi_j < \pi_i$.

Algorithm 1 PBR($\mathbf{Y}, \mathcal{O}, k, n_{max}, \delta, \mathcal{R}$)

1: $S = D \leftarrow \emptyset$ {*Set of selected (S) and discarded (D) options*}
2: Initialize with zeros: $\mathbf{B} = [c_{i,j}]_{N\times N}$, $\mathbf{B_u} = [u_{i,j}]_{N\times N}$, $\mathbf{B}_\ell = [l_{i,j}]_{N\times N}$ {*Confidence bound matrices*}
3: $n_{i,j} \leftarrow 0,\ \forall o_i, o_j \in \mathcal{O}$ {*Sample counts*}
4: $A \leftarrow \{(o_i, o_j) | i \neq j, 1 \leq i, j \leq |\mathcal{O}|\}$ {*Set of active option pairs*}
5: **while** $(n_{i,j} \leq n_{max}, \forall i \forall j) \wedge |A| > 0$ **do**
6: **for all** $(o_i, o_j) \in A$ **do**
7: $n_{i,j} \leftarrow n_{i,j} + 1$
8: $y_{i,j}^{(n_{i,j})} \sim Y_{i,j}$ { *Sample from the pairwise preference probability distribution*}
9: **end for**
10: Update $\bar{\mathbf{Y}} = [\bar{y}_{i,j}]_{N\times N}$ with the new samples {*Using Eq. (2.5)*}
11: **for** $i, j = 1$ to N **do**
12: $c_{i,j} \leftarrow \sqrt{\frac{1}{2n_{i,j}} \log \frac{2N^2 n_{max}}{\delta}}$ {*Update Hoeffding confidence bounds* $\mathbf{B_u}, \mathbf{B}_\ell, \mathbf{B}$}
13: $l_{i,j} \leftarrow \bar{y}_{i,j} - c_{i,j}, \quad u_{i,j} \leftarrow \bar{y}_{i,j} + c_{i,j}$
14: **end for**
15: $(A, S, D) \leftarrow$ SamplingStrategy$(\mathcal{R}, A, \bar{\mathbf{Y}}, N, k, \mathbf{B_u}, \mathbf{B}_\ell, \mathbf{B}, D)$ {*Algorithm 2*}
16: **end while**
17: **return** $S, \bar{\mathbf{Y}}$

Algorithm 2 SamplingStrategy$(\mathcal{R}, A, \bar{\mathbf{Y}}, N, k, \mathbf{B_u}, \mathbf{B}_\ell, \mathbf{B}, D)$

1: $S \leftarrow optionsToSelect(A, \mathbf{B}_\ell, \mathbf{B_u}, N, k, \mathcal{R})$
2: $D \leftarrow D \cup\ optionsToDiscard(A, \mathbf{B}_\ell, \mathbf{B_u}, N, k, \mathcal{R})$
3: **for** $(o_i, o_j) \in A$ **do**
4: **if** $!isStillToUpdate((o_i, o_j), S, D, \mathbf{B}_\ell, \mathbf{B_u}, \mathbf{B}, \bar{\mathbf{Y}}, \mathcal{R})$ **then**
5: $A = A \setminus \{(o_i, o_j)\}$
6: $S \leftarrow$ top-k options according to $\mathcal{R}$
7: **return** (A, S, D)

Preference-based Racing

The authors in [15] proposed a preference-based racing (PBR) procedure, shown in Algorithm 1, which is responsible for identifying, among the set of options $\mathcal{O}$, the top-k ones according to a predefined ranking model $\mathcal{R}$. Besides $k, \mathcal{R}$, and the probability guarantee (δ, cf. Sect. 2.3.2), the algorithm requires an additional parameter, n_{max}, to control the number of samplings for each pairwise preference probability distribution (i.e., $Y_{i,j}$, with $o_i, o_j \in \mathcal{O}$).

The algorithm also maintains a set of active pairs of options (A), i.e., options whose pairwise preference distributions need to be sampled more in order to decide which one is better, with enough high degree of *confidence*. Racing methods employ confidence intervals, typically computed through the Hoeffding bound, derived from the concentration property of the mean estimate [15]. To this purpose, Algorithm 1 maintains the estimates $y_{i,j}$ with their confidence intervals $[\ell_{i,j}, u_{i,j}]$ and iteratively samples from the pairwise preference distribution until there is enough confidence about the top-k nodes or the maximum number of samplings is reached (Line 5). According to the updates values of confidence bounds, the set of current selected options S (i.e., top-k ones) and discarded options D (not top-k) are updated. This is handled by procedure SamplingStrategy (Line 15), which is sketched in Algorithm 2.

According to [15], Algorithm 2 initially checks if some options can be included

among the top-k or discarded ones with high enough probability (Lines 1 and 2). This step is performed differently according to the ranking model $\mathcal{R}$ (cf. Sect. 2.3.2), whereby the confidence intervals are used to decide with high probability that an option is better or worse than another. Next we provide details about the different sampling strategies.

- CO-based strategy: the aforementioned step is performed by counting, for each option o_i, the set of better options $w_i = |\{o_j \mid l_{i,j} > 1/2,\ i \neq j\}|$ and worse options $z_i = |\{o_j \mid u_{i,j} < 1/2,\ i \neq j\}|$. Then, an option o_i is among the top-k options with high probability if $|\{o_j \mid |\mathcal{O}| - z_i < w_j\}| > |\mathcal{O}| - k$ while it is to be discarded if $|\{o_j \mid |\mathcal{O}| - w_i < z_j\}| > k$.

- SE-based strategy: in this case, first the ranking score definition is applied to lower/upper bounds, i.e., for each option o_i, the averages $l_i = \frac{1}{|\mathcal{O}|-1}\sum_{j\neq i} l_{i,j}$ and $u_i = \frac{1}{|\mathcal{O}|-1}\sum_{j\neq i} u_{i,j}$ are computed. Then, similarly to the CO case, an option o_i is included among the top-k options with high probability if $|\{o_j \mid u_j < l_i\}| > |\mathcal{O}| - k$ and discarded if $|\{o_j \mid u_i < l_j\}| > k$.

- RW-based strategy: when RW is used as ranking model, selecting and discarding of option is based on the exploitation of properties of the stationary distribution of transition matrices. Random walk ranking first requires a left-stochastic version $\bar{\mathbf{S}} = [s_{ij}]_{N\times N}$ of the matrix $\bar{\mathbf{Y}}$, such that $s_{i,j} = \frac{y_{\bar{i},j}}{\sum_{l=1}^{N} y_{\bar{l},j}}$. Given confidence intervals for the entries of matrix $\bar{\mathbf{Y}}$, denoted with matrix $\mathbf{B} = [c_{i,j}]_{N\times N}$, confidence intervals for elements in $\bar{\mathbf{S}}$, denoted with matrix $\tilde{\mathbf{B}} = [\tilde{c}_{i,j}]_{N\times N}$, are computed (by applying a result in [3]) as:

$$\tilde{c}_{ij} = \frac{N}{3}\max_k c_{k,j}\sum_l \bar{y}_{l,j} \tag{2.6}$$

  Note that the elements of a particular column of $\tilde{\mathbf{B}}$ are equal to each other, thus $\|\tilde{\mathbf{B}}\|_1 = \max_j \sum_i |\tilde{c}_{i,j}| = \frac{N^2}{3}\max_{k,j} c_{k,j}\sum_l \bar{y}_{l,j}$.

  Let $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_N)$ and $\bar{\boldsymbol{\pi}} = (\bar{\pi}_1, \ldots, \bar{\pi}_N)$ be the stationary distributions of S and $\bar{S}$ respectively. Then, by applying the result of [39], it follows that an upper bound on the difference between the estimated stationary distribution and the unknown true one can be computes as:

$$\|\boldsymbol{\pi} - \bar{\boldsymbol{\pi}}\|_{max} \leq \|\tilde{\mathbf{B}}\|_1 \|\bar{\mathbf{A}}^*\|_{max} \tag{2.7}$$

  where $\bar{\mathbf{A}}^* = [\bar{a}^*_{ij}]_{N\times N} = (I - \bar{\mathbf{S}} + \mathbf{1}\boldsymbol{\pi}^T)^{-1} - \mathbf{1}\boldsymbol{\pi}^T$. Notice that, in order to obtain better estimates of the preferences, the bound in Eq. 2.7 suggests the minimization of $\|\tilde{\mathbf{B}}\|_1$ which can be performed by sampling pairs $(i,j) = \text{argmax}_{i,j}\, c_{i,j}\sum_l \bar{y}_{l,j}$. At each time, the pairs of options that satisfy this condition are maintained as set of active options to be sampled next.

For each active pair of options (o_i, o_j), a condition is checked (Line 4) to decide whether it is not necessary anymore to sample from the pairwise preference distribution of (o_i, o_j) — this holds either because with high probability o_i is better (resp. worse) than o_j or because one of the two options need to be selected (resp. discarded).

2.3.3 Applications to dynamic community detection and network inference

The discussed learning paradigms (in Sect. 2.3.1 and 2.3.2) are conceived as learning tools in contexts where data are characterized by uncertainty, noise and limitedness. This also holds in time-evolving networks and in this thesis we show how CMAB and preference-based methods can help understanding graph dynamics.

In particular, the CMAB paradigm can be profitably adopted to temporal community detection problems since the inherent uncertainty and dynamicity in such network systems. Existing approaches share the nature of graph-based unsupervised learning paradigm to address the community detection problem in temporal networks. However, this may not be in principle the best way to do, primarily because of the inherent uncertainty about the environment i.e., the temporal network system, and the interactions within it, i.e., structural changes and consequent decisions to take about the node memberships and structure of the communities. Unlike traditional (un)supervised learning, RL is instead conceived to learn from interrelated actions with unknown "rewards" ahead of time, and choose which actions to take in order to maximize the reward; in the community detection problem in time-evolving graphs, this corresponds to evaluate the benefit of making a set of node assignments to communities at a particular time. Chapter 4 shows how the community detection problem in temporal networks can be mapped to a CMAB instance where a super arm corresponds to a set of base actions, i.e., a set of community assignments that constitute a whole community-structure. Moreover, the exploration-exploitation trade-off translates into the balancing over time between the need for embedding long-term changes observed in the community formation and the need for capturing short-term effects and newly observed community structures.

A trust network is a graph of entities (i.e., individuals) that are linked through asymmetric relationships that correspond to subjective trust statements. Trust relations forms from users' interactions thus must be determined from the available information in a social environment, e.g., the history of users' activities and their interactions. Chapter 5 shows how the preference-based learning paradigm can be exploited for the trust network inference problem: given a sequence of timestamped interaction networks as input, the goal of TNI is to infer from this sequence a directed weighted network, whose nodes are the users in the temporal networks and links denote trust relationships with associated trust scores. Computing trust relations is however a particularly difficult task, because of a number of challenges that already arise at data source level (i.e., not considering the inevitable bias of the particular algorithmic solution to the problem). In fact, the amount of information representing the observed interactions and activities of users in a social network, could be limited in size as well as in quality. More specifically, a social network may contain a significant amount of redundant or irrelevant relations as well as noise in the information that express the strength of interaction between any two users. Preference learning paradigm provides key advantages in addressing all the aforementioned issues, i.e., limitedness, redundancy and noisy of the information about the users' interactions from which a trust network is to be inferred. More specifically, under a preference-based top-k selection problem, the goal is to find a ranking of the preferential pairings that each target entity would choose to form its trust relationships. To this purpose, an adaptive sampling strategy can be adopted and instantiated according to three canonical ranking models that correspond to different levels of ranking pairwise preferences.

Chapter 3

Consensus Community Detection in Multilayer Networks.

Summary. The clustering ensemble paradigm has emerged as an effective tool for community detection in multilayer networks, which allows for producing consensus solutions that are designed to be more robust to the algorithmic selection and configuration bias. However, one limitation is related to the dependency on a co-association threshold that controls the degree of consensus in the community structure solution. The goal of this work is to overcome this limitation with a new framework of ensemble-based multilayer community detection, which features parameter-free identification of consensus communities based on generative models of graph pruning that are able to filter out noisy co-associations. We also present an enhanced version of the modularity-driven ensemble-based multilayer community detection method, in which community memberships of nodes are reconsidered to optimize the multilayer modularity of the consensus solution. Experimental evidence on real-world networks confirms the beneficial effect of using model-based filtering methods and also shows the superiority of the proposed method on state-of-the-art multilayer community detection.

3.1 Introduction

Multilayer networks are pervasive in many fields related to network analysis and mining [74, 26]. Particularly, *community detection in multilayer networks* (ML-CD) has attracted lot of attention in the past few years, as witnessed by a relatively large corpus of studies (see, e.g., [72] for a survey).

An effective approach to ML-CD corresponds to *aggregation methods*, whose goal is to infer a community structure by combining information from community structures separately obtained on each of the layers [119, 118, 114]. A special class of such methods resembles theory on *clustering ensemble* [110, 51]: given a set of clusterings as different groupings of the input data, a *consensus* criterion function is optimized to induce a single, meaningful solution that is representative of the input clusterings. A key advantage of using a consensus clustering approach is that the inconvenience of guessing the "best" algorithm selection and parametrization is avoided, and hence consensus results will be more robust and show higher quality when compared to single-algorithm clustering.

Despite the well-recognized benefits of using the consensus/ensemble clustering paradigm, its exploitation to ML-CD is, surprisingly, relatively new in the literature [118, 77, 114]; actually, to the best of our knowledge, only the most recent of these works goes beyond the use of a clustering ensemble approach as a black-box tool for ML-CD, by proposing the first well-principled formulation of the *ensemble-based community detection* (EMCD) problem. Indeed, in [114], aggregation is not limited at node membership level, but it also accounts for intra-community and inter-community

connectivity; moreover, the consensus function is optimized via *multilayer modularity* analysis, instead of being simply based on the sharing of a certain minimum percentage of clusters in the ensemble.

The EMCD method proposed in [114] relies on a *co-association-based consensus clustering scheme*, i.e., the consensus clusters are derived from a co-association matrix built to store the fraction of clusterings in which any two nodes are assigned to the same cluster. Low values in this matrix would reflect unlikely consensus memberships, i.e., noise, and hence should be removed; to this purpose, the matrix is subjected to a filtering step based on a user-specified parameter of minimum co-association, θ. Unfortunately, setting an appropriate θ for a given input network is a challenging task, since too low values will lead to few, large communities, while too high values will lead to many, small communities. Moreover, this approach generally fails to consider properties related to node distributions and linkage in the network.

In this work, we aim to overcome the above issue, by proposing a new EMCD framework featuring a *parameter-free identification of consensus clusters* from which the *consensus community structure* will be induced. Our idea is to exploit a recently developed class of *graph-pruning methods based on generative models*, which are designed to filter out "noisy" edges from weighted graphs. A key advantage of these pruning models is that they do not require any user-specified parameter, since they enable edge-removal decisions by computing a statistical p-value for each edge based on a null model defined on the node degree and strength distributions. We originally introduce these models to multilayer community detection and propose an adaptation to multilayer networks.

Another limitation of EMCD is that the community membership of nodes remains the same through the process of detecting the modularity-driven consensus community structure. In this work, we also address this point, by defining a three-stage process in the EMCD scheme, which iteratively seeks to improve the multilayer modularity of the consensus community structure based on intra-community connectivity refinement, community partitioning, and relocation of nodes from a community to a neighboring one.

Two main findings are drawn from experimental results obtained on real-world multiplex networks: (i) some of the model-filters are effective in simplifying an input multilayer network to support improved community detection, and (ii) our proposed framework outperforms state-of-the-art multilayer community detection methods according to modularity and silhouette quality criteria.

In the rest of the chapter, we provide background on generative-model-based filters (Section 3.2) and on the existing EMCD method (Section 3.3). Next, we present our proposed framework (Section 3.4). Experimental evaluation and results are discussed in Sections 3.5 and 3.6. Section 3.7 provides a summary of the chapter contents.

3.2 Generative models for graph pruning

Pruning is a graph simplification task aimed at detecting and removing irrelevant or spurious edges in order to unveil some hidden property/structure of the network, such as its organization into communities. A simple technique adopted in weighted graphs consists in removing all edges having weight below a pre-determined, global threshold. Besides the difficulty of choosing a proper threshold for the input data, this approach tends to remove all ties that are weak at network level, thus discarding local properties at node level.

A relatively recent corpus of study addresses the task of filtering out "noisy" edges from complex networks based on *generative null models*. The general idea is to define a null model based on node distribution properties, use it to compute a p-value for every edge (i.e., to determine the statistical significance of properties assigned to edges from a given distribution), and finally filter out all edges having p-value above a chosen significance level, i.e., keep all edges that are least likely to have occurred due to random chance.

Methods following the above general approach have been mainly conceived to deal with weighted networks, so that the node degree and/or the node strength (i.e., the sum of the weights of all incident edges) are used to generate a model that defines a random set of graphs resembling the observed network. One of the earliest methods is the *disparity* filter [108], which evaluates the strength and degree of each node locally. This filter however introduces some bias in that the strength of neighbors of a node are discarded. By contrast, a global null model is defined with the *GloSS* filter [103], as it preserves the whole distribution of edge weights. The null model is, in fact, a graph with the same topological structure of the original network and with edge weights randomly drawn from the empirical weight distribution. Unlike disparity and GloSS, the null model proposed by Dianati [25] is maximum-entropy based and hence unbiased. Upon it, two filters are defined: the *marginal likelihood filter* (*MLF*), which is a linear-cost method that assigns a significance score to each edge based on the marginal distribution of edge weights, and the *global likelihood filter*, which accounts for the correlations among edges. While performing similarly, the latter filter is more costly than MLF; moreover, both consider the strength of nodes, but not their degrees. Recently, Gemmetto et al. [43] proposed a maximum-entropy filter, *ECM*, for keeping only *irreducible edges*, i.e., the filtered network will retain only the edges that cannot be inferred from local information. ECM employs a null model based on the canonical maximum-entropy ensemble of weighted networks having the same degree and strength distribution as the real network [89]. Next, we report details of the MLF, GloSS and ECM filters whose impact on a community detection task is assessed in Sect. 3.4.

The MLF filter [25].

Given an undirected weighted graph $G = (V, E, w)$, with T unit edges,[1] the null model assigns each edge to a pair of nodes, which are selected independently and randomly with probabilities proportional to their strengths. Given two nodes v_i and v_j, the probability that an edge is associated to them is given by $p_{ij} = \frac{s_i s_j}{2T^2}$, where s_i is the strength of node v_i and $T = \frac{1}{2}\sum_{v_i} s_i$. The probability that w out of T unit edges will choose nodes v_i and v_j as their endpoints is given by a binomial probability with probability of success w and number of trials T. Therefore, the null model defines for every couple of nodes a probability for their edge weight, σ_{ij}, depending on their strengths. For each edge (v_i, v_j) with weight w_{ij}, its p-value, denoted as γ_{ij}, is computed as:

$$\gamma_{ij} = \sum_{w \geq w_{ij}} \Pr[\sigma_{ij} = w | s_i, s_j, T] = \sum_{w \geq w_{ij}} \binom{T}{w} p_{ij}^{w} (1 - p_{ij})^{T-w} \tag{3.1}$$

According to this null model, the higher the strengths of two nodes, the higher the weight of an edge connecting them is to be in order to be considered statistically

[1] Each weighted edge can be seen as multiple edges of unit weight.

significant. Conversely, the lower the strengths of the two nodes, the lower the weight of a linking edge must be in order to be retained by the filter.

The GloSS filter [103].

The null model in GloSS is a graph with the same topology of the original network but with edge weights randomly extracted from the empirical weight distribution. The probability to observe in the null model a weight w_{ij} on an edge (v_i, v_j), given the strengths and degrees of the two end point nodes, is computed as:

$$\Pr[\sigma_{ij} = w_{ij}|k_i, k_j, s_i, s_j] = P(w_{ij})\frac{P(s_i, s_j|w_{ij}, k_i, k_j)}{P(s_i, s_j|k_i, k_j)} \tag{3.2}$$

where k_i denotes the degree of node v_i, $P(w_{ij})$ is estimated as the fraction of edges of the input network having weight w_{ij}, while $P(s_i, s_j|k_i, k_j)$ is a normalizing factor. In order to compute the term on the numerator, it should be considered that, given w_{ij}, k_i, k_j, the null model must set the weights on the remaining $k_i - 1$ and $k_j - 1$ edges such that $\sum_{v_h \neq v_j} w_{ih} = s_i - w_{ij}$ and $\sum_{v_h \neq v_i} w_{jh} = s_j - w_{ij}$. According to this:

$$P(s_i, s_j|w_{ij}, k_i, k_j) = F(s_i - w_{ij}, k_i - 1) \times F(s_j - w_{ij}, k_j - 1) \tag{3.3}$$

where $F(s, k)$ is the probability of randomly extracting, from the distribution $P(w)$, k weights that sum to s. Finally, the p-value for an edge of weight w_{ij} is computed as:

$$\gamma_{ij} = \sum_{w \geq w_{ij}} \Pr[\sigma_{ij} = w \mid k_i, k_j, s_i, s_j] = \frac{\sum_{w \geq w_{ij}} P(w)P(s_i, s_j|w, k_i, k_j)}{\sum_{w \geq 0} P(w)P(s_i, s_j|w, k_i, k_j)} \tag{3.4}$$

The ECM filter [43].

It is based on a null model that constrains both the degree and strength sequence on average and, at the same time, requires that the probability distribution, $P(G)$, describing a canonical ensemble of graphs maximizes the Shannon's entropy. The solution to this maximization problem is found to be the probability distribution $P(G|\mathbf{x}, \mathbf{y}) = \prod_{i<j} q_{ij}(w_{ij}|\mathbf{x}, \mathbf{y})$, where $\mathbf{x}, \mathbf{y}$ are two $|V|$-dimensional vectors of Lagrange multipliers, with components $x_i \geq 1$ and $0 \leq y_i < 1, \forall v_i \in V$, which are used to impose the expected degrees and strengths of the nodes, and the term q_{ij} represents the probability that a link of weight w exists between nodes v_i and v_j:

$$q_{ij}(w|\mathbf{x}, \mathbf{y}) = \frac{(x_i x_j)^{\vartheta(w)}(y_i y_j)^w(1 - y_i y_j)}{1 - y_i y_j + x_i x_j y_i y_j} \tag{3.5}$$

where $\vartheta(w) = 1$ if $w > 0$, otherwise $\vartheta(w) = 0$. According to the above equation, the formation of a link of unit weight between two nodes has a different cost (i.e., higher if $x_i x_j > 1$) than the reinforcement of an existing one. The p-value associated to an edge (v_i, v_j) of weight w_{ij} is hence computed as:

$$\gamma_{ij} = \sum_{w \geq w_{ij}} \Pr[\sigma_{ij} = w] = \sum_{w \geq w_{ij}} q_{ij}(w|\mathbf{x}, \mathbf{y}) \tag{3.6}$$

3.3 Ensemble-based Multilayer Community Detection

This section provides a summary of the main definitions and algorithms proposed in [114], upon which the rest of the chapter is based.

Definition 2 (Ensemble of community structures) *Given a multilayer network $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$, with $\ell = |\mathcal{L}|$ layers, an ensemble of layer-specific community structures for $G_{\mathcal{L}}$ is a set $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\ell\}$, such that each $\mathcal{C}_h$ (with $h = 1..\ell$) is a a community structure of the layer graph G_h.*

The information provided by an ensemble of community structures can be summarized through a co-association matrix.

Definition 3 (Co-association matrix) *Given a multilayer network $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$, and an ensemble of community structures $\mathcal{E}$ for $G_{\mathcal{L}}$, the co-association matrix $\mathbf{M}$ is a matrix with size $|\mathcal{V}| \times |\mathcal{V}|$ and such that the (i, j)-th entry stores the number of communities shared by $v_i, v_j \in \mathcal{V}$, subject to the condition that the two nodes are linked to each other, divided by the number of layers (i.e., the size of the ensemble): $M(i, j) = \frac{|m_{ij}|}{\ell}$, where $m_{ij} = \{h \mid L_h \in \mathcal{L} \wedge \exists C \in \mathcal{C}_h, \mathcal{C}_h \in \mathcal{E},\ s.t.\ v_i, v_j\ in\ C \wedge (v_i, v_j) \in E_h\}$.*

Definition 4 (Consensus community structure) *Given a multilayer network $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$ and an ensemble of community structures $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\ell\}$ (with $\ell = |\mathcal{L}|$) defined over $G_{\mathcal{L}}$, a consensus community structure for $\mathcal{E}$ is a partitioning of a graph with nodes in $\mathcal{V}$ and edges in $E_{\mathcal{L}}$, which is representative of the community structures in $\mathcal{E}$.*

Problem 1 (Ensemble-based Multilayer Community Detection (EMCD)) *Given a multilayer network and an ensemble of layer-specific community structures for it, determine a consensus community structure from the ensemble.*

The above definitions correspond to meta-definitions since they capture only the intuition that the consensus should agree with the community structures in the ensemble, but no hint is provided about the quality the consensus should have. However, according to the general desiderata in community detection, each community in the consensus should have high internal connectivity and low external connectivity. Also, since the input is a multilayer graph, these two requirements should hold accross all layers. In order to solve Problem 1, the authors in [114] propose two baseline methods, namely C-EMCD and CC-EMCD, as well as a greedy algorithm, dubbed M-EMCD, whose main details are reported next.

C-EMCD

This first baseline method in [114] requires the co-association matrix $\mathbf{M}$ (c.f. Definition 3) to infer a clustering of $\mathcal{V}$, denoted as $\mathcal{S}$. In order to retain only meaningful co-association values, $\mathbf{M}$ is subjected to a filtering step based on a user-specified parameter of minimum co-association $\theta \in [0, 1]$. This pruning step is also beneficial in terms of efficiency because otherwise $\mathbf{M}$ would be a very dense matrix, which would make any clustering process computationally expensive. The row (or column) projections corresponding to the entries greater than or equal to θ, identify a clustering of $\mathcal{V}$, where each cluster contains nodes that are ensured to be (directly or indirectly) linked together in $G_{\mathcal{L}}$. Finally, a consensus community structure is obtained where each community corresponds to the multilayer subgraph of $G_{\mathcal{L}}$ induced from each of the clusters in $\mathcal{S}$. Note that any consensus community will correspond to a connected subgraph, but not necessarily to a maximal complete subgraph of the multilayer network.

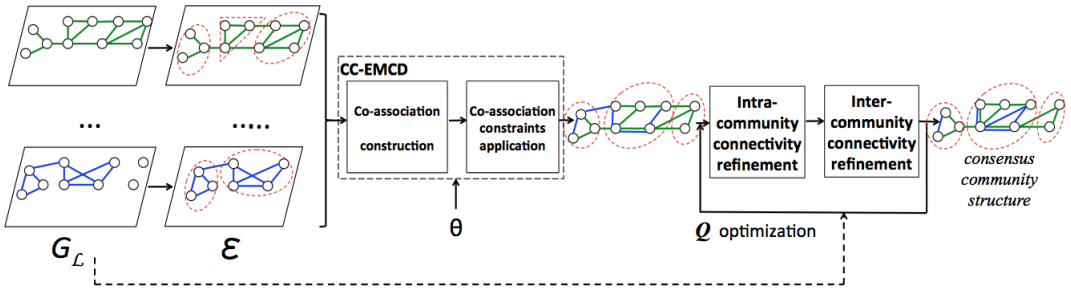


FIGURE 3.1: Overview of the modularity-based EMCD framework [114].

CC-EMCD

The topological lower-bound consensus solution in the EMCD problem in [114] produces a consensus community structure from information derived from the co-association matrix $\mathbf{M}$ and such that the structure of each consensus community only considers those specific layers that allow any two nodes to be connected in the shared community. More specifically, for each cluster S derived from $\mathbf{M}$, a community is obtained as the subgraph $C = \langle V, E \rangle$ of the multilayer network $G_{\mathcal{L}}$ with a set of nodes $V = S$ and a set of edges $E = \{(v_i, v_j, h) \in E_{\mathcal{L}} \mid v_i, v_j \in V \wedge h \in m_{ij}\}$.

Analogously, any two consensus communities are connected by using only the fraction of the multilayer graph that actually involves the connection of nodes from one community to another. Specifically, only edges are selected that correspond to those layers in which any two nodes are *not* present in the co-association matrix, i.e., given two communities $C^{(1)}, C^{(2)} \in \mathcal{C}^*$, with node sets $V^{(1)}, V^{(2)}$, the set of edges linking them is computed as $E(C^{(1)}, C^{(2)}) = \{(v_i, v_j, h) \in E_{\mathcal{L}} \mid v_i \in V^{(1)}, v_j \in V^{(2)} \wedge h \notin m_{ij}\}$.

M-EMCD

Intuitively, the topological-lower-bound solution may be poorly descriptive in terms of multilayer edges that characterize the internal connectivity of the communities, whereas the topological-upper-bound solution may contain superfluous multilayer edges connecting different communities. EMCD is thus modeled in [114] as an optimization problem in which the consensus community structure solution is optimal in terms of *multilayer modularity*, and is to be discovered within a hypothetical space of consensus community structures that is delimited by a "topological-lower-bound" solution and by a "topological-upper-bound" solution, for a given co-association threshold θ. The modularity-optimization-driven consensus community structure produced by the greedy algorithm in [114], dubbed M-EMCD, hence produces a solution that is ensured to have higher modularity than both the topologically-bounded solutions. Figure 3.1 sketches an overview of M-EMCD. Given an input multilayer graph $G_{\mathcal{L}}$ and an ensemble $\mathcal{E}$ for it, and a co-association threshold θ, the CC-EMCD method is first employed to produce the "lower-bound" consensus community structure. Then, this consensus is iteratively refined through two main steps, respectively within-community and across-community, until modularity is optimized to return the final consensus community structure. Note that the refinement is performed to preserve the topology of $G_{\mathcal{L}}$, according to the "upper-bound" consensus. Details on the adopted multilayer modularity in [114], as well as in this chapter, are reported next.

Definition 5 (Multilayer modularity [114]) *Given a community structure* $\mathcal{C} = \{C_1, \ldots, C_k\}$ *for* $G_{\mathcal{L}}$*, the multilayer modularity of* $\mathcal{C}$ *is defined as follows:*

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} Q(C) = \frac{1}{d(V_{\mathcal{L}})} \sum_{C \in \mathcal{C}} \sum_{L \in \mathcal{L}} \left(d_L^{int}(C) - \gamma_L \frac{(d_L(C))^2}{d(V_{\mathcal{L}})} + \beta \sum_{L' \in \mathcal{P}(L)} d_{L,L'}^{ext}(C) \right) \tag{3.7}$$

where, for any community $C \in \mathcal{C}$, $d_L(C)$ and $d_L^{int}(C)$ are respectively the degree of C and the internal degree of C according to the only edges of layer L, $d(V_{\mathcal{L}})$ is the total degree of the entire graph, i.e., $d(V_{\mathcal{L}}) = \sum_{L \in \mathcal{L}} \sum_{v \in V_L} d(v)$, γ_L is a resolution parameter for edges of layer L (set to 1, by default), $d_{L,L'}^{ext}(C)$ is the external degree of C, i.e., twice the sum of inter-layer edges involving nodes inside C, $\beta \in \{0, 1\}$ (set to 0, by default), and $\mathcal{P}(L)$ is the set of valid pairings with L defined as:

$$\mathcal{P}(L) = \begin{cases} \{L' \in \mathcal{L} \mid L \prec_{\mathcal{L}} L'\}, & \text{if } \prec_{\mathcal{L}} \text{ is defined} \\ \mathcal{L} \setminus \{L\}, & \text{otherwise} \end{cases}$$

where $\prec_{\mathcal{L}}$ is a partial order relation over the set of layers $\mathcal{L}$.

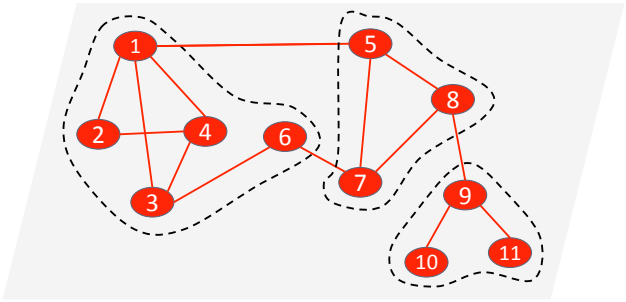
3.4 EMCD and parameter-free graph pruning

As previously discussed, the EMCD framework has one model parameter, i.e., the co-association threshold θ, which allows the user to control the degree of consensus required to every pair of nodes in order to appear in the same consensus community. Given a selected value for θ and any two nodes v_i, v_j, we say that their community linkage, expressed by $M(v_i, v_j)$, is considered as meaningful to put the nodes in the same consensus community iff $M(v_i, v_j) \geq \theta$.

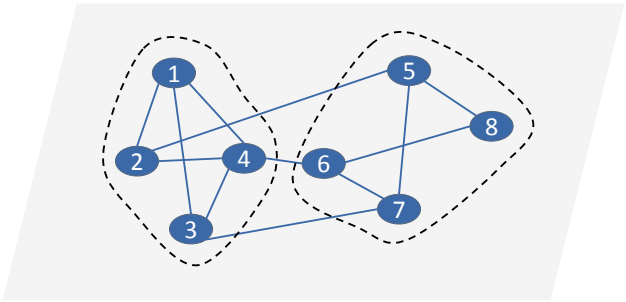
However, choosing a fixed value of θ equally valid for all pairs of nodes raises a number of issues. First, there is an intrinsic difficulty of guessing the "best" threshold — since too low values will lead to few, large communities, while too high values will lead to many, small communities. Second, the approach ignores any property of the input network, and consequently a single-shot choice of θ may fail to capture the natural structure of communities. Of course, to overcome the two issues in practical cases, one could always try different choices of the parameter and finally select the best-performing one (e.g., in terms of modularity, as done in [114]), but it is clear that the approach does not scale for large networks.

It would instead be desirable to evaluate the significance of the co-associations by taking into account the topology of the multilayer network, so that a relatively low value of co-association might be retained as meaningful provided that it refers to node relations that make sense only for certain layers, while on the contrary, a relatively high value of co-association could be discarded if it corresponds to the linkage of nodes that have high degree and co-occur in the same community in many layers — in which case, the co-association could be considered as superfluous in terms of community structure.

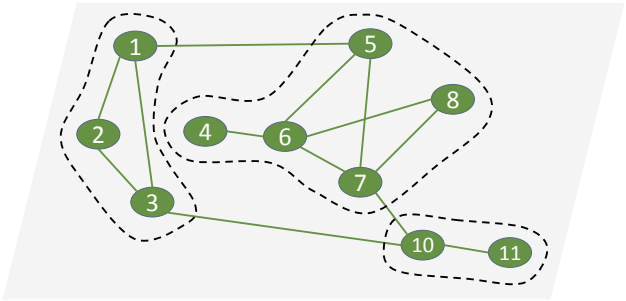
In order to fulfill the above requirement, we define a *parameter-free approach to EMCD* that exploits the previously discussed pruning models. Since such models are only designed to work with (monoplex) weighted graphs, our key idea is to first infer a *weighted graph representation of the co-association matrix* associated to a multilayer network and its ensemble of community structures, and then apply a pruning model on it to retain only meaningful co-associations.



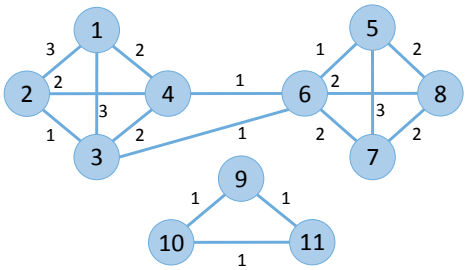
(a) Layer L_1



(b) Layer L_2



(c) Layer L_3



(d) Co-association graph

FIGURE 3.2: Community structures (denoted by dotted curves) on a 3-layer network, and corresponding co-association graph.

Definition 6 (Co-association graph) *Given a multilayer graph $G_{\mathcal{L}}$, an ensemble $\mathcal{E}$ of community structures defined over it, and associated co-association matrix $\boldsymbol{M}$, we define the co-association graph $G_M = \langle V_M, E_M, w \rangle$ as an undirected weighted graph such that $V_M = \mathcal{V}, E_M = \{(v_i, v_j) \mid m_{ij} \neq \emptyset, w_{ij} = |m_{ij}|\}$.*

Below is an example of how the pruning of the co-association graph based on a user-specified threshold could lead to poorly meaningful consensus communities.

Example 1 *Consider the 3-layer network and associated co-association graph in Figure 3.2. Focusing on the community membership of nodes, consider the following*

Algorithm 3 Co-association matrix filtering

Input: Multilayer graph $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$, ensemble of community structures $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\ell\}$ (with $\ell = |\mathcal{L}|$), generative model for graph pruning *WGP*.
Output: Filtered co-association matrix $\mathbf{M}$ for $G_{\mathcal{L}}$ and $\mathcal{E}$.

1: Let α be a statistical significance level (i.e., $\alpha = 0.05$) {*Co-association matrix initialization*}
2: $\mathbf{M} \leftarrow matrix(|\mathcal{V}|, |\mathcal{V}|)$
3: **for** $(i, j) \in \mathbf{M}$ **do**
4: $\quad m_{ij} \leftarrow \{h \mid L_h \in \mathcal{L} \wedge \exists C \in \mathcal{C}_h, \mathcal{C}_h \in \mathcal{E},\ s.t.\ v_i, v_j\ in\ C \wedge (v_i, v_j) \in E_h\}$
5: $\quad M(i, j) \leftarrow |m_{ij}|/\ell$
6: **end for**
7: $G_M = \langle V_M, E_M, w \rangle \leftarrow$ build_coassociation_graph$(G_{\mathcal{L}}, \mathbf{M})$ {*Using Def. 6*}
8: $(e, \gamma_{ij})_{e=(v_i,v_j)\in E_M} \leftarrow$ compute_pValues(G_M, WGP) {*Using Def. 7*}
9: **for** $(v_i, v_j) \in E_M$ **do**
10: $\quad$ **if** $\gamma_{ij} \geq \alpha$ **then** $M(i, j) \leftarrow 0$ {*Null hypothesis cannot be rejected*}
11: **return** $\mathbf{M}$

settings of a cutting threshold θ. For any $\theta \leq 1/3$, all edges will be kept (as the minimum valid weight is 1) and hence the co-association graph will be partitioned into the two communities corresponding to its two connected components, i.e., $\{1, .., 8\}$ and $\{9, 10, 11\}$; setting $1/3 < \theta \leq 2/3$ will lead to $\{1, .., 4\}$, $\{5, .., 8\}$, and $\{9\}, \{10\}, \{11\}$; finally, for $2/3 < \theta \leq 1$, the communities will be $\{1, 2, 3\}$, $\{5, 7\}$ and all the other nodes as singletons. It should be noted that no setting of θ can enable the identification of the three "natural" consensus communities, i.e., $\{1, .., 4\}$, $\{5, .., 8\}$, and $\{9, 10, 11\}$.

Definition 7 (Co-association hypothesis testing) *Given a co-association graph $G_M = \langle V_M, E_M, w \rangle$, let WGP denote a statistical inference method whose generative null model is parametric w.r.t. node degree and strength distributions in G_M. We define the co-association hypothesis testing as a parametric testing based on WGP, whose null hypothesis for every observed edge is that its weight has been generated by mere chance, given the empirical strength and degree distributions, and the associated p-value is the probability that the null model produces a weight equal to or greater than the observed edge weight. If the p-value is lower than a desired significance level, then the null hypothesis can be rejected, which implies that the co-association of the two observed nodes is considered as statistically meaningful.*

Algorithm 3 shows the general scheme of creation of the co-association matrix, for a given multilayer network and associated ensemble of community structures, and its filtering based on the co-association hypothesis testing.

Enhanced M-EMCD (M-EMCD*).

We propose an enhanced version of M-EMCD that has two main advantages w.r.t. the early M-EMCD method in [114]: (1) it incorporates parameter-free pruning of the co-association matrix described in Algorithm 3, and (2) it fixes the inability of the early M-EMCD in reconsidering the community memberships of nodes during the consensus optimization.

Algorithm 4 shows the pseudo-code of our proposed enhanced M-EMCD, dubbed M-EMCD*. Initially, the filtered co-association matrix computed by a selected model-filter *WGP* is provided as input to CC-EMCD, which computes the initial (i.e., lower-bound) consensus community structure (Line 2) [114]. This is iteratively improved in a

Algorithm 4 Enhanced Modularity-driven Ensemble-based Multilayer Community Detection (M-EMCD*)

Input: Multilayer graph $G_{\mathcal{L}} = (V_{\mathcal{L}}, E_{\mathcal{L}}, \mathcal{V}, \mathcal{L})$, ensemble of community structures $\mathcal{E} = \{\mathcal{C}_1, \ldots, \mathcal{C}_\ell\}$ (with $\ell = |\mathcal{L}|$), generative model for graph pruning *WGP*.
Output: Consensus community structure $\mathcal{C}^*$ for $G_{\mathcal{L}}$.

1: $\mathbf{M} \leftarrow$ co-associationMatrixFiltering($G_{\mathcal{L}}, \mathcal{E}$, *WGP*) {*Algorithm 3*}
2: $\mathcal{C}_{lb} \leftarrow$ CC-EMCD($G_{\mathcal{L}}, \mathbf{M}$) {*Compute topological-lower-bound consensus community structure*}
3: $\mathcal{C}^* \leftarrow \mathcal{C}_{lb}$
4: **repeat**
5: **for** $L_i \in \mathcal{L}$ **do**
6: $Q \leftarrow Q(\mathcal{C}^*)$
{*Refine intra-community connectivity of* C_j}
7: **for** $C_j \in \mathcal{C}^*$ **do**
8: $\langle C'_j, Q'_j \rangle \leftarrow$ update_community($\mathcal{C}^*, C_j, L_i$)
9: $j^* \leftarrow \operatorname{argmax} Q'_j$
10: **if** $Q'_{j^*} > Q$ **then** $Q \leftarrow Q'_{j^*}$, $\mathcal{C}^* \leftarrow \mathcal{C}^* \setminus C_j \cup C'_{j^*}$
{*Refine inter-community connectivity between* C_{j^*} *and each of its neighbors*}
11: **for** $C_h \in N(C_{j^*})$ **do**
12: $\langle \mathcal{C}_h^{IC}, Q_h^{IC} \rangle \leftarrow$ update_community_structure($\mathcal{C}^*, C_{j^*}, C_h, L_i$)
13: $\langle \mathcal{C}_h^{R}, Q_h^{R} \rangle \leftarrow$ relocate_nodes($\mathcal{C}^*, C_{j^*}, C_h$)
14: $\langle \mathcal{C}_h, Q_h \rangle \leftarrow \operatorname{argmax}\{Q_h^{IC}, Q_h^{R}\}$
15: $h^* \leftarrow \operatorname{argmax} Q_h$
16: **if** $Q_{h^*} > Q$ **then**
17: $Q \leftarrow Q_{h^*}$, $\mathcal{C}^* \leftarrow \mathcal{C}_{h^*}$
18: **if** $Q_{h^*} = Q_{h^*}^R$ **then** $\langle \mathcal{C}_h, Q_h \rangle \leftarrow$ update_community_structure($\mathcal{C}^*, C_{j^*}, C_{h^*}, L_i$)
19: **else** $\langle \mathcal{C}_h, Q_h \rangle \leftarrow$ relocate_nodes($\mathcal{C}^*, C_{j^*}, C_{h^*}$)
20: **if** $Q_h > Q$ **then** $Q \leftarrow Q_h$, $\mathcal{C}^* \leftarrow \mathcal{C}_h$
{*Evaluate partitioning of* C_{j^*} *into smaller communities*}
21: $\langle C'_s, Q'_s \rangle \leftarrow$ partition_community($\mathcal{C}^*, C_{j^*}$)
22: **if** $Q'_s > Q$ **then** $Q \leftarrow Q'_s$, $\mathcal{C}^* \leftarrow \mathcal{C}^* \setminus C_{j^*} \cup C'_s$
23: **end for**
24: **until** $Q(\mathcal{C}^*)$ cannot be further maximized
25: **return** $\mathcal{C}^*$

three-stage modularity-optimization process: (i) refinement of connectivity internal to a selected community, (ii) refinement of connectivity between the community and its neighbors also involving relocation of nodes, and (iii) partitioning of the community.

The within-community connectivity refinement step (Lines 7-10) consists in seeking in the current solution $\mathcal{C}^*$ the community C_{j^*} whose internal connectivity modification leads to the best modularity gain. The internal refinement of a community C_j, applied to the layer L_i, is performed by function update_community (Line 8) which tries to add as many edges of type L_i as possible between nodes belonging to C_j, i.e., the set of edges in E_i whose end-nodes are both in C_j and are not present in the current solution $\mathcal{C}^*$. The function then returns the modified C_j and the updated modularity.

Once identified the community C_{j^*} at the previous step, the algorithm tries to relocate nodes from C_{j^*} to its neighbor communities $N(C_{j^*})$ and/or to refine its external connectivity with them (Lines 11-20). The inter-community connectivity refinement is carried out by function update_community_structure (Line 12) which, for any layer L_i and neighbor communities C_j,C_h, evaluates the resulting modularity of adding and/or removing edges of type L_i in the current consensus $\mathcal{C}^*$ between C_j,C_h, compatibly with the set of edges of L_i in the original graph. The relocation of one node at a time from C_{j^*} to a neighbor community C_h is evaluated by relocate_nodes

TABLE 3.1: Main features of real-world multiplex network datasets used in our evaluation. Mean and standard deviation over the layers are computed for degree, average path length, and clustering coefficient measures

	#entities ($\|\mathcal{V}\|$)	#edges	#layers (ℓ)	node set coverage	edge set coverage	degree	avg. path length	clust. coef.
AUCS	61	620	5	0.73	0.20	10.43 ± 4.91	2.43 ± 0.73	0.43 ± 0.097
EU-Air	417	3 588	37	0.13	0.03	6.26 ± 2.90	2.25 ± 0.34	0.07 ± 0.08
FAO-Trade	214	318346	364	0.53	0.003	6.40 ± 4.69	2.53 ± 0.51	0.31 ± 0.11
FF-TW-YT	6 407	74 836	3	0.58	0.33	9.97 ± 7.27	4.18 ± 1.27	0.13 ± 0.09
London	369	441	3	0.36	0.33	2.12 ± 0.16	11.89 ± 3.18	0.036 ± 0.032
VC-Graders	29	518	3	1.00	0.33	17.01 ± 6.85	1.66 ± 0.22	0.61 ± 0.89

(Line 13) until there is no further improvement in modularity. The ordering of node examination is determined by a priority queue that gives more importance to nodes having more edges (of any type) towards C_h than edges linking them to nodes in their current community in $\mathcal{C}^*$.

The step of partitioning of C_{j^*} into smaller communities is carried out by function partition_community (Line 21). While this can in principle refer to the use of any (multilayer) modularity-optimization-based community detection method, we choose here to focus on the membership of nodes, and hence to devise this step in the simplified scenario of flattened representation of the consensus community C_{j^*}, i.e., a weighted monoplex graph with all and only the nodes belonging to C_{j^*} and weights expressing the number of layers on which two nodes are linked in $\mathcal{C}^*$. Upon this representation, we apply a graph clustering method based on modularity optimization (cf. Sect. 3.5) and finally maintain the resulting partitioning only if it led to an improvement in modularity.

3.5 Evaluation methodology

Datasets. We used six real-world multiplex networks for our evaluation, which are among the most frequently used in recent, relevant studies in multiplex/multilayer community detection.

AUCS [72] captures the interaction among university employees. They are: (i) work together, (ii) lunch together, (iii) off-line friendship, (iv) friendship on Facebook, and (v) co-authorship. EU-Air transport network [72] describes the connections between European airports from different airlines. FriendFeed, Twitter, and YouTube network (FF-TW-YT) [26] was designed by using the social media aggregator FriendFeed for modeling the interactions of users who were also members of Twitter and YouTube. London transport network [128] includes three types of connections between train stations in London: (i) underground lines, (ii) overground, and (iii) DLR. 7thGraders (VC-Graders) [128] models the interaction among students in terms of friendship, work together, and affinity relations in the class. FAO Trade network (FAO-Trade) [29] represents different types of trade relationships among countries, obtained from FAO (Food and Agriculture Organization of the United Nations).

Table 3.1 summarizes main characteristics of the evaluation networks. For each network we report the number of entities $|\mathcal{V}|$, the number of total edges, the average coverage of the node set is computed as $1/|\mathcal{L}| \sum_{L\in\mathcal{L}}(|V_L|/|\mathcal{V}|)$, and the average coverage of edge set is $1/|\mathcal{L}| \sum_{L\in\mathcal{L}}(|E_L|/\sum_{L'}|E_{L'}|)$.

We used six networks for our evaluation (Table 3.1), which are among the most frequently used in relevant studies in multilayer community detection.

Competing methods. We selected four of the most representative methods for multilayer community detection: *Generalized Louvain* (GL) [93], *Multiplex Infomap* (M-Infomap) [28], *Principal Modularity Maximization* (PMM) [119], and the consensus clustering approach in [77] (hereinafter denoted as ConClus). Note that the latter two are aggregation-based methods; in particular, ConClus is a simple approach for consensus clustering in weighted networks.

GL is an extension of the classic Louvain method using multislice modularity, which assigns each node-layer tuple separately to a community.

PMM detects a concise representation of features from the various layers (dimensions) by performing two main steps: (i) structural feature extraction, and (ii) cross-dimension integration. Firstly, modularity maximization is used for extracting structural features from each dimension. Then, the features are concatenated and subjected to PCA to extract the top eigenvectors, corresponding to possible community partitions. Starting from these eigenvectors, a low-dimensional embedding is computed in order to detect the principal patterns across all the dimensions of the network. Finally a simple k-means on this embedding is performed to find the discrete community assignment.

M-Infomap extends the classic *Infomap* algorithm for multiplex networks. In the Infomap search algorithm, the flow-based *map equation* model is minimized, based on the concept that communities are identified as groups of nodes among which the flow, based on a random walk model, persists for a long time once entered.

ConClus is a simple approach for consensus clustering in weighted networks, whose general scheme is as follows: given a weighted graph G, a selected community detection algorithm A, a desired number of clusterings n_p, and a real-valued threshold θ, it performs the steps: (i) apply A on G n_p times to obtain n_p clusterings, (ii) build the co-association matrix $\mathbf{M}$ (without any constraint on node linkage) and threshold it using θ, (iii) apply A on $\mathbf{M}$ n_p times; (iv) if the obtained clusterings are all equal then stop, otherwise go back to step (ii). To avoid the generation of disconnected components, an additional step at the end of each iteration is to restore the entries in the co-association matrix with the highest weight among the pruned ones.

Assessment criteria and setting. We employed the *multilayer modularity* defined in [114], the *multilayer silhouette* defined in [114], and *NMI* [110].

To generate the ensemble for each evaluation network, following the lead of the study in [114], we used the serial version of the Nerstrand algorithm [78], a very effective and efficient method for discovering non-overlapping communities in (single-layer) weighted graphs via modularity optimization. We also used Nerstrand for the community-partitioning step in our M-EMCD*.

As concerns the competing methods, we used the default setting for GL and M-Infomap. We varied the number of communities in PMM from 5 to 100 with increments of 5, and finally selected the value corresponding to the highest modularity. Also, we equipped ConClus with Nerstrand (for the generation of the clusterings), set n_p to the number of layers, and varied θ in the full range (with step 0.01) to finally select the value that determined the consensus clusters with the highest average NMI w.r.t. the initial ensemble solutions.

3.6 Results

3.6.1 Impact of model-filters on M-EMCD*

For every network, we analyzed size, modularity and silhouette of the consensus solution obtained before (i.e., at lower-bound CC-EMCD) and at convergence of the

optimization performed by M-EMCD*, when using either global threshold θ pruning or one among MLF, ECM, and GloSS; in the former case, the value of modularity refers to the consensus solution corresponding to the best-performing θ value. Results are reported in Table 3.2 and discussed next. At the end of this section, we also mention aspects related to time performance evaluation.

Size of consensus solutions.

MLF and ECM tend to produce similar number of communities. By contrast, GloSS is in general much more aggressive than the other models, which causes proliferation of communities in the co-association graph. Also, the final solution by M-EMCD* can differ in size from the initial consensus by CC-EMCD, due to the optimization of modularity.

Modularity analysis.

Looking at the modularity results, besides the expected improvement by M-EMCD* over CC-EMCD in all cases, the following remarks stand out. First, MLF and ECM again behave similarly in most cases, while GloSS reveals to be much weaker; this is clearly also dependent on the tendency by GloSS of heavily pruning the co-association graph, as discussed in the previous analysis on the size of consensus solutions. Second, using MLF or ECM leads to higher modularity w.r.t. the best-performing global threshold, in all networks but VC-Graders. This would support the beneficial effect deriving from the use of a model-filter for the co-association graph matrix; note however that such results should be taken with a grain of salt, since modularity is computed on differently prunings of the same network. Also, FAO-Trade deserves a special mention, since its much higher multigraph density (13.97) and dimensionality (i.e., number of layers) (cf. Table 3.1) also caused a densely connected co-association graph, with average degree of 74, average path length of 1.67, clustering coefficient of 0.64, and 1 connected component. This makes FAO-Trade a difficult testbed for a community detection task, which explains the outcome reported in Table 3.2: 11 consensus communities are produced when using ECM, 41 and 40 with θ-based approach and GloSS, respectively, with most of them singletons and disconnected, and even 1 community for MLF.

It is worth noting that most of the performance gains by M-EMCD* over M-EMCD are obtained for θ-based pruning, but not for model-filter pruning. This would suggest the ability of M-EMCD* of achieving high quality consensus even when a refined model-filter would not be used.

Silhouette and NMI analysis.

In terms of silhouette, the use of model-filter pruning is beneficial to both CC-EMCD and M-EMCD* consensus solutions, where the latter achieve significantly higher silhouette in most cases. Among the filters, again MLF and ECM tend to perform closely — with a slight prevalence of ECM — and better than GloSS (except for VC-Graders, where the number of communities is close to the number of nodes in the co-association graph).

TABLE 3.2: Size and modularity (upper table) and silhouette (bottom table) of lower-bound (CC-EMCD) and M-EMCD* consensus (in brackets, when applicable, the increments over M-EMCD), with or without model-filters.

	CC-EMCD modularity				M-EMCD* modularity				M-EMCD* #communities			
	θ-based	MLF	ECM	GloSS	θ-based	MLF	ECM	GloSS	θ-based	MLF	ECM	GloSS
AUCS	0.60	0.68	0.66	0.21	0.86 *(+0.03)*	**0.91**	**0.91**	0.25	14	13	18	52
EU-Air	0.73	0.60	0.60	0.07	**0.91**	**0.91**	0.90	0.09	274	39	45	397 *(-2)*
FAO-Trade	0.74	0.59	0.30	0.20	**1.00**	**1.00**	0.99 *(+0.29)*	0.99 *(+0.56)*	41 *(+1)*	1 *(-2)*	11 *(+3)*	40 *(-17)*
FF-TW-YT	0.48	0.44	0.44	0.05	0.73 *(+0.12)*	**0.94**	**0.94**	0.05	119 *(+33)*	115	133	5134
London	0.89	0.85	0.85	0.41	0.90	**0.97**	**0.97**	0.49 *(+0.06)*	45	46	46	340 *(-3)*
VC-Graders	0.22	0.33	0.27	-0.01	**0.88** *(+0.54)*	0.44	0.43	0.03 *(-0.01)*	3 *(-8)*	16	17	26 *(-1)*

	CC-EMCD silhouette				M-EMCD* silhouette			
	θ-based	MLF	ECM	GloSS	θ-based	MLF	ECM	GloSS
AUCS	0.07	0.23	0.28	0.14	0.37 *(+0.01)*	0.38	**0.40**	0.15
EU-Air	0.01	0.16	0.18	-0.05	0.09	0.27	**0.30**	0.04 *(-0.02)*
FAO-Trade	-0.06	0.01	0.02	0.01	0.08	1.00 *(+0.91)*	0.06 *(-0.05)*	0.06 *(-0.05)*
FF-TW-YT	0.00	0.06	0.06	0.03	0.00 *(-0.04)*	0.15	0.12	0.03
London	0.14	0.06	0.06	0.03	0.18	**0.20**	**0.20**	0.12 *(+0.04)*
VC-Graders	0.24	0.20	0.21	0.05	0.52 *(+0.23)*	0.24	0.28	**0.83** *(+0.77)*

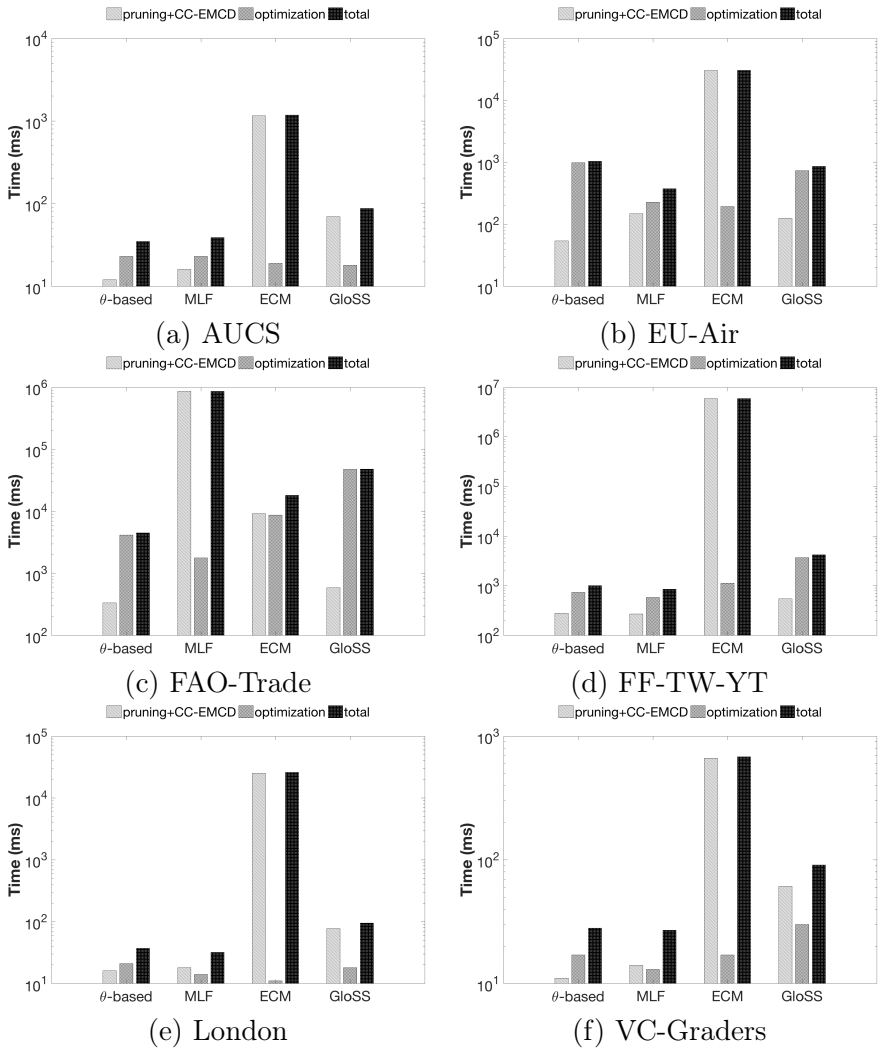


FIGURE 3.3: Time performance by M-EMCD*. Logarithmic scale is used for the y-axis.

We also measured the NMI of M-EMCD and M-EMCD* model-filter consensus solutions vs. the corresponding solutions obtained by θ-based pruning (results not shown). NMI was found very high (above 0.8, up to 1.0) in EU-Air, AUCS, and VC-Graders, around 0.60-0.70 in FF-TW-YT and London, and around 0.40-0.50 in FAO-Trade. Overall, this indicates that the model-filter pruning has similar capabilities as the best θ-based pruning in terms of community membership, though with the advantage of not requiring parameter selection.

Time performance analysis.

We analyzed the M-EMCD* time performance broken down into pruning and CC-EMCD time, modularity optimization time, and total execution time;[2] the former refers either to the execution of one of the model-filters or to the time required for thresholding the co-association matrix based on the best-performing θ, which is in both cases cumulated with the overall time required for generating the consensus solution by CC-EMCD. Figure 3.3 shows results for all networks.

[2]Experiments were carried out on a Linux (Mint 18) machine with 2.6 GHz Intel Core i7-4720HQ processor and 16GB ram

We observe that the various methods lead to substantial differences in the M-EMCD* performance behavior, which is only partly dependent on the number of consensus communities in the respective solutions. When using the θ-based filtering, as expected the pruning time is lower than in all the other cases, while the modularity optimization performed by M-EMCD* is generally slower w.r.t. the optimization time corresponding to the use of any model-filter (all networks) or at least MLF in the largest networks (i.e., FF-TW-YT, FAO-Trade). This may happen even when the consensus communities via θ-based filtering are more numerous than in the model-filter cases, which suggests a beneficial effect of model-based pruning of the co-association graph matrix in the modularity optimization of the multilayer community structure.

Among the model-filters, ECM appears in general to be more costly than GloSS, and this in turn more costly than MLF. The computational gap — at least one order of magnitude — that ECM has w.r.t. the other two filters can be explained since its higher requirements due to its capability of preserving both degree and strength distributions. One exception however corresponds to FAO-Trade: indeed, as previously discussed, this network has a very high density, which implies higher likelihood of co-association of nodes to communities; this causes an increase in the number of unit-weight edges in the MLF co-association graph, thus determining a bottleneck in the computation of the p-value that depends on binomial distributions (see Section 1).

3.6.2 Evaluation with competing methods

Table 3.3 summarizes the increments in terms of size, modularity, silhouette (Table 3.2), and NMI of M-EMCD* solutions w.r.t. the corresponding solutions obtained by each of the competitors, by varying model-filters. For the NMI evaluation, we distinguished two cases: the one, valid for GL, PMM, or M-Infomap, whereby the reference community structure is the solution obtained by the method in case of θ-based pruning, with θ selected according to the best-modularity performance; the other one, valid for ConClus, whereby we computed the average NMI over the layer-specific community structures.

This comparative analysis was focused on the impact of using the various model-filters on the methods' performance. To this end, for every network and model-filter, we first generated an ensemble of layer-specific community structures via Nerstrand, then we built the co-association graph and applied the filter, finally we removed from the original multilayer network the edges pruned by the model-filter, before providing it as input to each of the competing methods.

One general remark is that M-EMCD* equipped with MLF or ECM outperforms all competing methods in terms of both modularity and silhouette, and tends to produce more communities, with very few exceptions. Concerning NMI results for the first three methods, again the increments by M-EMCD* are mostly positive, thus implying that model-filter pruning appears to be more beneficial, w.r.t. a global threshold based pruning approach, for M-EMCD* than GL, followed by PMM and M-Infomap. Also, it is interesting to observe that, with the exception of FAO-Trade for MLF and ECM, M-EMCD* has average NMI of ensemble comparable to or even better than ConClus, whose performance values are optimal in terms of NMI (i.e., the parameter threshold corresponded to the best NMI over each network).

TABLE 3.3: Increments of number of communities, modularity, silhouette and NMI of M-EMCD* solutions, by varying model-filters, w.r.t. corresponding solutions obtained by GL, PMM, M-Infomap, and ConClus.

	Gains by M-EMCD* vs. GL											
	#communities			Modularity			Silhouette			NMI w.r.t. θ-based pruning		
	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS
AUCS	+6	+8	+48	+0.09	+0.08	-0.39	+0.11	+0.10	-0.01	+0.06	+0.21	+0.47
EU-Air	-23	-27	+364	+0.12	+0.11	-0.23	+0.26	+0.29	+0.08	+0.51	+0.48	+0.3
FAO-Trade	-5	+4	+30	+0.53	+0.60	+0.70	+0.97	+0.07	+0.07	-0.55	-0.28	+0.21
FF-TW-YT	+111	+130	+5131	+0.29	+0.27	-0.29	-0.07	-0.10	-0.05	+0.02	+0.05	+0.4
London	+23	+23	+318	+0.05	+0.05	-0.42	+0.08	+0.08	-0.30	-0.14	-0.13	-0.06
VC-Graders	0	+2	+18	-0.23	-0.26	-0.40	+0.15	+0.21	+0.71	+0.3	+0.31	+0.08
	Gains by M-EMCD* vs. PMM											
	#communities			Modularity			Silhouette			NMI w.r.t. θ-based pruning		
	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS
AUCS	-1	+4	+38	+0.43	+0.29	0.00	+0.12	+0.13	-0.04	+0.24	+0.26	+0.18
EU-Air	-47	-41	+311	+0.66	+0.65	+0.04	+0.30	+0.33	+0.12	+0.61	+0.61	+0.47
FAO-Trade	-39	-29	0	+0.91	+0.90	+0.90	+1.02	+0.06	+0.07	-0.61	-0.4	+0.06
FF-TW-YT	+104	+122	+5123	+0.66	+0.60	-0.03	-0.14	-0.15	-0.12	-0.1	-0.11	-0.13
London	+1	+1	+295	+0.26	+0.28	0.00	+0.03	+0.03	-0.02	+0.06	+0.07	+0.16
VC-Graders	+1	+2	+11	-0.05	-0.01	-0.13	+0.25	+0.27	+0.95	+0.24	+0.2	-0.29
	Gains by M-EMCD* vs. M-Infomap											
	#communities			Modularity			Silhouette			NMI w.r.t. θ-based pruning		
	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS
AUCS	+4	+4	+45	+0.18	+0.23	-0.12	+0.17	+0.11	+0.11	+0.48	+0.46	+0.38
EU-Air	-255	-251	+167	+0.38	+0.37	-0.20	+0.35	+0.37	+0.18	+0.74	+0.74	+0.56
FAO-Trade	0	+10	+39	+1.00	0.00	+0.99	+2.00	+1.06	+1.06	0	+0.22	+0.66
FF-TW-YT	+113	+130	+5132	+0.20	+0.24	-0.53	-0.15	-0.15	-0.23	+0.4	+0.3	+0.23
London	+37	+38	+338	+0.52	+0.52	+0.05	+0.21	+0.20	+0.12	+0.39	+0.4	+0.84
VC-Graders	+15	+16	+25	-0.49	-0.50	-0.58	+1.24	+1.28	+1.83	+0.66	+0.64	+0.47
	Gains by M-EMCD* vs. ConClus											
	#communities			Modularity			Silhouette			avg NMI of ensemble		
	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS	MLF	ECM	GloSS
AUCS	+5	+9	+42	+0.33	+0.38	-0.26	+0.13	+0.17	-0.11	-0.03	+0.00	+0.03
EU-Air	-25	-18	+323	+0.71	+0.71	-0.07	+0.23	+0.27	+0.06	-0.05	-0.04	+0.20
FAO-Trade	-16	-11	+21	+0.59	+0.77	+0.74	+0.92	-0.02	-0.01	-0.55	-0.27	+0.01
FF-TW-YT	+17	+74	+4885	+0.48	+0.47	-0.33	+0.15	+0.12	+0.02	-0.06	-0.04	+0.18
London	+16	+21	+298	+0.15	+0.14	-0.30	+0.09	+0.10	-0.01	+0.01	+0.02	+0.12
VC-Graders	+10	+10	+20	+0.21	+0.24	-0.20	+0.09	+0.11	+0.68	+0.02	-0.04	-0.14

3.7 Chapter review

We proposed a new framework for consensus community detection in multilayer networks. This is designed to enhance the modularity-optimization process w.r.t. existing EMCD method. Moreover, by exploiting parameter-free generative models for graph pruning, our framework overcomes the dependency on a user-specified threshold for the global denoising of the co-association graph.

Chapter 4

Dynamic Consensus Community Detection in Temporal Networks.

Summary. Community detection and evolution has been largely studied in the last few years, especially for network systems that are inherently dynamic and undergo different types of changes in their structure and organization in communities. Because of the ineherent uncertainty and dynamicity in such network systems, we argue that temporal community detection problems can profitably be solved under a particular class of multi-armed bandit problems, namely combinatorial multi-armed bandit (CMAB). More specifically, we propose a CMAB-based approach to the novel problem of dynamic consensus community detection, i.e., to compute a single community structure that is designed to encompass the whole information available in the sequence of observed temporal snapshots of a network in order to be representative of the knowledge available from community structures at the different time steps. Unlike existing approaches, the dynamic consensus solution has unique capability of embedding both long-term changes in the community formation and newly observed community structures. Experimental evaluation based on publicly available real-world and ground-truth-oriented synthetic networks, also involving competitors based on evolutionary or consensus approaches, has confirmed the meaningfulness and key benefits of the proposed method.

4.1 Introduction

Community detection in temporal networks is a very active field of research, which can be leveraged for several strategic decisions, including enhanced group-recommendation, user behavior prediction, and evolution of user interaction patterns in relation to real-world events.

The problem of identifying the community behavior at any given time is often jointly considered with the need for modeling the change events in the communities and tracking their evolution [23]. While there exist various models for time-varying network data (i.e., series of snapshots, interval graphs, or interactions), detecting, monitoring and correlating the events of community evolution is particularly challenging. In this regard, one issue is related to making an appropriate choice of timestep width that can provide sufficient resolution to detect temporal events. An even bigger issue is that the community evolution events are of different type (e.g., birth/death, growth/decay, merge/split), and may occur at different rates (i.e., smoothly or drastically, at varying degrees).

Despite the variety of methodologies developed for the community detection and evolution problem, each of the existing approaches is designed to address a limited subset of challenges by adopting a particular perspective on the problem [44]. Some

methods provide heuristics that try to discover a sequence of mappings for the community structures independently derived at each time step; by contrast other methods aim to detect a community structure for the current topology as dependent on the structure(s) from prior time step(s), according to some parameter models to control the temporal smoothness. Further strategies include updating a community structure in order to reflect newly observed changes, or aggregating the various snapshots of the network in order to enable a static community detection method.

All the aforementioned approaches nonetheless share the nature of graph-based unsupervised learning paradigm to address the community detection problem. However, this may not be in principle the best way to do, primarily because of the inherent uncertainty about the *environment*, i.e., the temporal network system, and the *interactions* within it, i.e., structural changes and consequent decisions to take about the node memberships and structure of the communities. Unlike traditional (un)supervised learning, *reinforcement learning* (RL) is instead conceived to learn from *interrelated actions* with *unknown* "rewards" ahead of time, and choose which actions to take in order to maximize the reward; in the problem under consideration in this work, this corresponds to evaluate the benefit of making a set of node assignments to communities. A further key aspect is to achieve a trade-off between making decisions that yield high current rewards, or *exploitation*, and making decisions that sacrifice current gains with the prospect of better future rewards, or *exploration*.

Multi-armed bandit (MAB) problems are well-suited to model the aforementioned trade-off [112, 68]. However, they cannot be directly applied to our problem since they deal with individual actions to take at any time. In this work, we focus on a particular extension of MAB problems, called *combinatorial multi-armed bandit* (CMAB) [20, 41], in order to deal with choosing *a set of* actions, i.e., a set of community assignments that constitute a whole community-structure. Moreover, the exploration-exploitation trade-off translates into the *balancing over time between the need for embedding long-term changes observed in the community formation and the need for capturing short-term effects and newly observed community structures*. Upon this remark, we devise the solution of the problem of community detection in a temporal network by introducing the novel concept of *dynamic consensus community structure*, that is, loosely speaking, a community structure that encompasses the knowledge about newly observed as well as the previously detected communities in a temporal network.

Our contributions can be summarized as follows:

- We formulate the novel problem of dynamic consensus community detection in temporal networks, and originally define a hybrid reinforcement learning/clustering framework based on the combinatorial MAB paradigm.

- To solve the proposed problem, we develop CreDENCE– **C**MAB-based **D**ynamic Cons**EN**sus **C**ommunity D**E**tection method. This is conceived to be versatile in terms of the static community detection approach used to identify the communities at each snapshot, and robust in terms of a number of parameters that control the CMAB-learning rate, temporal smoothness factors, and the node-relocation bias.

- We provide insights into technical as well as computational complexity aspects of CreDENCE. Upon this, we propose an enhancement of CreDENCE to ensure its linear complexity in the size of the temporal network.

- Our experimental evaluation was conducted using 5 real-world networks and ground-truth-oriented synthetically generated networks, including comparison

with 3 competing methods. Results have provided useful indications about the quality of the consensus solutions obtained by CreDENCE, which is able to cope with temporal networks having different evolution rates.

4.2 Related work

In a recently published survey [23], temporal community detection methods are classified into four categories. The first one is to connect the results of community detection tasks independently executed over the different snapshots of the network (e.g., [49, 13, 115]). The second category includes methods that detect communities at each time based on the current snapshot and on communities found in earlier time steps (e.g., [60, 42, 123, 125]). The third category refers to the approach of using a classic community detection method on an aggregated (i.e., multiplex) representation of the snapshots (e.g., [93]). The fourth category assumes that the temporal network is modeled as a series of changes in a network, and hence communities at time t are updated according to local changes occurring at time $t + 1$ (e.g., [127, 106, 1]).

Little research has been conducted on the temporal counterpart of the *consensus community detection* problem [77, 114]. One main issue is that the consensus community structure is to be inferred from a knowledge base (i.e., set of community structures) that is not fully available at a given initial time, but it evolves over time along with the associated temporal network. In [35], La Fond et al. define a representative clustering solution determined by aggregation of multiple runs of an MCMC algorithm. The approach is restricted to dynamic stochastic block model graphs, and focuses on some dynamics of community only (i.e., birth, death, split, merge). Jiao et al. [64] study the common or coincident structure in the snapshots of a temporal network, based on the optimization of a function incorporating Markov steady-state matrices, similarity matrices and community membership matrices. However, the approach depends on the assumption that there are same nodes and fixed number of communities for each snapshot (resp. slice) of the temporal (resp. multiplex) network. Crawford and Milenkovic [22] propose to capture inter-snapshot relationships by grouping nodes based on regular equivalence (i.e., topological similarity), instead of structural equivalence.

Our proposed approach is related to the second of the aforementioned categories of temporal community detection; however, unlike methods in this category, it is designed to compute a dynamic consensus structure in a time-evolving network, therefore it is also partly related to the third category. Moreover, the proposed approach has the following key advantages: it does not require to match and/or track the evolution of communities over time, it does not depend on specific community-change events or restricted graph models, and it can handle variable size of node sets and community structures over time. Also, the consensus solution produced by our method can be updated incrementally with information about a newly observed snapshot of the network. More importantly, to the best of our knowledge, *our method is the first to bring the CMAB learning paradigm into the context of (consensus) community detection in temporal networks.*

4.3 Problem Statement

We are given a set $\mathcal{V}$ of *entities* (i.e., users) in a social environment, and a *temporal network* $\mathcal{G}$ as a series of graphs over discrete time steps $(G_1, G_2, \ldots, G_t, \ldots)$, where $G_t = \langle V_t, E_t \rangle$ is the graph at time t, with set of nodes V_t and set of undirected edges

E_t. We denote with $\mathcal{G}_{\leq t}$ a *series of graphs observed until time t*. Each node in V_t corresponds to a specific instance from the set $\mathcal{V}_t \subseteq \mathcal{V}$ of entities that occur at time t. The snapshot graphs can share different subsets of entities.

Given any G_t, we denote with $\mathcal{C}^{(t)}$ a *community structure* for G_t, which is a set of non-overlapping communities, and is assumed to be unrelated to any other $\mathcal{C}^{(t')}$ ($t' \neq t$), both in terms of number of communities and set of entities involved. We will use the term *dynamic ensemble* at time t, to refer to a set of community structures *incrementally* provided along with the snapshot graphs observed until time t, and we denote it as $\mathcal{E}_{\leq t} = \{\mathcal{C}^{(1)}, \ldots, \mathcal{C}^{(t)}\}$. We consider the following problem:

Problem 2 (Dynamic Consensus Community Detection (DCCD)) Given a temporal graph sequence $\mathcal{G}_{\leq t}$ and associated dynamic ensemble $\mathcal{E}_{\leq t}$, for any time $t \geq 1$ compute a community structure, called *dynamic consensus community structure* and denoted as $\mathcal{C}^*_{\leq t}$ such as to maximize:

$$R(T) = \sum_{t=1}^{T} Q_t(\mathcal{C}^*_{\leq t})$$

where T is the time horizon and Q_t is a chosen quality criterion for a community structure, over the history (before t) of the network (e.g. multilayer modularity [114]).

It is worth noticing that here the focus is only on cluster membership

Given $\mathcal{G}_{\leq t}$ and $\mathcal{E}_{\leq t}$, the representation model underlying the dynamic consensus being discovered over time is a matrix $\mathbf{M}$ we call ***dynamic co-association*** (or ***consensus***) ***matrix*** (DCM). The size of this matrix is initially $\mathcal{V}_t \times \mathcal{V}_t$ with $t = 1$, and at a generic time t is $|\mathcal{V}| \times |\mathcal{V}|$. The (i, j)-th entry of $\mathbf{M}$, denoted as m_{ij}, stores the probability of co-association for entities $v_i, v_j \in \mathcal{V}$, i.e., the probability that v_i and v_j are assigned to the same community, in the observed timespan.

Computing meaningful co-associations for the nodes in the temporal network and properly maintaining and updating the consensus community structure over time is challenging. On the one hand, we want to avoid (re)computation of the consensus from scratch, e.g., from a predetermined, finite set of community structures as in conventional consensus community detection [77, 114]; on the other hand, we also do not want to depend on any mechanism of tracking of the evolution of communities [23]. More importantly, *the dynamic consensus community structure should be able to embed long-term changes in the community formation as well as to capture short-term effects and newly observed community structures.*

Example 2 To support the importance of the above requirement, consider the example in Fig. 4.1. The color-filled node forms a community in its own for a long period (i.e., interval $[t, t']$), then at time $t'+1$ an event occurs to occasionally bind the node to the right-most community, finally a new event (starting at $t'+2$) causes the binding of the node to the left-most community for a new long period. Notably, existing methods for dynamic community detection, which are conceived to exploit information at the current and at the previous timestep only, will take community-assignment decisions that are mainly conditioned by short-term effects; for example, at any time within $[t'+2, t'']$, the node's community will be either of the two. By contrast, a community structure that aims to be of "consensus" by incorporating long-term effects as well, will more easily recognize the occasionality of the event at time $t'+1$ and may even decide to leave the node into its own singleton.

To address the above problem, we adopt a perspective that is different from the typical unsupervised learning approach to community detection problems. We argue

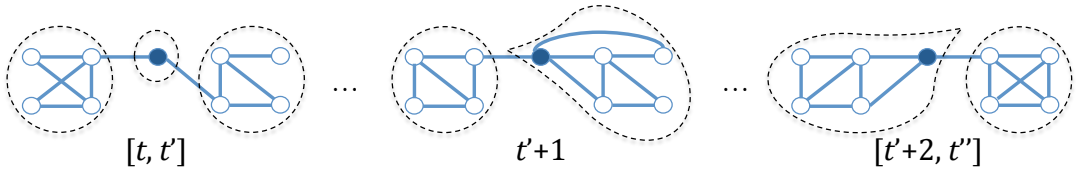


FIGURE 4.1: Three stages of the evolution of a network and its community structure (with $t \ll t' \ll t''$).

that the dynamic consensus community detection problem is well-suited to be solved under a *reinforcement learning* (RL) framework [112]. If we interpret the decisions to learn about the assignments of nodes to communities as *interrelated actions*, with *unknown* "rewards" ahead of time, then it emerges the need for learning which actions to take in order to maximize a "reward", which is related to how much benefit is gained by node assignments to communities. By learning from interactions, RL becomes particularly useful when there is uncertainty in the learning environment: this clearly holds in our setting due to the dynamics of the network, the evolution of its structural changes, and consequent effect on the community structure.

Moreover, actions taken affect not only the immediate reward, but also the next step in taking actions, and so the subsequent rewards. Thus, a further key aspect in our problem is the dilemma between "exploitation", i.e., making decisions that yield high current rewards, vs. "exploration", i.e., making decisions that sacrifice current gains with the prospect of better future rewards. *Multi-armed bandit* (MAB) refers to a class of stochastic resource allocation problems in the presence of alternative (competing) choices, that are paradigms of the exploration-exploitation trade-off. In this work, we focus on a particular class of MAB problems, called *combinatorial multi-armed bandit* (CMAB), whose distinguishing key is in the need for choosing *a set of* actions at any time.

4.3.1 Translating the problem of dynamic consensus community structure into CMAB

In our context, each pair of entities $\langle v_i, v_j \rangle$ in $\mathcal{G}_{\leq t}$ is hypothetically associated with an unknown distribution (with unknown mean μ_{ij}) for the probabilities of co-association over time, whose mean estimate is the entry m_{ij} in DCM. Each observation of a community structure of a snapshot network, can be considered as a sample from such distributions. Moreover, these may change their mean over time, thus our CMAB setting is non-stationary (cf. Sect. 2.3.1): in fact, for groups of entities which tend to maintain their membership to stable communities over time, we will observe a similar degree of co-association between pairs of entities belonging to the same, stable community; however, in general, the network structure along with the communities is subjected to several changes.

Each pair of entities $\langle v_i, v_j \rangle$ corresponds to a base arm, whose semantics is "*to assign v_i and v_j to the same community at a given time*". We will use symbol $c_i^{(t)}$ to denote the community of v_i at round t. A superarm A at round t is a set of arms, i.e., a set of pairs $\langle v_i, v_j \rangle$ such that $c_i^{(t)} = c_j^{(t)}$.

Playing a superarm A at each round t corresponds to a two-stage process: (i) inducing a community structure from the played superarm and (ii) performing stochastic relocation of nodes to neighbor communities. The stochastic nature of the process depends on both the random order with which we consider the node relocations and on the fact that, according to the optimization of a quality criterion, an improvement

due to relocation is accepted with a certain probability. Intuitively, this allows us to account for uncertainty in the long-term overall quality improvement of the consensus due to local relocations at a given time; for instance, it is unknown if the relation that explains two users share the same community at a given time could become meaningless in subsequent times.

After playing a superarm A, the rewards associated to the entity pairs (base arms) corresponding to the status of communities after the relocation phase, are revealed; these pairs include both the nodes that did not move from their community and the arms $\langle v_i, v_j \rangle$ triggered with the accepted relocations, i.e., such that node v_i was moved to the community of v_j. Furthermore, for the base arms that were neither selected nor triggered (i.e., pairs of nodes that were not in the same community before and after the relocation phase), we assume an implicit reward of zero that corresponds to the observation of the "no-coassociation" event. (This is in line with the possibility in CMAB of enabling the probabilistic triggering of *all* base arms.) The reward of a superarm corresponds to the *quality* of the community structure at the end of the relocation phase, which is a non-linear function of the base arms' rewards. More specifically, we resort to *modularity* as quality criterion for a community structure.

4.3.2 Relation between base arms and super arms

Let X_{ij}^t be the reward associated to the base arm corresponding to node pair $\langle v_i, v_j \rangle$ at time step t. The value of X_{ij}^t should be proportional to the "strength" of co-association between the node pair at the end of the relocation phase. The reward of the played superarm A (leading to the clustering $\mathcal{C}^*_{\leq t}$ after the stochastic relocation of nodes) can be defined in term of the base arms' rewards as follows:

$$R_t(A) = \frac{1}{d(\mathcal{V}_{[1..t]})} \sum_{c \in \mathcal{C}^*_{\leq t}} \sum_{\ell=1}^{t} \beta^{t-\ell} \left(d_\ell^{\text{int}}(c) - \frac{(d_\ell(c))^2}{d(\mathcal{V}_{[1..t]})} \right) =$$

$$= \frac{1}{d(\mathcal{V}_{[1..t]})} \sum_{i,j} \sum_{\ell=1}^{t} \beta^{t-\ell} \Big(A_{ij}^l - \frac{k_i^\ell k_j^\ell}{d(\mathcal{V}_{[1..t]})} \Big) \delta(X_{ij}^t)$$

where k_i^ℓ is the degree of v_i in the ℓ-th snapshot, A_{ij}^l is the (i,j)-th entry of the adjacency matrix of the ℓ-th snapshot graph, $d(\mathcal{V}_{[1..t]})$ is the total degree of the multiplex graph including snapshots from the first one to the t-th (i.e., $d(\mathcal{V}_{[1..t]}) = \sum_{\ell=1}^{t} \sum_{v \in V_\ell} d(v)$), $d_\ell(c)$ and $d_\ell^{\text{int}}(c)$ are the total degree and the internal degree of community c, respectively, measured w.r.t. edges of the ℓ-th snapshot network only and $\delta(X_{ij}^t) = 1$ if $X_{ij}^t > 0$, 0 otherwise. The stochastic nature of the above defined reward is determined by the random variables X_{ij}^t.

Moreover, as discussed in Sect. 4.4.2, we adopt a simplified version of the above reward equation that focuses on the latest ω snapshots of the networks.

4.4 The *CreDENCE* method

To solve the dynamic consensus community detection problem, we develop a CMAB-based method called CreDENCE – **C**MAB-based **D**ynamic Cons**EN**sus **C**ommunity D**E**tection, which is sketched in Algorithm 5.

Initially, the dynamic consensus matrix $\mathbf{M}$ is set as an identity matrix (Line 1), which reflects that no information has been processed yet, and hence each entity-node has co-association with itself only. At each round t, the algorithm chooses to

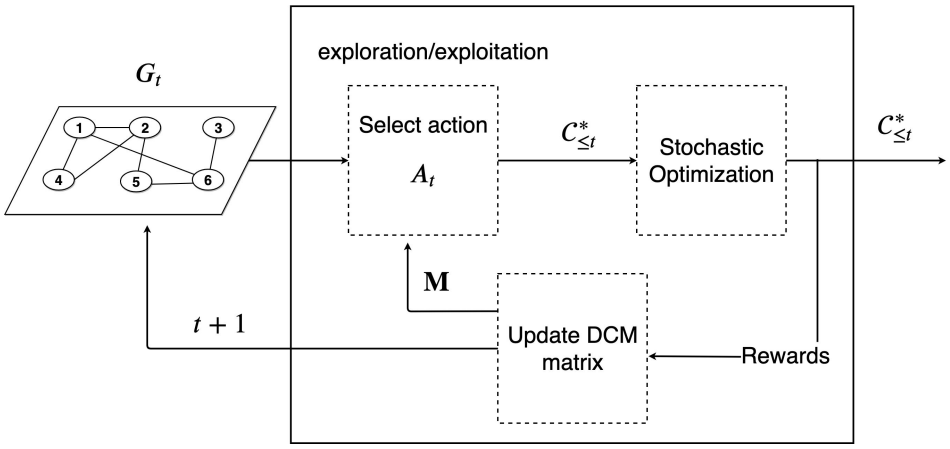


FIGURE 4.2: Overview of the CMAB-based Dynamic Consensus Community Detection method.

perform either exploration or exploitation, according to a given bandit strategy ($\mathcal{B}$). Intuitively, in the exploitation phase, we seed an oracle (i.e., a conventional method for community detection) with the mean estimates of co-association of the current DCM to infer the communities in the new snapshot graph observed at time t; by contrast, in the exploration phase, the new communities are identified using the t-th graph only. In either phase, the community structure generated at time t is finally used to produce a superarm that will correspond to the dynamic consensus community structure up to t ($\mathcal{C}^*_{\leq t}$).

Besides the involvement of a conventional community detection method $\mathcal{A}$ and a bandit strategy $\mathcal{B}$ to control the exploration-exploitation trade-off, we introduce a few parameters to ensure robustness in the algorithmic scheme of CreDENCE: (i) the learning rate α for the update of the mean estimates (i.e., m_{ij} entries), (ii) the relocation bias λ, and (iii) the temporal smoothness factor β and window size ω to control the amount of past knowledge for the step of node-relocations. Nonetheless, some of these parameters are interrelated, or reasonable values can be chosen as default.

Another remark on CreDENCE concerns its *incremental* nature: whenever a new step of evolution is observed, say at $T+1$, the last-update status of the DCM matrix along with $\mathcal{G}_{\leq T+1}$ will become the input for a further CMAB round. Figure 4.2 sketches an overview of CreDENCE.

4.4.1 Finding communities

At each round t, CreDENCE invokes a community detection method $\mathcal{A}$. This is just required to deal with (*static*) *simple graphs*. While in the exploration phase it directly applies to the snapshot graph G_t (Line 4), to handle the exploitation phase, the method should also be able to deal with *weighted* graphs: in this case, $\mathcal{A}$ is executed on the graph G_M(Line 7), which is built from the current DCM matrix in such a way that the edge weights in $G_M$correspond to the entries of $\mathbf{M}$ (Line 6). Next, from the obtained partitioning $\mathcal{C}_\mathbf{M}$ of G_M(Line 7), the knowledge about the community memberships of entity nodes in $\mathcal{C}_\mathbf{M}$ is used to infer a community structure $\mathcal{C}^{(t)}$ on the snapshot graph G_t (Line 8). Each community in $\mathcal{C}^{(t)}$ will have node set corresponding to exactly one community in $\mathcal{C}_\mathbf{M}$, and edge set consistent with the topology of G_t. Also, any entity v that newly appears in G_t (i.e., $v \in \mathcal{V}_t \wedge v \notin \mathcal{V}_{t'}, \forall t' < t$) and is disconnected will form a community in its own.

Algorithm 5 **C**MAB-based **D**ynamic Cons**EN**sus **C**ommunity D**E**tection (CreDENCE)

Input: Temporal graph sequence $\mathcal{G}_{\leq T}$ ($T \geq 1$), (static) community detection method $\mathcal{A}$, bandit strategy $\mathcal{B}$, learning rate $\alpha \in (0, 1)$, relocation bias $\lambda \in [0, 1]$, temporal smoothness $\beta \in (0, 1)$, temporal window width $\omega \geq 1$.
Output: Dynamic consensus community structure $\mathcal{C}^*_{\leq T}$.

1: $\mathbf{M} \leftarrow I_{|\mathcal{V}_1|\times|\mathcal{V}_1|}$
2: **for** $t = 1$ **to** T **do**
3: **if** $\mathcal{B}$ decides for EXPLORATION **then**
4: $\mathcal{C}^{(t)} \leftarrow$ *findCommunities*$(G_t, \mathcal{A})$
5: **else** {EXPLOITATION}
6: $G_M \leftarrow$ *buildDCMGraph*$(\mathbf{M})$
7: $\mathcal{C}_{\mathbf{M}} \leftarrow$ *partitionDCMGraph*$(G_M, \mathcal{A})$
8: $\mathcal{C}^{(t)} \leftarrow$ *inferCommunities*$(G_t, \mathcal{C}_{\mathbf{M}})$
9: **end if**
10: $\mathcal{C}^*_{\leq t} \leftarrow$ *project*$(\mathcal{C}^{(t)}, \mathcal{G}_{\leq t})$
11: $\mathcal{C}^*_{\leq t} \leftarrow$ *evalRelocations*$(\mathcal{G}_{\leq t}, \mathcal{C}^*_{\leq t}, \lambda, \beta, \omega)$ {*Using Eq.* (4.2)}
12: $\mathbf{M} \leftarrow$ *updateDCM*$(\mathbf{M}, \mathcal{C}^*_{\leq t}, \alpha)$ {*Using Eq.* (4.3)}
13: **end for**
14: **return** $\mathcal{C}^*_{\leq T}$

It should be noted that, although *any* method can in principle be used as $\mathcal{A}$, our preferred choice is towards efficient, modularity-optimization-based methods, such as [7]. This is motivated for consistency with our choice of using (multiplex) modularity as quality criterion in the (consensus) community structure refinement, as discussed in Sect. 4.4.2.

4.4.2 Generating the dynamic consensus community structure

The dynamic consensus community structure $\mathcal{C}^*_{\leq t}$, for each t, is generated in two steps. The first step (Line 10) corresponds to a simple projection of the community memberships from $\mathcal{C}^{(t)}$ onto $\mathcal{G}_{\leq t}$. The second step (Line 11) corresponds to *stochastic refinement* of the candidate $\mathcal{C}^*_{\leq t}$ obtained at the previous step. This stochastic refinement is performed through local search optimization, which is designed to relocate some nodes from their assigned community in $\mathcal{C}^*_{\leq t}$ to a neighboring one by acting greedily w.r.t. a quality criterion.

As previously anticipated, one appropriate choice refers to *modularity*. However, to account for the multiplexity of $\mathcal{G}_{\leq t}$ as well as the dynamic aspects, we revise the definition of modularity. In particular, we parameterize the temporal window by which the modularity context is set. The reason behind this choice is twofold: (i) to focus on a limited number of latest snapshots of the network, and (ii) to reduce the computational burden in the local search optimization.

Given $\mathcal{C}^*_{\leq t}$, temporal-window width ω and temporal smoothness factor β, we denote with $d(\mathcal{V}_{[t-\omega+1..t]})$ the total degree of the multiplex graph including snapshots from the $t-\omega+1$-th to the t-th and, for any community c, $d_\ell(c)$ and $d^{\text{int}}_\ell(c)$ are the total degree and the internal degree of c, respectively, measured w.r.t. edges of the ℓ-th

snapshot network only. We define the (ω, β)-*multiplex modularity* of $\mathcal{C}^*_{\leq t}$ as follows:

$$Q(\mathcal{C}^*_{\leq t}, \omega, \beta) \quad = \quad \frac{1}{d(\mathcal{V}_{[t-\omega+1..t]})} \sum_{c \in \mathcal{C}^*_{\leq t}} \sum_{\ell=t}^{t-\omega+1} \beta^{t-\ell} \left(d_\ell^{\text{int}}(c) - \frac{(d_\ell(c))^2}{d(\mathcal{V}_{[t-\omega+1..t]})} \right) \quad (4.1)$$

As previously mentioned, the meaning of β is to smooth the contribution of earlier snapshots in the computation of the quality of the dynamic consensus, i.e., lower values of β will penalize older snapshots. It is worth noting that β may take a role that is opposite to that of the learning rate α in Algorithm 5. Therefore, by default, we set $\beta = 1 - \alpha$.

The local search optimization, at any time t, evaluates the possible improvement in terms of modularity due to the relocation of nodes v_i that lay on the boundary of their assigned communities towards one of the communities that at time t contain nodes linked to v_i. By denoting with c_i the initial community of a boundary node v_i, and simplifying the modularity notation with function symbol Q, the best *modularity variation*, denoted as ΔQ_i, corresponding to moving v_i to a neighbor community is as follows:

$$\Delta Q_i = Q(c_i \setminus \{v_i\}) - Q(c_i) + \max_{c_j \in NC_i^{(t)}} (Q(c_j \cup \{v_i\}) - Q(c_j)) \quad (4.2)$$

where $NC_i^{(t)}$ denotes the set of neighbor communities for node v_i at time t. If $\Delta Q_i > 0$, then there is a single chance to accept the relocation of v_i to c_j with probability $1 - \lambda e^{-\lambda \Delta Q_i}$, where $\lambda \in [0, 1]$ is a smoothing coefficient to control the bias towards relocations. Intuitively, this allows us to account for uncertainty in the long-term overall quality improvement of the consensus due to local relocations at a given time; for instance, it is unknown if the relation that explains two users share the same community at a given time could become meaningless in subsequent times.

4.4.3 Updating the dynamic consensus

The DCM-update scheme in Algorithm 5 (Lines 12) follows a standard principle in reinforcement learning, whereby as the agent explores further, it is capable of updating its current estimate according to a general scheme of the form $newEstimate \leftarrow oldEstimate + \alpha(target - oldEstimate)$, which intuitively consists in moving the current estimate in the direction of a "target" value, with slope α. In our setting, we want to control the update of co-associations by subtracting a quantity α of resource from the co-associations of each node, at time t, and redistributing this quantity among the nodes in $c_i^{(t)}$, for each v_i. This redistribution corresponds to the *reward* of a single co-association, i.e., given v_i, the reward of assigning any v_j to the same community of v_i. Upon this, given $\alpha \in [0, 1]$ and any (i, j)-th entry of $\mathbf{M}$, we define the *update* equation as:

$$m_{ij}^{(t+1)} = m_{ij}^{(t)} + \alpha \left(\frac{1}{|c_i^{(t)}|} [v_j \in c_i^{(t)}] - m_{ij}^{(t)} \right) = \frac{\alpha}{|c_i^{(t)}|} [v_j \in c_i^{(t)}] + (1 - \alpha) m_{ij}^{(t)} \quad (4.3)$$

where $[x \in X]$ denotes the Iverson-bracket notation for the indicator function.

Properties of the update equation. It should be noted that the reward $1/|c_i^{(t)}|$ produces the effect of making it stronger the co-association between nodes belonging to smaller communities. This is consistent with a major finding in a recent study proposed in [61] whereby co-memberships of nodes in larger communities are statistically

less significant (than in smaller ones), because members in such communities have limited influence upon each other in the network. A further reason to favor co-associations in smaller communities is to compensate for a typical bias relating to a tendency of producing large communities (e.g., resolution limit in modularity-optimization based methods). Another important property of Eq. (4.3) is the *exponential smoothing of earlier actions*, with constant α [112], i.e., the update scheme leads to weight recently obtained rewards more heavily than earlier ones, and the reward of a past co-association between two nodes decreases exponentially in time.

Proposition 1 (*Exponential smoothing of earlier actions*) *The update rule in Eq. 4.3 ensures that the rewards of past co-association between any two nodes v_i, v_j decreases by a factor $(1-\alpha)^{t-s}$, with $s \leq t$.*

Proof. Let us assume that nodes v_i, v_j are assigned to the same community and remain therein over time. In this scenario, by repeated substitutions we derive that:

$$\begin{aligned}
m_{ij}^{(t+1)} &= \frac{\alpha}{|c_i^{(t)}|} + (1-\alpha)m_{ij}^{(t)} = \\
&= \frac{\alpha}{|c_i^{(t)}|} + \Big(1-\alpha\Big)\left[\frac{\alpha}{|c_i^{(t-1)}|} + \Big(1-\alpha\Big)m_{ij}^{(t-1)}\right] = \\
&= \frac{\alpha}{|c_i^{(t)}|} + \frac{(1-\alpha)\alpha}{|c_i^{(t-1)}|} + (1-\alpha)^2 m_{ij}^{(t-1)} = \\
&= \frac{\alpha}{|c_i^{(t)}|} + \frac{(1-\alpha)\alpha}{|c_i^{(t-1)}|} + \ldots + \frac{(1-\alpha)^{t-1}\alpha}{|c_i^{(1)}|} + (1-\alpha)^t m_{ij}^{(1)} = \\
&= (1-\alpha)^t m_{ij}^{(1)} + \sum_{s=1}^{t}(1-\alpha)^{t-s}\frac{\alpha}{|c_i^{(s)}|}
\end{aligned}$$

Above, the last term offers evidence that the weight associated to rewards of past co-associations decreases exponentially over time by a factor $(1-\alpha)^{t-s}$.

Proposition 2 (*Stochasticity of the dynamic co-association matrix*) *The update rule in Eq. 4.3 ensures that $\boldsymbol{M}$ is a right stochastic matrix.*

Proof. Let us consider the i-th row of $\mathbf{M}$, updated by means Eq. (3) at time step t. By summing all entries in the row, we observe that:

$$\begin{aligned}
&\sum_{v_j \in c_i^{(t)}} \left(\frac{\alpha}{|c_i^{(t)}|} + (1-\alpha)m_{ij}^{(t)}\right) + \sum_{v_j \notin c_i^{(t)}} (1-\alpha)m_{ij}^{(t)} = \\
&\sum_{v_j \in c_i^{(t)}} \frac{\alpha}{|c_i^{(t)}|} + \sum_{v_j \in c_i^{(t)}} (1-\alpha)m_{ij}^{(t)} + \sum_{v_j \notin c_i^{(t)}} (1-\alpha)m_{ij}^{(t)} =
\end{aligned}$$

$$\sum_{v_j \in c_i^{(t)}} \frac{\alpha}{|c_i^{(t)}|} + \Big(1-\alpha\Big) \underbrace{\sum_{v_j \in \mathcal{V}} m_{ij}^{(t)}}_{1} = \cancel{|c_i^{(t)}|}\frac{\alpha}{\cancel{|c_i^{(t)}|}} + 1 - \alpha = 1$$

4.5 Computational complexity aspects

The time complexity of the basic version of CreDENCE is determined by the update operations on $\mathbf{M}$ given by Eq. (3) and by the community detection step. The latter, i.e., identifying communities in the t-th snapshot network, or in the graph $G_M$induced by $\mathbf{M}$, can actually be solved linearly in the number of entities, when an appropriate community detection is used [117]. As concerns the update step, we first observe that the relocation of nodes can be executed in $O(|V_t| + \omega|E_t|) = O(|\mathcal{V}| + \omega|E_t|)$, since for each node we look at its neighbor communities, which are bounded by the degree of the node. Evaluating the modularity improvement (Eq. (2)) is $O(\omega)$, provided that ω indexes are maintained to store the degree of communities for each of the last ω time steps, and to store the number of links of v with nodes in community c at time t, for each node v, time t and community c. Therefore, since we constraint the number of relocation trials to be of the order of the number of nodes, the overall time cost of relocation of nodes is $O(|\mathcal{V}| + \omega|E_t|)$. However, the update of $\mathbf{M}$ involves a number of entries that is at least equal to $\sum_{c_i \in \mathcal{C}^{(t)}} |c_i|^2$. This could lead to a cost that becomes quadratic in the number of entities as soon as some of the communities have size of the order of $\mathcal{V}$. Moreover, the spatial complexity of CreDENCE is determined by number of non-zero entries of $\mathbf{M}$, which again could be quadratic in the number of entities, due to the denseness of the matrix.

4.5.1 Speeding up *CreDENCE*

Maintaining and updating the DCM matrix represents a computational bottleneck of CreDENCE, which may lead to a quadratic cost in the number of entities in case some of the communities had size of the order of $\mathcal{V}$. By definition, $\mathbf{M}$ can easily become dense, yet noisy, since many co-associations may be weak (e.g., outdated co-associations), thus corresponding to poorly significant consensus memberships.

One way to alleviate this issue is to prune the matrix by zeroing those entries that are below a predefined threshold; in practice, this will unlikely be enough to solve the issue. Rather, we notice that it is more appropriate to introduce a constraint of linkage between nodes when evaluating Eq. (4.3): this is not only consistent with the requirement of having as high density as possible within a (consensus) community (as studied in [114]), but it will also impact on making $\mathbf{M}$ sparser. However, one drawback would be the loss of symmetry in $\mathbf{M}$.

We hence propose a modification to the update equation that both integrates the linkage constraint and preserves the stochasticity property of the matrix:

$$m_{ij}^{(t+1)} = \frac{\alpha}{|c_i^{(t)} \cap N_i^{(t)}|}[v_j \in c_i^{(t)} \cap N_i^{(t)}] + (1-\alpha)m_{ij}^{(t)}, \tag{4.4}$$

where $N_i^{(t)}$ denotes the set of neighbors of v_i in G_t. The entry m_{ij} now is meant to store the strength of co-association of v_i conditionally to the topological link with v_j. Moreover, the graph representation of $\mathbf{M}$ becomes directed: to keep the scheme presented in Algorithm 5, we simply modify the definition of the consensus graph G_M so that the weight of an edge (v_i, v_j) is set as $\max\{m_{ij}, m_{ji}\}$. This allows us to preserve the importance of a co-association between any two entities when finding a community structure in G_M.

We incorporate the above modifications into Algorithm 5 to obtain an enhanced, efficient version of CreDENCE. It can be noticed that the time complexity of CreDENCE now becomes $O(T \times (|\mathcal{V}| + |E_{\leq T}|))$, while the spatial cost is determined by the size of $\mathbf{M}$, i.e., $O(|E_{\leq T}|)$, with $E_{\leq T} = \bigcup_{t=1}^{T} E_t$.

4.6 Evaluation methodology

Real-world networks. *Epinions* [87] is the trust and distrust network of Epinions, an online product rating site. For this network we discarded the orientation of edges.

Facebook [122] contains friendship data of Facebook users, where each undirected edge expresses the birth of a friendship between two users in the online platform. The dataset refers to the period from September 2006 to January 2009, with some edges having unknown timestamp. Because these edges are the majority in the dataset, we decided to not discard them and associated them the smallest timestamp (i.e. August 2006). We firstly aggregated the edges by month but, because each edge appeared only in one snapshot, we built a cumulative version of the network which consists in projecting each edge from the month in which it appeared to all next ones.

Wiki-Conflict [12] refers to conflicts between users of the English Wikipedia, as users involved in a edit-war of a page. Each edge represents a conflict between two users, with the edge sign representing positive and negative interactions. An example for a negative interaction would be when one user revert the edit of another user. In our context, we removed the sign of edges and considered only the related timestamps aggregated by month, joining the first 6 snapshots (from 8-2001 to 1-2002) and the 5 snapshots from 3-2002 to 7-2002 because they contained a few edges. We also removed self-loops, which occasionally occurred in some snapshots.

Wiki-Elections [79] is the network of users from the English Wikipedia that voted for and against each other in admin elections. We did not consider orientation and sign of edges. We split the networks by month, aggregated the first 4 snapshots (from 3-2004 to 6-2004) that contains very few links, and finally removed the edges with timestamp after 2050 which were included in the original version of the network [79].

YouTube [90] refers to the friendship data of the social network of YouTube. For this dataset, as in Facebook, we built a cumulative network for our evaluation.

Table 4.1 reports statistics for each evaluation network. Note that with terms 'static', 'hapax', and 'dynamic' we mean nodes/edges that are present in all snapshots, present in only one snapshot, and present in multiple, not necessarily contiguous snapshots, respectively. Also, symbols e_t^+ and e_t^- refer to the fraction of new edges and disappeared edges, respectively, when transitioning from the t-1-th to the t-th snapshot; analogously for nodes corresponding to symbols v_t^+, v_t^-. Note that, while the friendship-based networks (i.e., Facebook and YouTube) evolve very smoothly, the other selected networks undergo to drastic changes in terms of disappearing/appearing edges and nodes.

Synthetic networks. We also used synthetic networks generated through *RDyn* [105], which is designed to handle community dynamics and change events (merge/split). Starting from an initial community structure, it simulates some edge events to change the community structure over time. RDyn adopts the notion of stable iteration to mimic *ground-truth* communities; in particular, when a community structure reaches a minimum quality (i.e., conductance), then it is recognized as ground-truth. We believe that the latter property of RDyn is important since it fills a lack in the literature about the unavailablilty of ground-truth data for (large) time-evolving multilayer networks.

When using RDyn, almost all parameters related to the dynamics of the network were set with their default values. Moreover, because the generator outputs the ground-truth communities only for the stable iterations (which are fewer than the number of specified snapshots), we adopted the communities at the previous stable iteration as ground-truth community structure.

TABLE 4.1: Main characteristics (after preprocessing) of our evaluation data. Mean ± standard deviation values refer to all snapshots in a network.

	#entities ($\lvert\mathcal{V}\rvert$)	#edges	#time steps	node set coverage	edge semantics	% static (nodes, edges)	% hapax (nodes, edges)	% dynamic (nodes, edges)
Epinions	131 828	727 344	32	0.05	trust/distrust	(0.1, 0)	(80.8, 95.6)	(19, 2.2)
Facebook	63 731	17 676 817	30	0.87	friendship birth	(82.9, 2.7)	(0.2, 0)	(16.9, 1.9)
Wiki-Conflict	118 100	2 272 276	82	0.05	wikipage editing	(0, 0)	(60.1, 83.4)	(38.9, 5.8)
Wiki-Election	7 118	102 906	44	0.08	vote assignment	(0, 0)	(49.7, 95.7)	(50.3, 2.2)
YouTube	3 223 589	41 955 741	8	0.62	friendship birth	(33.4, 6.7)	(12.4, 4)	(54.2, 11.6)

	network evolution rate			
	$e_t^+ = \frac{\lvert E_t \setminus E_{t-1}\rvert}{\lvert E_t\rvert}$	$e_t^- = \frac{\lvert E_{t-1} \setminus E_t\rvert}{\lvert E_{t-1}\rvert}$	$v_t^+ = \frac{\lvert V_t \setminus V_{t-1}\rvert}{\lvert V_t\rvert}$	$v_t^- = \frac{\lvert V_{t-1} \setminus V_t\rvert}{\lvert V_{t-1}\rvert}$
Epinions	0.97 ± 0.007	0.98 ± 0.008	0.65 ± 0.08	0.69 ± 0.06
Facebook	0.02 ± 0.01	0	0.006 ± 0.006	0
Wiki-Conflict	0.95 ± 0.02	0.95 ± 0.02	0.52 ± 0.1	0.51 ± 0.12
Wiki-Election	0.99 ± 0.004	0.99 ± 0.005	0.5 ± 0.07	0.49 ± 0.08
YouTube	0.16 ± 0.06	0	0.14 ± 0.06	0

Competing methods. We conducted a comparative evaluation of CreDENCE with the following three methods.

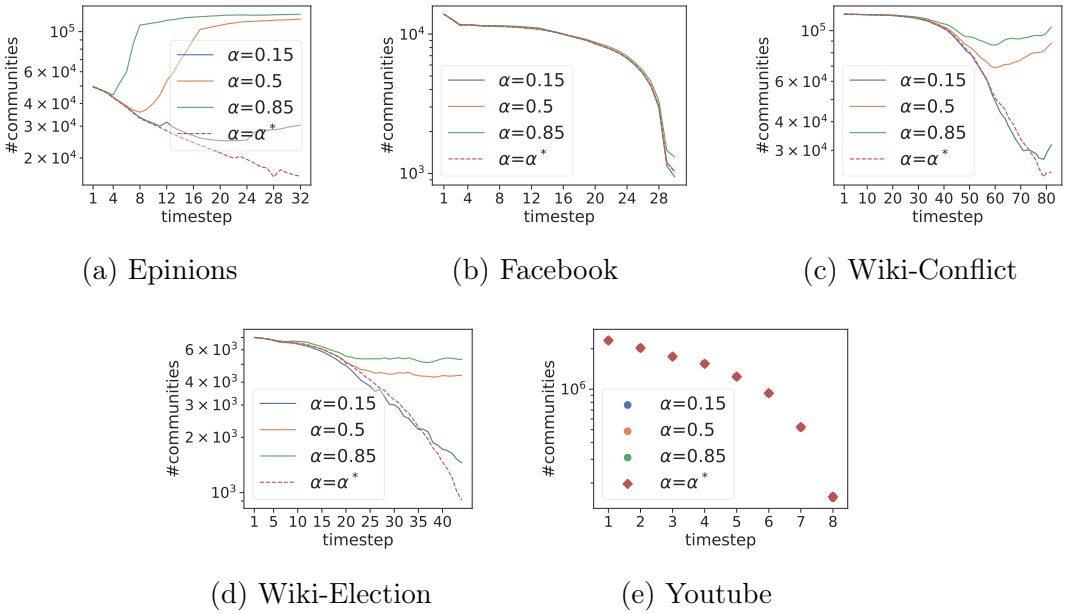
- DynLouvain [60]: it applies Louvain method [7] to a condensed network based on the topology of the snapshot at current time t and community structure at time t-1.
- EvoAutoLeaders [42]: this is an evolutionary method based on a notion of community as a set of follower nodes congregating close to a potential leader (i.e., the most central node in the community).
- M-EMCD* [86]: this is a parameter-free enhanced version of the consensus-based method in [114], which filters noisy co-associations via marginal likelihood filter and optimize the multilayer modularity of the consensus w.r.t. a static ensemble of community structures.

Our choice of competitors relies on the following: (i) the methods are representative of the second category (DynLouvain and EvoAutoLeaders) and third category (M-EMCD*) of dynamic community detection approaches (cf. Sect. 4.2), to which CreDENCE is close too; also like CreDENCE, (ii) they are based on modularity optimization and (iii) do not require an input number of communities.

Evaluation settings. We varied the learning rate α in $\{0.15, 0.5, 0.85\} \cup \{\alpha^*\}$, where α^* is an *adaptive* learning rate set to the fraction of times a base arms is used, and the temporal-window width ω from 2 to 10; however, unless otherwise specified, we used the setting $\omega = 2$, $\beta = 1 - \alpha$ to emphasize the importance of few, more recent snapshots. We set the relocation bias λ to 0, i.e., a relocation is accepted if it leads to an improvement in modularity (Eq. (4.2)).

To detect communities from each snapshot (i.e., $\mathcal{A}$ in Algorithm 5), we used the classic Louvain method [7]. This choice is not only consistent with our modularity-optimization-based relocation phase, but also with the choice of static algorithm in most approaches for dynamic community detection [23].

As for the bandit strategy $\mathcal{B}$, we resorted to ϵ-*greedy*, i.e., with a small probability ϵ we take an exploration step, otherwise (i.e., with probability $1 - \epsilon$) an exploitation step. We set $\epsilon = 0.1$, which revealed to lead to a suitable trade-off for our networks, which have different evolution rates.



(a) Epinions (b) Facebook (c) Wiki-Conflict

(d) Wiki-Election (e) Youtube

FIGURE 4.3: Size of the dynamic consensus community structures (singleton disconnected communities included)

4.7 Results

We present our main results, whereby performance scores obtained by CreDENCE correspond to the *average over 100 runs*, in order to avoid sensitivity issues due to the randomness in the interleaving of exploration-exploitation phases.

4.7.1 Impact of learning rate

Consensus size. Figure 4.3 shows the consensus size over time, for different α. One premise here is that the static community detection method applied on each snaphot (cf. Sect. 4.6) produced a huge number of communities, which is clearly expected considering the characteristics of our real-world networks; this clearly impacted on the consensus size as well. Looking at the plots, we observe that the number of detected consensus communities generally increases for higher values of α, because this more quickly leads to lose memory of past co-associations, thus causing proliferation of communities in the consensus solution. Moreover, on the networks having high rate of structural change (in terms of both node and edge sets), the trends for the various settings of α tend to deviate in correspondence of the time steps associated with most change events; by contrast, in the friendship-based networks (i.e., Facebook and YouTube), which are characterized by a much smoother evolution, the temporal trends of consensus size are similar w.r.t. the various α.

Multilayer modularity. Figure 4.4 shows *multilayer modularity* [114] results by varying α. Lower values of α generally lead to higher modularity except for Facebook and YouTube networks. This is explained since, in the networks having high rate of structural change, a lower learning rate helps remember past co-associations, thus information about older snapshots. Moreover, in such networks we observe a decreasing trend in modularity since the consensus must embed an increasing number of snapshots, each very different from the others (cf. Table 4.1). By contrast, for

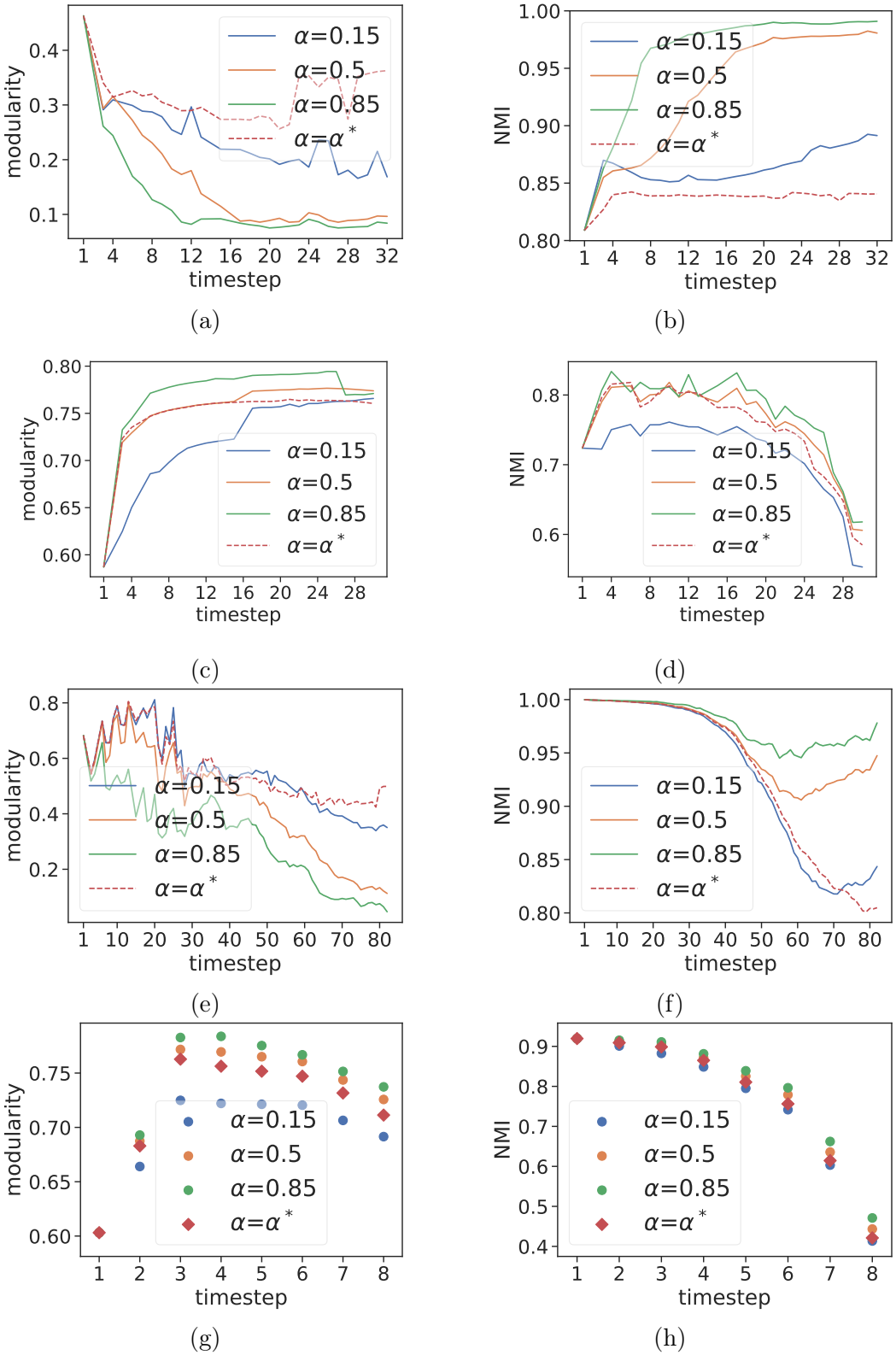
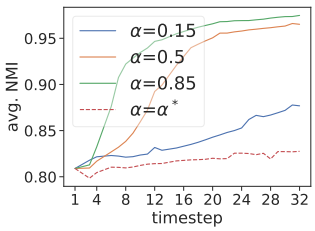
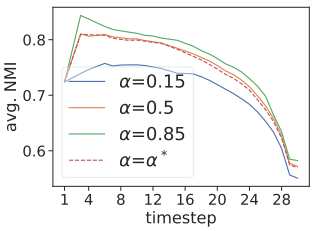


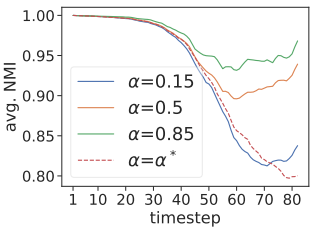
FIGURE 4.4: Multilayer modularity of the CreDENCE solutions (left-most plots) and NMI between the CreDENCE consensus community structure and the snapshot's community structure, at each t (right-most plots): (a)-(b) Epinions, (c)-(d) Facebook, (e)-(f) Wiki-Conflict, (g)-(h) YouTube.



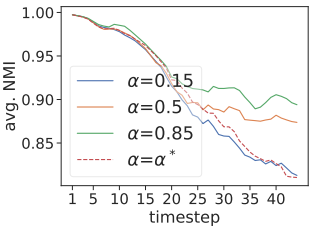
(a) Epinions



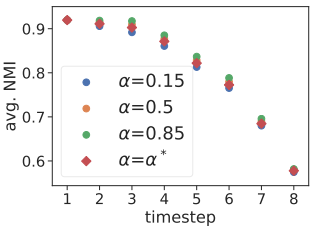
(b) Facebook



(c) Wiki-Conflict

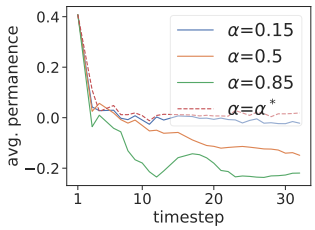


(d) Wiki-Election

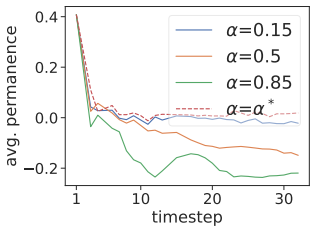


(e) Youtube

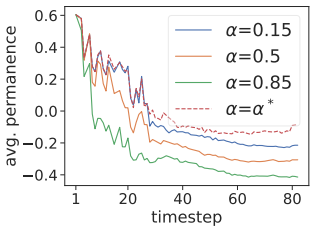
FIGURE 4.5: Average cumulative NMI between the dynamic consensus community structures



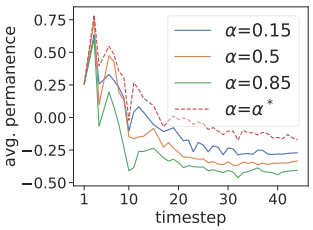
(a) Epinions



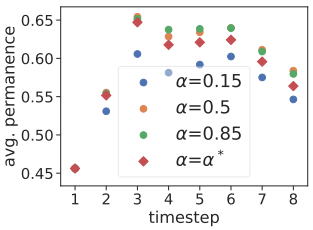
(b) Facebook



(c) Wiki-Conflict



(d) Wiki-Election



(e) Youtube

FIGURE 4.6: Average cumulative permanence of the dynamic consensus community structures

Facebook and YouTube, a high learning rate reveals to be beneficial to discovering consensus communities with higher modularity.

NMI. We measured the *NMI* [111] between the dynamic consensus and the community structure of snapshot, for each time step (Fig. 4.4). As expected, the two structures are more similar (i.e., higher NMI values) as α increases, which implies weighting more the current snapshot in the consensus generation. Analogous remarks were drawn for the *average cumulative NMI*, which is computed at each t by averaging the NMI between the dynamic consensus at t and the community structures over all snapshots at any time $t' \leq t$ (Figure 4.5).

Permanence. Moreover, we include a further validation criterion, namely *permanence* [18]. Like modularity, permanence is an internal validation criterion, however, as opposed to modularity, it is based on local node-centric properties (i.e., node's connectedness, node's cohesiveness, node's external pull), such as the permanence of a node quantifies its propensity to remain in its assigned community and the extent to which it is pulled by the neighboring communities. Global permanence will be closer to 1 as more vertices have high permanence, that is more vertices are in well-defined communities. Like for NMI, we will present *average cumulative permanence* results, i.e., we computed the global permanence of the solution produced at time t over the network snapshots at times $t' \leq t$ and averaged to get the final permanence values.

Figure 4.6 shows the impact of the learning rate on the average global permanence of the computed dynamic consensus community structures. As for modularity results, lower values of α (and remarkably with the adaptive setting α^*) generally lead to higher average permanence except for Facebook and YouTube networks. This is explained since, in the networks having high rate of structural change, a lower constant learning rate helps remember past co-associations, thus information about older snapshots. Moreover, in such networks we observe a decreasing trend in average permanence since the consensus must embed an increasing number of snapshots, each very different from the others (cf. Table 4.1). By contrast, for Facebook and YouTube, a high learning rate reveals to be beneficial to discovering consensus communities with higher average permanence.

4.7.2 Impact of temporal-window width

To understand the effect of varying ω on the CreDENCE behavior, we focused on combining various settings of this parameter with the configuration $\alpha = 0.15$, in order to ensure keeping more information about past snapshots in the dynamic consensus while looking backward for the refinement phase (results shown in Figure 4.7).

One general remark is that, regardless of the trend specific for each measure, higher values of ω actually lead to better multilayer modularity (this is expected since higher ω enables the relocation phase to optimize a quality criterion that is closer to the measured multilayer-modularity); nonetheless, modularity improvements are found to be negligible as $\omega > 4$. Even better, no evident differences are instead observed in terms of NMI and number of consensus communities. This indicates robustness of CreDENCE with variation of the ω parameter.

4.7.3 Efficiency evaluation

To assess the scalability of CreDENCE, we used *RDyn* [105] to generate different synthetic networks by varying the number of snapshots and community events.[1]

[1]Experiments were carried out on a Linux (Mint 18) machine with 2.6 GHz Intel Core i7-4720HQ processor and 16GB ram

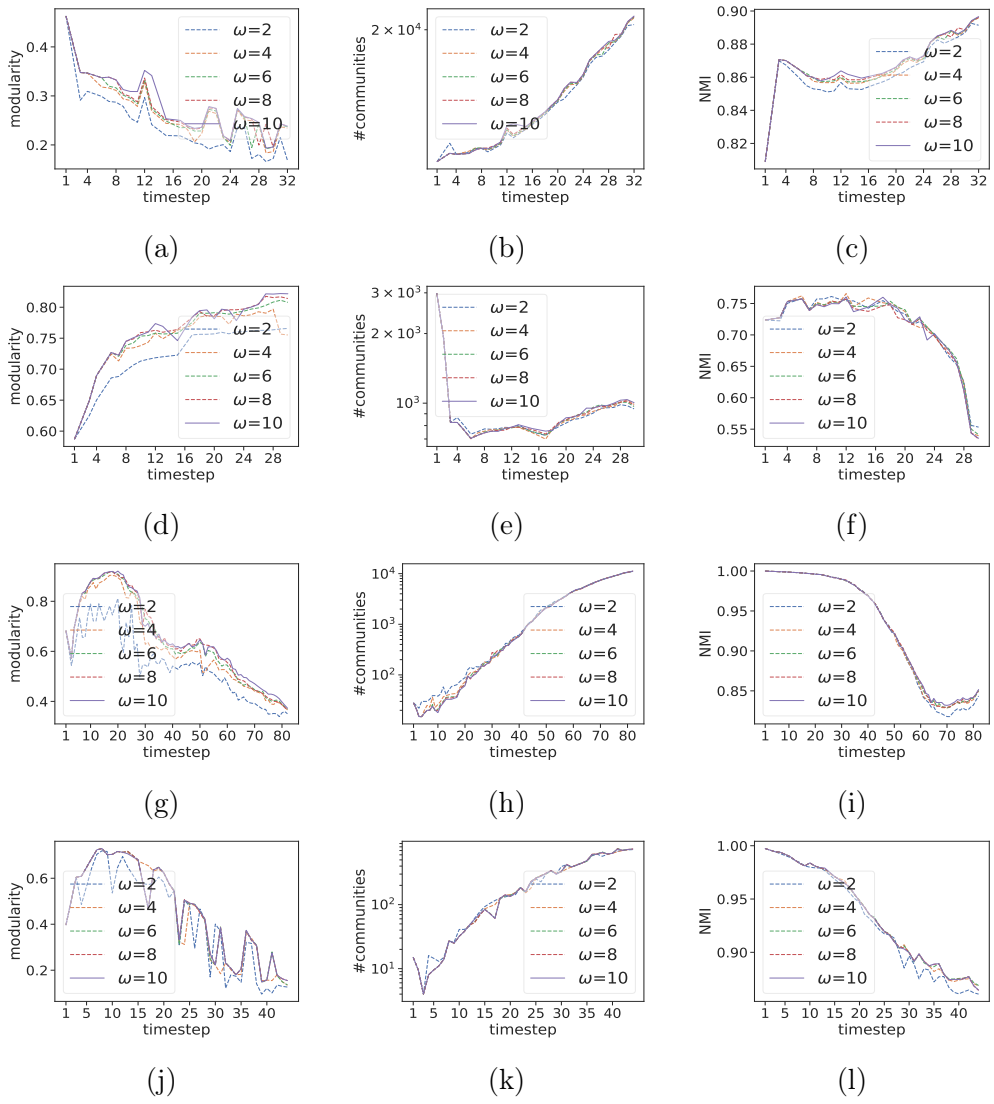


FIGURE 4.7: Performance by varying temporal-window width (ω): (a)-(b)-(c) Epinions, (d)-(e)-(f) Facebook, (g)-(h)-(i) Wiki-Conflict, (j)-(k)-(l) Wiki-Election.

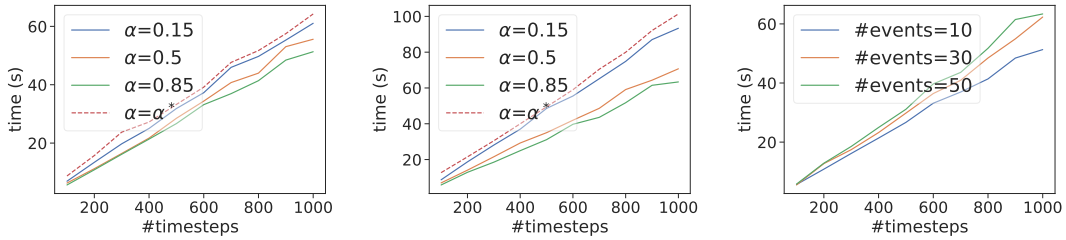


FIGURE 4.8: Time performance on RDyn synthetic networks

The plots in Fig. 4.8(a)–(b) report the execution times for different settings of α, over a temporal network with 1K entities, 1K time steps, and change rate of 10, resp. 50, community events. In all cases, CreDENCE scales linearly with the number of considered timesteps, which is consistent with our complexity analysis (cf. Sect. 4.5.1). We also observe that the execution time is generally higher for the adaptive learning

TABLE 4.2: Increment percentages of CreDENCE w.r.t. DynLouvain and M-EMCD*. Values correspond to the increment percentages averaged over all snapshots in a network, using the average best-performing α.

	DynLouvain		M-EMCD*	
	Modularity	NMI	Modularity	NMI
Epinions	1789.0 %	-2.2 %	13.9 %	37.6%
Facebook	3.5 %	9.4 %	60.0 %	37.5 %
Wiki-Conflict	> 1.0 E+05 %	-1.8 %	-6.8 %	37.6 %
Wiki-Election	660.5 %	-2.1 %	32.0 %	58.5 %
YouTube	-0.1 %	8.4 %	21.1 %	11.6 %
RDyn	2.0 %	24.97 %	103.22 %	81.1 %

rate α^* as well as for lower values of α (i.e., as the past co-associations are preserved longer), thus making the DCM matrix denser and more costly to process. Figure 4.8(c) shows the execution times of our method with $\alpha = 0.85$, on three different synthetic networks all of 1K entities, 1K snapshots, and with 10, 30, 50 community events, respectively. As expected, the higher the evolution rate, the higher the execution time; nonetheless, CreDENCE shows again to scale linearly with the size of the network.

4.7.4 Comparison with competing methods

Table 4.2 and Fig. 4.9 compare CreDENCE with the other methods.Concerning modularity results, our method outperforms both DynLouvain and M-EMCD*, where performance gains vs. the former (resp. latter) are outstanding for networks with high (resp. low) rate of structural change. NMI by CreDENCE is always significantly higher than the competitors' ones, especially against M-EMCD*; one exception is represented by a gap of just 2% w.r.t. DynLouvain for three networks with high evolution rate. The latter fact can be explained since DynLouvain only accounts for the most recent two snaphots thus it is able to better capture the community structure of the current snapshots (against which NMI is computed) also for networks with high evolution rate. On the contrary, CreDENCE accounts in the consensus generation also for long-term communities information which decreases, especially in networks with high evolution rate, its possibility to generate solutions which are similar to the community structure of the snapshots. Moreover, we emphasize that CreDENCE also outperforms the evolutionary EvoAutoLeaders, as long as the competitor results were available—indeed, it incurred in processing-time issues (tens hours) in all networks but the smallest ones, i.e., Wiki-Election and RDyn.

4.8 Chapter review

In this chapter, we proposed CreDENCE, a CMAB-based method for the problem of dynamic consensus community detection in temporal networks. Experimental evidence on real and synthetic networks has shown the meaningfulness of the consensus solutions produced by CreDENCE, also revealing its unique ability of dealing with temporal networks that can have different evolution rate.

We plan to further investigate on the impact of different bandit strategies (e.g., UCB, Thompson sampling), and on learning our model parameters to best fit the community structure and evolution in a given temporal network.

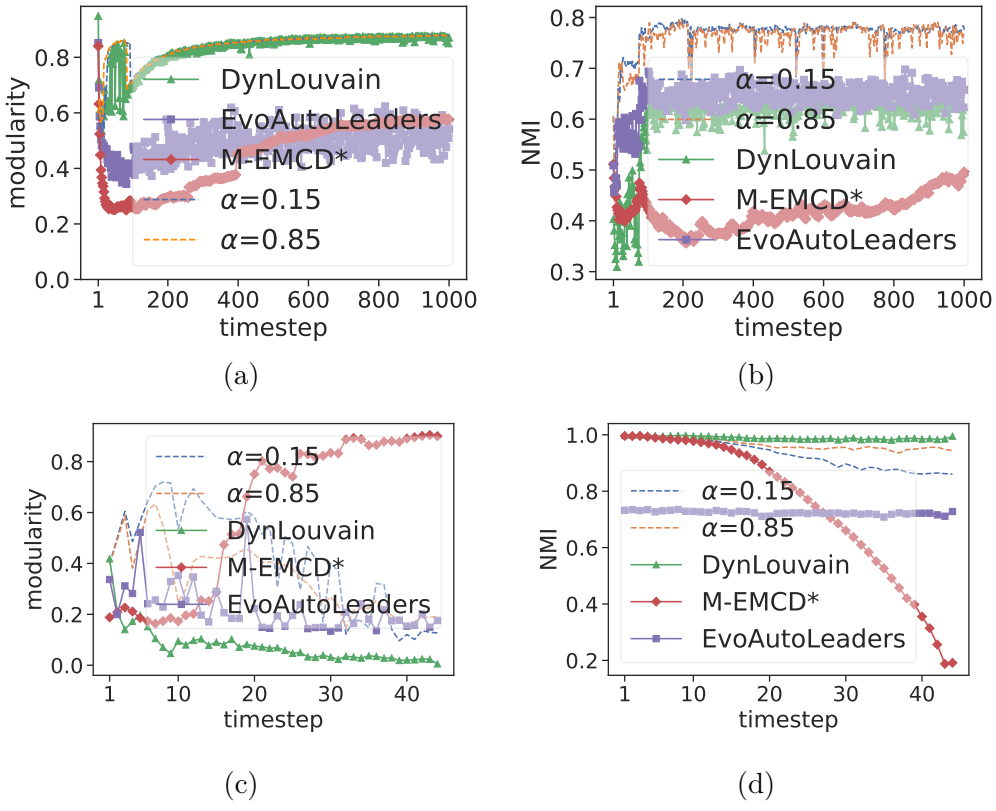


FIGURE 4.9: Competing methods vs. CreDENCE on (a)-(b) RDyn network and (b)-(c) Wiki-Election.

Chapter 5

Trust Network Inference.

Summary. Trust inference is essential in a plethora of data mining and machine learning applications. Unfortunately, conventional approaches to trust inference assume trust networks are available, while in practice they must be derived from social network features. This is however a difficult task which has to cope with challenges relating to scarcity, redundancy and noise in the available user interactions and other social network features. In this work, we introduce the new problem of Trust Network Inference (TNI), that is, inferring a trust network from a sequence of timestamped interaction networks. To solve the TNI problem, we propose a principled approach based on a preference learning paradigm, under a preference-based racing formulation. The proposed approach is suitable for addressing the above challenges, moreover it is versatile (i.e., independent from the social network platform) and flexible w.r.t. the use of topological and content-based information. Extensive experimental evaluation focusing on two distinct ground-truth scenarios, has provided evidence of the meaningfulness and uniqueness of our TNI approach, which can be regarded as key-enabling for any application that requires to handle a trust network associated with a social environment.

5.1 Introduction

The term trust-based social network, or simply *trust network*, commonly refers to a graph of entities (i.e., individuals) that are linked through asymmetric relationships that correspond to subjective trust statements. Given a trust network, *trust inference* is the task of predicting a new relation between two nodes, so that the locally inferred trust score can be regarded as a personalized opinion of one user (trustor) with respect to another user (trustee). Trust inference is an essential task in many data analysis and machine learning applications, from social influence propagation and opinion spreading to recommender systems and privacy preserving, whose impact extends also to peer-to-peer networks and mobile ad-hoc networks [109].

Challenges in trust inference. The conventional approach to trust inference is to compute the trust between any two non-adjacent nodes in a trust network by considering the different paths from one node to the other, as well as strategies for trust propagation and for aggregating the propagated trust values through different paths [109, 126]. Unfortunately, all existing trust-inference approaches rely on the assumption that a trust network has been already formed, while in reality *trust networks are not naturally available*. Rather, trust relations must first be determined from the available information in a social environment, e.g., the history of users' activities and their interactions. The latter can be modeled through a temporal graph where each snapshot graph describes users' interactions at a particulare time. However, computing trust relationships by considering interactions as a proxy for trust may represent a risk in some practical cases, e.g. in disassortative graphs where nodes of different

groups are connected, while within the groups the connections are sparse. For example, fraudsters in transaction networks are more likely to be connected to accomplices than to other fraudsters. In this case, fraudsters form heterophilic connections with the nodes of accomplices.

Computing trust relations is however a particularly difficult task, because of a number of challenges that already arise at data source level (i.e., not considering the inevitable bias of the particular algorithmic solution to the problem). In fact, the amount of information representing the observed interactions and activities of users in a social network, could be *limited* in size as well as in quality. More specifically, a social network may contain a significant amount of *redundant* or irrelevant relations as well as *noise* in the information that express the strength of interaction between any two users.

Contributions. In this work, we face the above challenges by addressing a new problem we named *Trust Network Inference* (TNI). Given a sequence of timestamped interaction networks as input, the goal of TNI is to infer from this sequence a directed weighted network, whose nodes are the users in the temporal networks and links denote trust relationships with associated trust scores.

It should be emphasized that in TNI there is no dependency on existing trust relations to make predictions on trustworthiness scores or on new trust relations. Therefore, TNI emerges as a divergence from the conventional trust inference and trust link prediction problems (e.g., [46, 88, 82, 63]). Also, TNI differs from trust ranking methods (e.g., [55, 94, 48]), since in TNI the building of trust relations is extended to all nodes in a network, not only to the most trusted or reputable ones. Furthermore, our TNI problem is different from the one treated in [33], which considers trustworthiness and untrustworthiness inference through clustering all entities into two groups (i.e., good and misbehaved), under various representative attack models.

We propose to solve the TNI problem based on a generalized *preference learning* paradigm. We believe that preference learning provides key advantages in addressing all the aforementioned issues, i.e., limitedness, redundancy and noisy of the information about the users' interactions from which a trust network is to be inferred. More specifically, under a preference-based top-k selection problem, *our proposed approach aims to find a ranking of the preferential pairings that each target entity would choose to form its trust relationships.* To this purpose, we resort to an adaptive sampling strategy, and instatiate it according to three canonical ranking models that correspond to different levels of ranking pairwise preferences. One further key feature of our approach is *domain-independency*, as it does not rely on platform-specific types of user interactions. Nonetheless, our approach is designed to exploit both topological information and, when available, content information relating to the user interaction dynamics.

Evaluating inferred trust relations and associated scores is another critical aspect in research contexts related to trust computing. In this work, we also cope with such a challenge and devise two scenarios based on distinct notions of *ground-truth*: the one referring to the availability of *trust classes* (i.e., cohesive groups of mutually trusted users), and the other corresponding to the availability of a *reference trust network*. Our extensive, ground-truth-driven experimental evaluation has shown the meaningfulness of our proposed approach in both evaluation scenarios, on several dynamic interaction networks and against competing methods.

Plan of the chapter. The remainder of this chapter is organized as follows. Section 5.2 briefly discusses related work on trust inference — note that, given the relative novelty of the TNI problem under consideration, we shall provide a deliberately concise summary of major existing notions and approaches to trust inference,

without any ambition to survey methods for trust computing. Section 5.3 introduces the problem of Trust Network Inference, and Section 5.4 describes our proposed approach. Sections 5.5 and 5.6 present methodology, data and results of our experimental evaluation. Finally, Section 5.7 concludes the chapter.

5.2 Related work on trust inference

Trust inference has attracted much attention in data mining and related fields, and a variety of studies have been proposed in literature [109]. One way of interpreting the problem of trust inference is to model it in terms of either *edge feature* or *node feature*, a.k.a. "local" and "global" trust computing. In the first case, a trust relation is to be created for any two non-adjacent nodes in a network, through a mechanism of *inference*, resp. *prediction*, if the network is modeled on existing trust relations (i.e., it is a trust network) (e.g., [46, 88, 82, 63]), resp. on social network features (e.g., [9, 116]). Conversely, trust inference at node-level corresponds to computing a trust score for each node in a network, and hence it is more appropriately regarded as a trust-oriented global *ranking* of the users, which can be useful to build trust communities [94], or in general to discriminate between objectively trust and distrust entities in a network (e.g., [55, 99, 48]).

Our work refers to the local-trust computing perspective. However, as already mentioned in the Introduction, we address the trust network inference problem, for which the trust network is the output, rather than the input as in conventional trust inference approaches. Note also that our work is substantially different from previous attempts to TNI-related problems, such as [63]: in that work, a user-domain-based trusted acquaintance chain discovery algorithm is developed to make the computation of short trusted paths more efficient; however, unlike our approach, the method in [63] strongly depends on the definition of domains/categories for the content in the input social network. Also, our inference problem is different from the one considered in [33], which assumes that all entities are clustered into two groups (i.e., good and misbehaved entities), and a belief propagation method is developed to estimate that one entity belongs to different groups, simultaneously inferring its trustworthiness and untrustworthiness values, according to different attack models in interactional networks.

5.3 Problem statement

We are given a set $\mathcal{V}$ of *entities* in a social environment (i.e., users), and a *temporal network* $\mathcal{G}$ as a series of graphs over discrete time steps $(G_1, G_2, \ldots, G_T)$, with time horizon T, where $G_t = \langle V_t, E_t, w_t \rangle$, with $1 \leq t \leq T$, is the graph at time t, with set of nodes V_t and set of directed edges E_t. Each node in V_t corresponds to a specific instance from the subset $\mathcal{V}_t$ of entities that occur at time t. Note that entities might occasionally appear and disappear in different time steps. Each edge $e = (v_i, v_j) \in E_t$ corresponds to an observed *interaction* between nodes v_i, v_j, which can be of different type depending on the specific functionalities and information available from the online social environment under consideration (e.g., mentions, answers/replies, re-posts, etc). The snapshot graphs G_t are also associated with an edge weighting function $w_t(\cdot)$ to quantify the strength of each interaction; by default, the weight of an edge is set to 1.

We consider the Trust Network Inference (TNI) problem, that is, generating a new network from interactional dynamics observed through $\mathcal{G}$, whose nodes correspond to the entities $\mathcal{V}$ in $\mathcal{G}$ and links are inferred to denote a trust/distrust relationship between

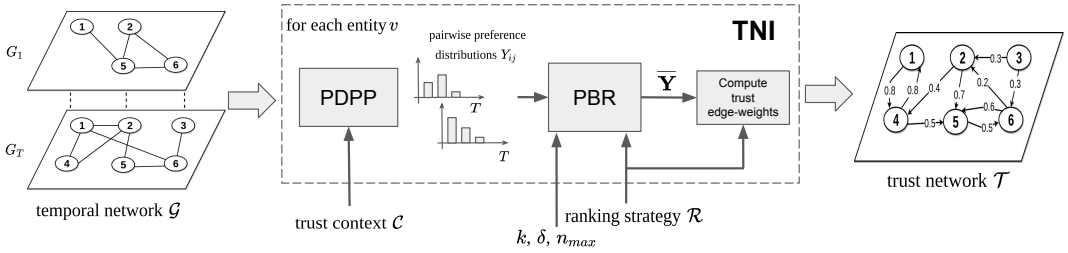


FIGURE 5.1: Overview of our proposed framework for trust network inference

any two entities that satisfy certain relational constraints. Such constraints are meant to be specified w.r.t. a predetermined scheme of selection of **trust-context**, denoted as $\mathcal{C}$.

The trust-context is a model for inducing a subgraph of $\mathcal{G}$ from each entity v, denoted as C_v, whose structural expansion intuitively corresponds to the extent of trust that v can exert towards other entities. Note that the induced trust-context subgraphs of any two entities are not to be necessarily disjoint.

We will refer to the **trust network** inferred from $\mathcal{G}$, w.r.t. a trust-context scheme $\mathcal{C}$, as a *weighted directed graph* $\mathcal{T} = \langle \mathcal{V}, \mathcal{E}, \omega \rangle$, with set of trust links $\mathcal{E} = \bigcup_{v \in \mathcal{V}} \mathcal{E}_v$, where $\mathcal{E}_v$ is a set of edges between entities in the node-set of the induced subgraph C_v for v in accord with $\mathcal{C}$, and $\omega : \mathcal{E} \rightarrow [0, 1]$ is a weighting function that specifies the trust level of each link, where 0 means total lack of trust (i.e., distrust) and 1 means fully trust from a source to a target node. We intuitively formulate the TNI problem as follows:

Problem 3 (Trust Network Inference (TNI)) Given a temporal network $\mathcal{G}$ built over interactions observed in a time period T between entities in a set $\mathcal{V}$, and given a trust-context scheme $\mathcal{C}$, infer a trust network $\mathcal{T}$ for all entities in $\mathcal{V}$ by exploiting the topological information available from each snapshot of $\mathcal{G}$ (along with, optionally, content-based information of the interactions) according to the trust-context scheme $\mathcal{C}$.

5.4 Our proposed method for Trust Network Inference

We propose to solve the TNI problem through a generalization of the *preference-based top-k selection problem* over each entity in the input temporal network. Next, we provide background on that, then we discuss details on our proposal. Table 5.1 summarizes main notations used throughout the rest of the chapter.

5.4.1 The TNI algorithm

Given a temporal network $\mathcal{G}$, we solve the TNI problem as a generalized preference-based top-k selection problem, for each entity in $\mathcal{G}$, under constraints given by a predefined trust-context scheme $\mathcal{C}$. The model $\mathcal{C}$ is used to determine the options $\mathcal{O}$ for pairing each target entity with its "trustworthy" entities.

Our idea is to generate the edges and associated scores of the trust network to be inferred on the basis of the solution of a *preference-based racing* (PBR) algorithm applied to each target entity. PBR is a particular approach to the top-k selection problem based on an adaptive sampling strategy.

TABLE 5.1: Main notations and their descriptions

symbol	*description*
$\mathcal{G}$; $G_t = \langle V_t, E_t, w_t \rangle$	series of graphs; graph at time t
$\mathcal{V}$; $\mathcal{V}_t$	set of entities or actors in $\mathcal{G}$; in G_t
$\mathcal{T} = \langle \mathcal{V}, \mathcal{E}, \omega \rangle$	trust network (to be inferred from $\mathcal{G}$)
$\mathcal{C}$; C_v	trust-context model; trust-context (induced subgraph) for entity v
o; $\mathcal{O}$	option; set of options (alternatives)
N; k	total (resp. selected) number of options
$\mathcal{R}$	ranking model
$\prec^{\mathcal{R}}$	strict preference order relation over a pair of options, according to $\mathcal{R}$
CO	Copeland's ranking model
SE	sum-of-expectations ranking model
RW	random-walk ranking model
$1 - \delta$	predefined confidence (for the top-k selection problem)
$Y_{i,j}$	random variable associated to comparison of o_i with o_j
$y_{i,j}^{(t)}$	t-th observed outcome of $Y_{i,j}$
$\mathbf{Y}$	preference relation matrix
n_{max}	number of samplings for each pairwise preference probability distribution $Y_{i,j}$
$sim_S^{(t)}$; $sim_C^{(t)}$	structural (resp. content) affinity function for node comparison in G_t
α	smoothing parameter to weight $sim_C^{(t)}$ w.r.t. $sim_S^{(t)}$

Algorithm 6 TRUST NETWORK INFERENCE($\mathcal{G}$=(G_1, ., G_T), $\mathcal{C}$, $\mathcal{R}$, k, n_{max}, δ)

1: $\Upsilon \leftarrow \emptyset$
2: **for all** $v \in \mathcal{V}$ **do**
3: $\quad \mathcal{O}_v \leftarrow computeTrustContextOptions(\mathcal{G}, v, \mathcal{C})$
4: $\quad \mathbf{Y} \leftarrow \mathsf{PDPP}(\mathcal{G}, v, \mathcal{O}_v)$ {*Probability distributions of pairwise preferences for* v}
5: $\quad \bar{\mathbf{Y}} \leftarrow \mathsf{PBR}(\mathbf{Y}, \mathcal{O}_v, k, n_{max}, \delta, \mathcal{R})$ {*Preference-based racing to compute the ranking scores (Algorithm 2)*}
6: $\quad \Upsilon \leftarrow \Upsilon \cup \{\bar{\mathbf{Y}}\}$
7: **end for**
8: $\langle \mathcal{E}, \omega \rangle \leftarrow computeTrustEdges(\Upsilon, \mathcal{R})$
9: **return** $\mathcal{T} = \langle \mathcal{V}, \mathcal{E}, \omega \rangle$

A schematic depiction of the proposed framework for trust network inference is presented in Figure 5.1, whereas Algorithm 6 shows the pseudo-code of our TNI method. The algorithm works as follows: for each entity v in the temporal network $\mathcal{G}$, it starts with the identification of the entity-options for v according to a predefined trust-context model $\mathcal{C}$ (Line 3). Then, the probability distributions of pairwise preferences (PDPP) $\mathbf{Y}$ are computed for v based on its interaction activities observed in $\mathcal{G}$ (Line 4). Using a preference-based racing algorithmic scheme, a ranking of trust relations is

computed for v according to a selected ranking model $\mathcal{R}$ (Line 5). Finally, the solutions provided by the racing procedure for all entities are used to determine both the edges and the trust scores (Line 8) to output the trust network $\mathcal{T}$. We now elaborate on each of the main steps in Algorithm 6.

Computing the trust-context of entities

The trust-context model $\mathcal{C}$ corresponds to the search space for the entity-options to identify as the trustworthy ones for any given target entity. One intuitive way of defining $\mathcal{C}$ is to instantiate it as the ego-network of the target entity. This notion is also supported by previous studies on trust inference which have provided evidence on that shorter paths from the trustor are more accurate to predict trust [46], and that the dilution of trust through the propagation process tends to weaken the predicted trust [65].

In the following, we will refer to the above definition of trust-context model, restricted to the out-neighborhood of any target entity v, i.e., all entities occurring as out-neighbors of v in at least one snapshot graph in $\mathcal{G}$. Clearly, the search space for the TNI problem can also be defined according to other topological structures, such as expanded ego-networks or community structures. This is *left as a further direction of research.*

Building the preference distributions

As previously discussed in Sect. 2.3.2, the true pairwise preference distributions are assumed to be unknown, however their realizations (i.e., outcomes of random variables $Y_{i,j}$) can be estimated as the observations of interaction at each snapshot G_t.

Given a target entity v in $\mathcal{G}$, every pair of entities occurring within its trust-context C_v are regarded as options for v, which can be compared at most T times. For entities v_i and v_j, we denote the outcomes of these comparisons (w.r.t. v) as $Y_{i,j} = y_{i,j}^{(1)}, \ldots, y_{i,j}^{(T)}$. To build each of the pairwise preference distributions for any entity v, we consider, for each snapshot $G_t = \langle V_t, E_t, w_t \rangle$, the set of v's outgoing nodes, denoted as $N_t(v)$, and evaluate the following outcomes for the variables $Y_{i,j}$ associated to v:

Outcome 1: $v_i \notin N_t(v) \wedge v_j \notin N_t(v)$. In this case, the two entities v_i and v_j are not comparable at time t, although, being both in C_v, they will be in some other snapshot. But at time t, v_i and v_j will not be considered to determine $y_{i,j}^{(t)}$.

Outcome 2: $v_i \in N_t(v) \vee v_j \in N_t(v)$. Let us consider a node-similarity function $sim^{(t)} : V_t \times V_t \mapsto [0, 1]$ and define it as a linear combination of two functions:

- a ***structural affinity*** function $sim_S^{(t)}$: this can efficiently be computed by resorting to standard *neighborhood-based overlap* measures; for instance, Jaccard similarity, i.e., $sim_S^{(t)}(v, v_i) = \frac{|N_t(v) \cap N_t(v_i)|}{|N_t(v) \cup N_t(v_i)|}$), or Adamic-Adar index, i.e., $sim_S^{(t)}(v, v_i) = \sum_{u \in N_t(v) \cap N_t(v_i)} \log(|N_t(u)|)^{-1}$. One alternative is to consider a vector similarity function to apply to the multidimensional representations of any two nodes, which would be obtained through *node-embedding* techniques in graphs, such as, e.g., *node2vec* (see [16] for a comprehensive survey).

- a ***content affinity*** function $sim_C^{(t)}$: the edge-weighting function w_t expresses the strength of content-based interaction for any two nodes in G_t, therefore $sim_C^{(t)}(v, v_i) := w_t(v, v_i)$. To this aim, we might consider the opportunity of computing a *sentiment score* associated with the available text content (cf.

Algorithm 7 PDPP($\mathcal{G} = (G_1, \ldots, G_T), v, \mathcal{O}$)

Initialize $\mathbf{Y} = [Y_{i,j}]_{N \times N}$ with empty lists
for $t = 1$ to T **do**
  for $(v_i, v_j) \in \mathcal{O} \times \mathcal{O}$, $v_i \neq v_j \neq v, v_i \neq v$ **do**
    if $v_i \in N_t(v) \vee v_j \in N_t(v)$ **then**
      Compute $sim^{(t)}(v, v_i)$ and $sim^{(t)}(v, v_j)$
      $y_{i,j}^{(t)} \leftarrow \Pr(v_i \succ v_j)$ {*Using Eq. 5.2*}
      $add(Y_{i,j}, y_{i,j}^{(t)})$
return $\mathbf{Y}$

Sect. 5.5). Note that, if no content-based information is associated with the interaction between v and v_i at time t, $w_t(v, v_i)$ is assumed to be 1.

The two above functions are hence combined as follows:

$$sim^{(t)}(v, v_i) = \alpha \cdot sim_C^{(t)}(v, v_i) + (1 - \alpha) \cdot sim_S^{(t)}(v, v_i), \tag{5.1}$$

for any pair (v, v_i), with $\alpha \in [0, 1]$ (by default set to 0.5).

Finally, we compute the v's preference of choosing v_i over node v_j at time t as the probability value given by the following logistic function:

$$y_{i,j}^{(t)} := \Pr(v_i \succ v_j) = \frac{1}{1 + e^{-f(i,j) \cdot (sim^{(t)}(v,v_i) - sim^{(t)}(v,v_j))}}, \tag{5.2}$$

where $f(i, j)$ corresponds to the steepness of the logistic, we define as $f(i, j) = \lambda \cdot (sim^{(t)}(v, v_i) + sim^{(t)}(v, v_j))$, where λ is a scaling factor. Our motivation behind this analytical choice is twofold. First, since the similarity values range in $[0, 1]$, and hence their differences range in $[-1, 1]$, the full domain of values of the logistic function would not be used if the steepness value was 1. Therefore, we introduce a scaling factor to better distribute the $y_{i,j}^{(t)}$ values within $(0, 1)$; for this purpose, we set λ to 10, which ensures the spanning through the interval $(0, 1)$. Moreover, our definition of the steepness function and λ setting is such that the sum of similarities is considered to weight more pairwise comparisons between more similar entities than dissimilar ones. Note also that Eq. 5.2 is symmetric, i.e., $\Pr(v_i \succ v_j) = 1 - \Pr(v_j \succ v_i)$.

Upon the above definitions, we build the pairwise preference distributions for a target node v as shown in Algorithm 7. Sampling from these distributions will correspond to randomly extracting an element from the lists $Y_{i,j}$. It should be emphasized that this sampling is important to ensure robustness of the whole approach w.r.t. noisy comparisons; we shall discuss this point later in Sect. 5.4.1 The output of Algorithm 7 then becomes the input for the preference-based racing algorithm (Algorithm 1). It should be noted that our approach to the computation of pairwise preference distributions diverges from the one adopted in [15]: here, while we still do not evaluate single options quantitatively (as in value-based racing), we let any variable $Y_{i,j}$ assume values within the range $(0, 1)$, to express a *degree* of preference of o_i over o_j, rather than a 0/1 (or ternary) decision (cf. Sect. 2.3.2).

Preference-based Racing

Following [15], the preference-based racing (PBR) procedure, shown in Algorithm 1, is responsible for identifying, among the entities in the context $\mathcal{O}_v$ of an input target entity v, the top-k trustworthy ones (or equivalently the top-k trust edges) according

to a predefined ranking model $\mathcal{R}$. Besides $k, \mathcal{R}$, and the probability guarantee (δ, cf. Sect. 2.3.2), the algorithm requires an additional parameter, n_{max}, to control the number of samplings for each pairwise preference probability distribution (i.e., $Y_{i,j}$, with $o_i, o_j \in \mathcal{O}$).

As mentioned before, the sampling step from each of the pairwise preference probability distributions lends the algorithm more robust to the presence of "noise", i.e., irrelevant node-relations such as sporadical links and/or wrongly observed links that may occur across the input temporal network.

The role of k in the PBR procedure. It is worth noting that Algorithm 1 outputs the top-k trustworthy nodes together with the *whole* preference estimates $\bar{\mathbf{Y}}$, which are fed into the *computeTrustEdges* function to finally compute the trust edge-weights in the trust network. This is done since, besides identifying the top-k trust edges (i.e., trust relationships), our goal is also to infer distrust links, which can be extracted through $\bar{\mathbf{Y}}$. In other terms, k takes the role of model parameter in the PBR procedure and only within the scope of this procedure; by contrast, in order to infer the trust network, all preference estimates may be taken into account so that each node may have more than k trust/distrust outgoing links.

Computing the trust edge-weights

For any given target entity v, the edge-weights in the trust network being generated are differently computed depending on the chosen ranking model and sampling strategy. For each v_i in the $\bar{\mathbf{Y}}$ matrix associated to v, using the Copeland's ranking, we set $\omega(v, v_i) = \frac{|\bar{y}_{i,j}:\bar{y}_{i,j}>\frac{1}{2}, i\neq j|}{|\mathcal{O}_v|}$. Note that the normalization is required since we want trust scores ranging in $[0, 1]$. For the other two sampling strategies, no normalization is required since the ranking scores are already in $[0, 1]$. In fact, for the SE-based strategy, we set $\omega(v, v_i) = \bar{y}_{i,j}$, whereas for the RW-based strategy, we set the edge weights to the values stored in the stationary distribution π, i.e., $\omega(v, v_i) = \pi_{v_i}$.

5.4.2 Computational complexity aspects

The time complexity of TNI is determined by the cost of its two main phases: computing the preference probability distributions and preference-based racing.

Given a target entity v and its context $\mathcal{O}_v$, the time complexity of building its preference distributions (Algorithm 7) is $O(T|\mathcal{O}_v|^2\tau_{sim})$, where τ_{sim} is the cost of similarity computation. This is explained since we need to make $|\mathcal{O}_v|(|\mathcal{O}_v| - 1)/2$ pairwise preference comparisons (through Eq. 5.1) between entities $v_i, v_j \in \mathcal{O}_v$ for each of the T timesteps, and each of these comparisons involves two structural similarity computations (i.e., $sim(v, v_i)$ and $sim(v, v_j)$).

The asymptotic cost of the second phase (Algorithm 1) is determined by the loop which, in the worst case, is executed n_{max} times when a satisfactory (according to δ) solution to the PBR problem cannot be found before.The cost of each iteration is $O(|\mathcal{O}_v|^2 + \tau_{SS})$, where τ_{SS} is the cost of the sampling strategy. Moreover, $\tau_{SS} = O(|\mathcal{O}_v|^2)$ for each of the sampling strategies we considered, because we need to check (in constant time) a condition for each pair of options (Line 4 in Algorithm 2). Thus, the cost of the second phase is $O(n_{max}|\mathcal{O}_v|^2)$.

The temporal cost of TNI for each entity v is $O(T|\mathcal{O}_v|^2\ \tau_{sim} + n_{max}|\mathcal{O}_v|^2) = O(|\mathcal{O}_v|^2(T \cdot \tau_{sim} + n_{max}))$, and the total cost is $O(\sum_{v\in\mathcal{V}}|\mathcal{O}_v|^2\ (T\tau_{sim} + n_{max}))$.

The spatial cost to solve TNI for each target entity v is determined by the space needed to store the pairwise preference distributions, thus its asymptotic growth is $O(T \cdot |\mathcal{O}_v|^2)$, since we need to store for each timestep the $O(|\mathcal{O}_v|^2)$ pairwise preference

TABLE 5.2: Ground-truth based evaluation types

	trust relation	*explicit network information*	*domain type*	*case studies*
Trust-Class	within-group links	no	Inferring trust network from interactions in real-life parties	Political parties
		yes	Inferring trust network from interactions in online collaborative system	Wikipedia editing
Trust-Network	individual pairs	yes	Inferring trust network from interactions in profit-based circles	Product rating

realizations which made up the distributions. The overall space complexity is $O(T \cdot (max_{v \in \mathcal{V}} |\mathcal{O}_v|)^2)$, since we can sequentially and independently solve the set of $|\mathcal{V}|$ PBR problems. Nonetheless, the approach is easily parallelizable by partitioning the set of entities, independently solving the PBR subproblems, then merging the results.

5.5 Evaluation methodology

We present our ground-truth-based methodology (Sect. 5.5.1), the evaluation criteria (Sect. 5.5.2) and datasets (Sect. 5.5.3). Also, in Sect. 5.5.4, we discuss the methods involved in a stage of comparative evaluation with TNI.

5.5.1 Ground-truth for trust network inference

To assess the meaningfulness of the results obtained by our TNI, we conducted different stages of evaluation based on two general, all-inclusive notions of ground-truth. These are hereinafter referred to as ***trust-class ground-truth*** and ***trust-network ground-truth***. As reported in the summary of Table 5.2, a ground-truth in our setting is either based on the notion of *trust class* or on the availability of a *reference trust network* for the input dynamic network.

The former corresponds to trust relations existing within a cohesive group of users in the input dynamic network, i.e., a trust class is regarded as a group of individuals whereby it is likely that they trust each other while they do not trust individuals outside the group. As exemplary domains, we recognize inferring trust network from interactions in real-life parties (i.e., contact networks) and from interactions occurring in collaborative networks (Table 5.2). Notably, given the relation of trust classes with the time-evolving interaction data, this ground-truth-based approach can help assess the discovery of trust/distrust relationships that are latent in the interaction data.

Trust-network ground-truth instead relies on a finer-grain type of trust relation, i.e., between pairs of users, which corresponds to the availability of a trust network that is regarded as a reference for the input dynamic network. The challenge in this case is that the ground-truth network may not be necessarily derived from interactions observed in the input time-evolving network data (i.e., two users may trust each other even though they never had a direct connection).

It should be emphasized that both the ground-truth classes and the reference trust networks were used *not to infer* a trust network, but *only for evaluation purposes*.

5.5.2 Assessment criteria

Given a trust network $\mathcal{T} = \langle \mathcal{V}, \mathcal{E}, \omega \rangle$ inferred from a dynamic interaction network $\mathcal{G}$, we define a ground-truth trust classification as a partitioning Γ of the set of entities $\mathcal{V}$ into disjoint trust classes. Also, we denote with $\Gamma(v)$ the trust-class of entity v. We considered the following trust-class ground-truth based assessment criteria, for each entity v:

- *Binary preference* (Bpref) [14], which measures how many judged relevant candidates *Rel* are retrieved (i.e., occur in $\mathcal{T}$) ahead of judged irrelevant candidates

TABLE 5.3: Main structural features of our evaluation network datasets

	#entities ($\lvert\mathcal{V}\rvert$)	*#edges*	*#time steps* (T)	*avg. density*
DKpol, *DKpol-c*	490	1 821	30	0.074
DKpol-exp, *DKpol-exp-c*	490	288 680	32	0.047
WikiEdit *WikiEdit-exp*	1 115	33 304	49	0.06
CiaoDVD *CiaoDVD-c*	17 615	348 791	27	0.0174

notRel:

$$bpref(v) = \frac{1}{|Rel|} \sum_{v_r \in Rel} 1 - \frac{|rankedHigher(v_r)|}{|Rel|},$$

where v_r is a relevant retrieved candidate, v_i is a member of the first $|Rel|$ irrelevant retrieved candidates, and $rankedHigher(v_r) = \{v_i \in notRel \mid \omega(v, v_i) > \omega(v, v_r)\}$. We define Rel (resp. $notRel$) as the set of out-neighbors of v in $\mathcal{T}$ such that $\Gamma(u) = \Gamma(v)$ (resp. $\Gamma(u) \neq \Gamma(v)$). The *global bpref* of a trust network is computed as the average entity bpref, i.e., $bpref(\mathcal{V}) = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} bpref(v)$.

- *Average intra-class trust*, as the average trust amount settled by v towards individuals within the same trust-class:

$$\Omega_{\Gamma}(v) = \frac{1}{|Rel|} \sum_{v_r \in Rel} \omega(v, v_r).$$

- *Average extra-class trust*, as the average trust amount settled by v towards individuals outside the v's trust-class:

$$\Omega_{\neg\Gamma}(v) = \frac{1}{|notRel|} \sum_{v_i \in notRel} \omega(v, v_i).$$

For the second type of ground-truth-based evaluation, given the availability of a reference trust network, we used it for a network similarity evaluation task, measuring *Precision*, *Recall* and *F1-score*. For any given $\mathcal{T}$ produced by TNI and reference network $\mathcal{T}^*$, both with set of nodes $\mathcal{V}$, precision (resp. recall) corresponds to the fraction of edges in $\mathcal{T}$ (resp. $\mathcal{T}^*$) shared with the other network, whereas F1-score is the harmonic mean of precision and recall.

5.5.3 Case studies and datasets

We used 3 real-world, publicly available datasets: *DKpol* [58], *WikiEdit*,[1] and *CiaoDVD* [53]. According to Table 5.2, the former two were used for the trust-class ground-truth evaluation, the latter for the trust-network ground-truth evaluation. Table 5.3 provides a summary of structural characteristics of our evaluation networks. Also, we considered content-based variants, for a *total of 8 networks used in our evaluation.*

[1] It will be made available at http://people.dimes.unical.it/andreatagarelli/data/.

DKpol: Trust inference for political parties

DKpol contains Twitter following and activity data (i.e., tweets, retweets and replies) originally collected from the profiles of Danish politicians during the month leading to the parliamentary election in 2015. The profiled 494 politicians are distributed across 10 parties, each of which was regarded as one trust-class, i.e., politicians who are affiliated to the same party are supposed to trust each other, while distrusting politicians of other parties. By aggregating the user interactions on a daily basis, we extracted 30 directed networks such that, in the t-th snapshot, an edge from u to v is drawn if, at time t, u mentioned v, retweeted a v's tweet, or replied to a v's tweet. Starting from *DKpol*, we built a weighted network variant, dubbed *DKpol-c*, whereby the tweet contents are subjected to tool for sentiment analysis in Danish texts [98]. Each edge (u, v) in *DKpol-c* is weighted with a float value in $[0, 1]$ corresponding to the highest mood-score computed by the tool for the text of the tweet(s) posted by v and mentioned/replied/retweeted by u.

In order to stress our approach, we added noise to the data by simulating a *multicast propagation* of tweets/retweets made by a user towards her/his followers. In this scenario, which resulted in the *DKpol-exp* network, a tweet/retweet of user u triggers a set of links from u's followers to u. In addition, we built a content-based weighted variant, *DKpol-exp-c*, whose follower links are weighted with the neutral score of 0.5. Our rationale is that such links correspond to weak ties and, therefore, they would not be considered for the direct linkage contribution in Eq. 5.1 (i.e., $sim_C^{(t)}$ set to zero), however they are still considered in the structural similarity computations.

WikiEdit: Trust inference for a collaboration system

Our second case study concerns the context of Wikipedia page editing, which normally gives rise to either controversy or agreement among the editors. Our goal was to infer a trust network by observing the editing activities made by a set of users over a selection of pages of VIPs (from politics, sport, and other categories). The possible edit events are 'add', 'delete' or 'restore' content. The amount of text involved in each edit is quantified by the number of used words. Based on this information, we built the temporal network *WikiEdit* by considering the edits related to 10 among the top-edited pages and aggregating the events on a monthly basis. The *WikiEdit* network was obtained by modeling each edit event (of any type) made by a user u at time t as a set of edges in the t-th snapshot directed from u to each other user involved in the edit. In particular, the 'add' event involves only the active user (who performs the edit) while each 'delete' or 'restore' event is also annotated with the target user (the one who previously added/deleted the text). Each interaction e between two users is also labeled with a sign: 'positive' if they agree with the edit corresponding to the interaction, 'negative' otherwise. We exploit this additional information, together with the number of words nw_e involved in the edit, in order to compute the weight w_e of the interaction by means the following logistic function:

$$w_e = (1 + e^{-sign(e)\cdot \log_{10}(1+wc_e)})^{-1},$$

where $sign(e) = +1$ if e is a positive interaction, -1 otherwise. Note that positive (resp. negative) interactions will have weights higher (resp. lower) than 0.5.

We also considered an expanded version of *WikiEdit*, dubbed *WikiEdit-exp*. In this case, for each 'add' edit to page p made by user u at time t, we created weak ties (with neutral weight 0.5) from u to each other user that added content to p before t

in order to represent a weak form of agreement of u towards the past 'add' edits made to p.

We created a graph where nodes are the page editors and links correspond to positive interactions between editors. On this graph, we applied the well-known Louvain community detection method [7] to obtain a partitioning of nodes that we consider as ground-truth communities for the evaluation.

Inferred trust network vs. reference trust network

For the trust-network ground-truth evaluation task, we considered the *CiaoDVD* dataset where users provide movie ratings (from 0 to 5) and can define their own local trust network by adding other users to their trust circle. The latter is considered as the ground-truth trust network for our evaluation.

We derived two temporal networks, *CiaoDVD* and *CiaoDVD-c*, where we aggregated the ratings on a monthly basis and extracted an edge from node u to v, in the t-snapshot, if there is at least one movie rated by both users in that month and v rated it before u. The rating similarity of the users is exploited to quantify the strength of interaction in the weighted version of the network, dubbed *CiaoDVD-c*. More specifically, given two users u and v and a set of M movies rated by both users at time t, and let $\mathbf{r}_u = [r_{u,1} \ldots, r_{u,M}]$ and $\mathbf{r}_v = [r_{v,1}, \ldots, r_{v,M}]$ be the associated ratings vectors, we quantify the strength of the interaction as:

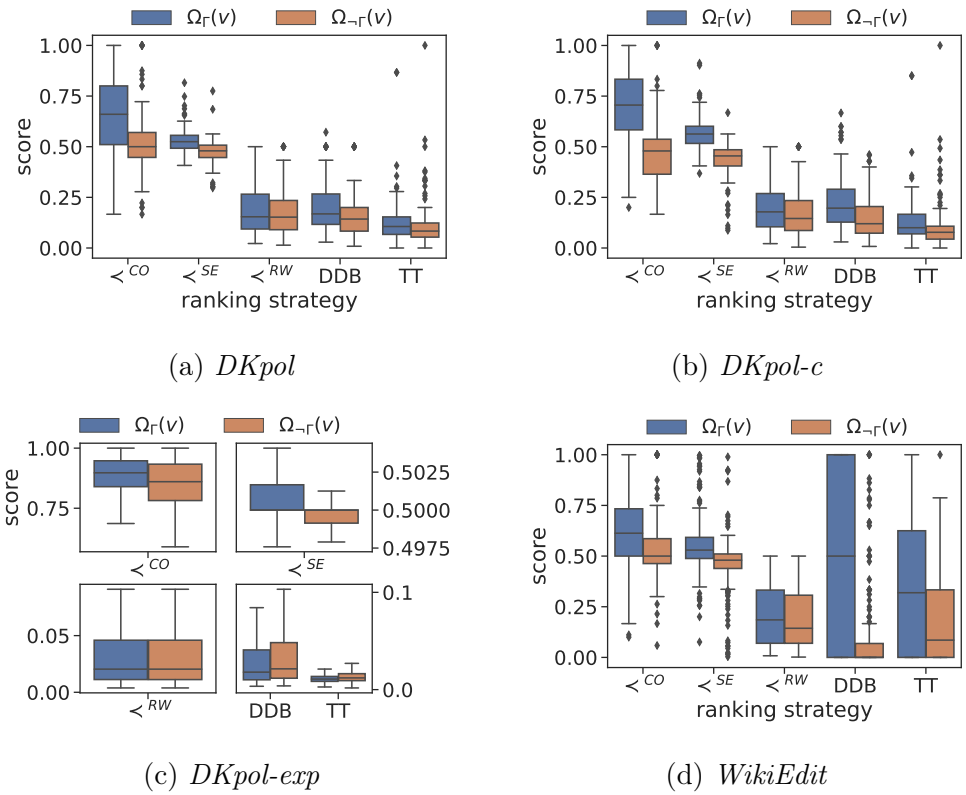
$$w(u,v) = 1 - \frac{1}{M} \sum_{i=1}^{M} \left| \frac{r_{u,i}}{5} - \frac{r_{v,i}}{5} \right|.$$

5.5.4 Competing methods

We finally considered a comparative evaluation stage with a twofold goal: comparing the trust network inferred by our TNI w.r.t. a trust network built by (i) a data-driven baseline and (ii) a local-trust inference method (cf. Sect. 5.2).

Our defined *data-driven baseline* (DDB) infers a trust network by aggregating the interactions observed in an input temporal network over all timesteps. In particular, for DKpol and CiaoDVD networks, the trust score of an edge (u, v) is computed as $W_{u,v}/W_u$, where $W_{u,v}$ here denotes the sum of weights of the interactions from u to v over all timesteps and W_u is the total sum of weights of interactions of u with any other node. For WikiEdit networks, the trust score of an edge (u, v) is computed as $W^+_{u,v}/(W^+_{u,v} + W^-_{u,v})$ where $W^+_{u,v}$ is the sum of weights of positive edits between u and v, while $W^-_{u,v}$ is the sum of the complement-one values of the weights of negative edits. This is explained to balance the numerical contributions given by positive interactions (i.e., edge weights above 0.5) vs. negative interactions (edge weights below 0.5). For example, suppose node u has one positive interaction with node v with weight 0.9 and five negative interactions all with weight 0.02: without complementing the negative interaction weights, the trust score of u to v would be $0.9/(0.9 + 5 * 0.02) = 0.9$ (i.e., high trust, which is counterintuitive); otherwise, it would be $0.9/(0.9 + 5 * 0.98) = 0.155$, which is more reasonable since it likely denotes a distrust relation rather than a trust one.

We chose the classic *TidalTrust* [46] (TT in short) as a representative local-trust inference method. This is designed to exploit the topological information in an input trust network for predicting a trust score for each pair of nodes that do not have a direct connection. The choice of selecting the shortest path derives from the hypothesis that reliability of trust values progressively decays proportionally to their distance



(a) *DKpol* (b) *DKpol-c*

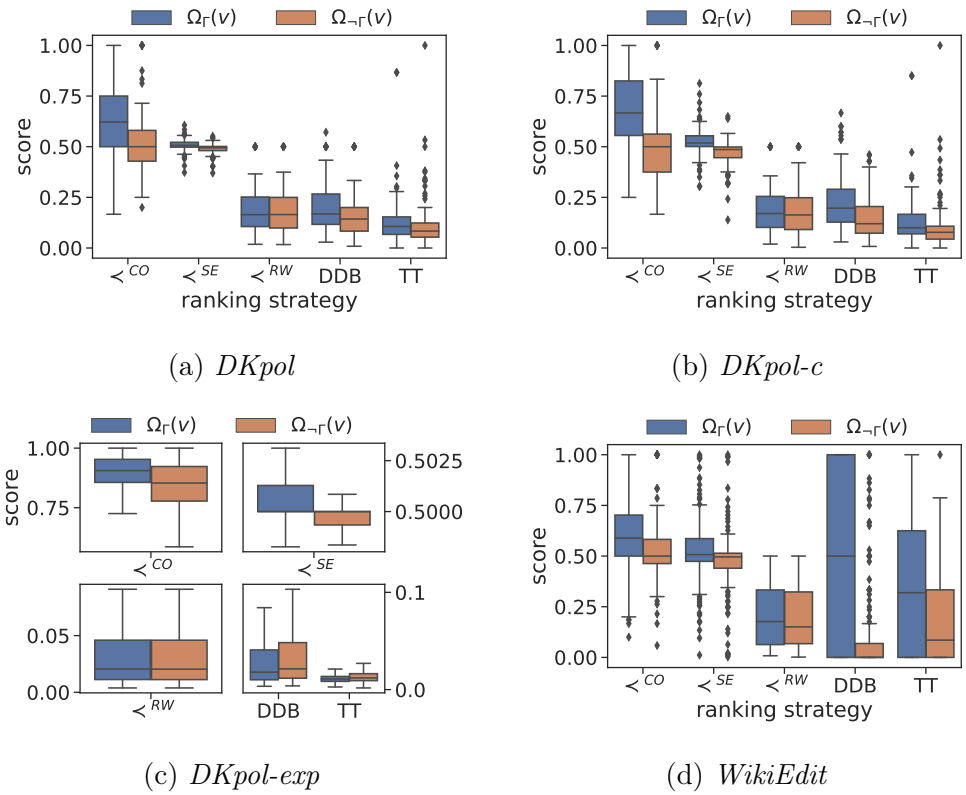
(c) *DKpol-exp* (d) *WikiEdit*

FIGURE 5.2: Trust-class ground-truth evaluation: Boxplots of the distributions of the average intra-class trust ($\Omega_\Gamma(v)$) and of the average extra-class trust ($\Omega_{\neg\Gamma}(v)$) values, with $\alpha = 0.5$

from the source node. The trust between non-adjacent nodes is inferred by considering only shortest paths through trusted neighbors. The trust from a source to a destination node is calculated by calling a recursive trust function on the trusted neighbors, which terminates when the destination is reached. When the trust is back propagated to the source, it is averaged and rounded among the different trusted paths. Also, a path-pruning threshold is set to the maximum of the lowest trust values in each individual path from source to destination node. We used TT as follows: From the trust network obtained through DDB, repeatedly remove one edge at a time from the baseline network, then apply TT to compute its trust score, until all edges in the network are examined.

5.6 Results

We present our main experimental results for each of the ground-truth-based evaluation stages (Sects. 5.6.1–5.6.2). In this regard, note that a major goal of our experimental analysis is the assessment of TNI by varying the setting of its main parameters; nonetheless, unless otherwise specified, we will present results that correspond to default settings for parameters δ (0.1), α (0.5), k ($|\mathcal{O}_v|/2$, for any v), n_{max} (100), and Jaccard similarity as topological overlap function. In Sect. 5.6.3, we also discuss TNI efficiency aspects. Finally, in Sect. 5.6.4, we summarize main experimental findings.



(a) *DKpol* (b) *DKpol-c*

(c) *DKpol-exp* (d) *WikiEdit*

FIGURE 5.3: Trust-class ground-truth evaluation: Boxplots of the distributions of the average intra-class trust ($\Omega_\Gamma(v)$) and of the average extra-class trust ($\Omega_{\neg\Gamma}(v)$) values, with $\alpha = 0.15$

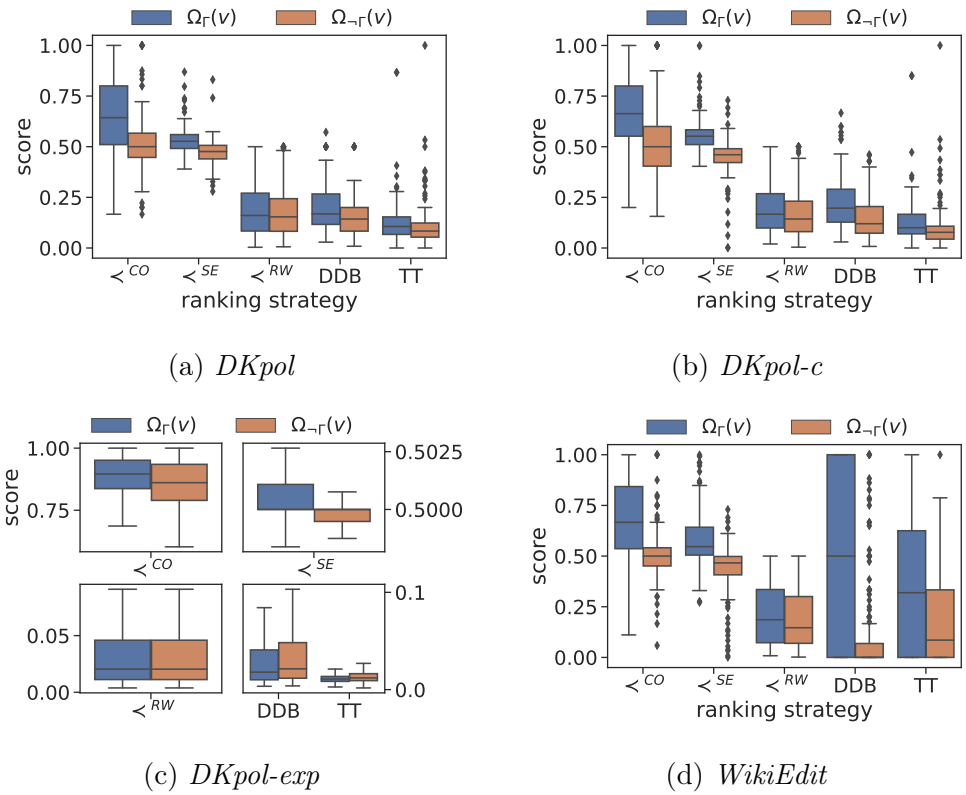
TABLE 5.4: Trust-class ground-truth evaluation: Global *bpref* results. Bold text refers to the best values per dataset

	TNI with $\prec^{CO}$			TNI with $\prec^{SE}$			TNI with $\prec^{RW}$			DDB	TidalTrust
	α=0.85	α=0.5	α=0.15	α=0.85	α=0.5	α=0.15	α=0.85	α=0.5	α=0.15		
DKpol	0.531	**0.532**	0.493	0.484	0.484	0.499	0.418	0.423	0.401	0.248	0.406
DKpol-c	0.576	0.635	0.579	0.603	**0.644**	0.582	0.438	0.456	0.456	0.566	0.463
DKpol-exp	0.433	0.436	**0.522**	0.177	0.194	0.209	0.155	0.168	0.178	0.234	0.266
DKpol-exp-c	0.445	**0.548**	0.522	0.195	0.210	0.213	0.163	0.175	0.174	0.239	0.272
WikiEdit	0.524	0.402	0.391	**0.554**	0.46	0.441	0.443	0.386	0.386	0.392	0.293
WikiEdit-exp	**0.378**	0.354	0.352	0.1	0.1	0.1	0.142	0.138	0.14	0.02	0.286

5.6.1 Trust-class ground-truth evaluation

Trust score distributions. For each entity (i.e., user in the input temporal network), we analyzed the distribution of its trust values among entities in the same ground-truth class and in the other classes. More specifically, we analyzed the boxplots of the distributions of $\Omega_\Gamma(v)$ and $\Omega_{\neg\Gamma}(v)$ values, over all entities in a network, for various sampling strategies and varying α. For the sake of presentation, we report here results corresponding to the default, balanced setting of α (i.e., 0.5) in Fig. 5.2, while results for $\alpha = 0.15$ (resp. 0.85) can be found in Fig. 5.3 (resp Fig. 5.4).

One important remark that supports the effectiveness of TNI is that, on average, an entity v tends to assign higher trust scores to entities in its ground-truth class ($\Gamma(v)$) than entities outside. This particularly holds, reagrdless of α in non-noisy networks (i.e., *DKpol*, *DKpol-c*, *DKpol-exp*) and for strategies CO and SE, which allow much clearer separation of the two distributions than the RW strategy. By contrast, it is worth emphasizing that the competitors can have an opposite trend, as in *DKpol-exp*,



(a) *DKpol* (b) *DKpol-c*

(c) *DKpol-exp* (d) *WikiEdit*

FIGURE 5.4: Trust-class ground-truth evaluation: Boxplots of the distributions of the average intra-class trust ($\Omega_\Gamma(v)$) and of the average extra-class trust ($\Omega_{\neg\Gamma}(v)$) values, with $\alpha = 0.85$

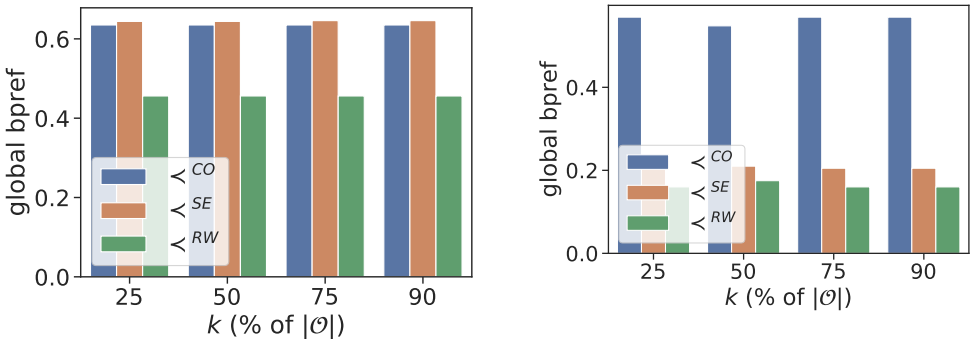
TABLE 5.5: Trust-network ground-truth evaluation: Precision, recall and F1-score results. Bold text refers to the best values per dataset, for each criterion

		Precision					Recall					F1-score				
		$\prec^{CO}$	$\prec^{SE}$	$\prec^{RW}$	DDB	TT	$\prec^{CO}$	$\prec^{SE}$	$\prec^{RW}$	DDB	TT	$\prec^{CO}$	$\prec^{SE}$	$\prec^{RW}$	DDB	TT
CiaoDVD	$\alpha = 0.85$	0.231	0.235	**0.236**	0.232	0.231	0.804	0.842	**0.849**	0.796	0.812	0.359	0.367	**0.369**	0.359	0.36
	$\alpha = 0.5$	0.231	0.231	**0.236**			0.804	0.822	**0.85**			0.358	0.36	**0.37**		
	$\alpha = 0.15$	0.22	0.231	**0.234**			0.743	0.82	**0.844**			0.34	0.36	**0.367**		
CiaoDVD-c	$\alpha = 0.85$	**0.239**	0.238	0.238	0.236	0.226	0.838	0.847	**0.87**	0.807	0.793	0.372	0.372	**0.374**	0.365	0.352
	$\alpha = 0.5$	0.233	0.237	**0.238**			0.826	0.859	**0.899**			0.363	0.371	**0.377**		
	$\alpha = 0.15$	0.228	**0.237**	0.234			0.828	0.871	**0.899**			0.358	**0.372**	**0.372**		

or even an overly positive-bias, as in *WikiEdit*.

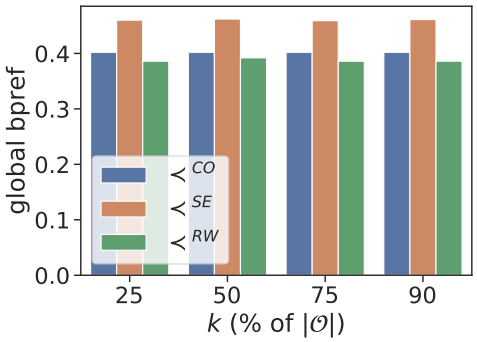
Moreover, considering the effect on the distributions by varying α (Figs. 5.4–5.3, in *Appendix*), while negligible differences can be observed between the corresponding cases, for each network and method, we also found no monotonic behavior in the distribution overlap by progressively varying α; for instance, the default value of 0.5 (cf. Fig. 5.2b) ensures better separation between the distribution boxplots than the other settings of α in *DKpol-c*, whereas for a network like *WikiEdit-exp* which was built on content-based collaborative editing, $\alpha = 0.85$ (cf. Fig. 5.4d) might be preferred to other settings.

Bpref analysis. Table 5.4 shows *bpref* results obtained by different variants of TNI and by competing methods. Several remarks stand out. First, concerning the sampling strategies, CO and SE models generally lead to better performance of TNI than in the RW case, on every dataset and regardless of the α setting. In particular, CO improves upon SE especially in the noisy (i.e., expanded) networks, while SE

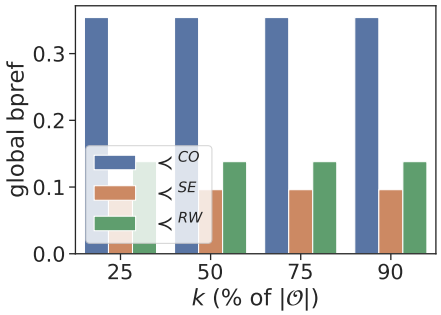


(a) *DKpol-c*

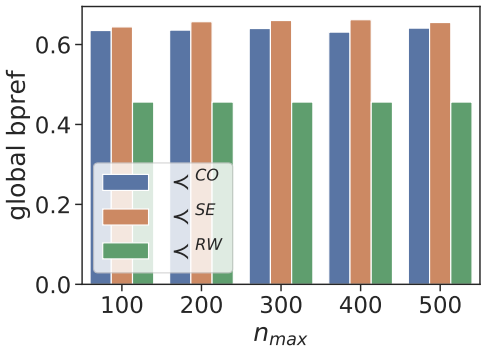
(b) *DKpol-exp-c*



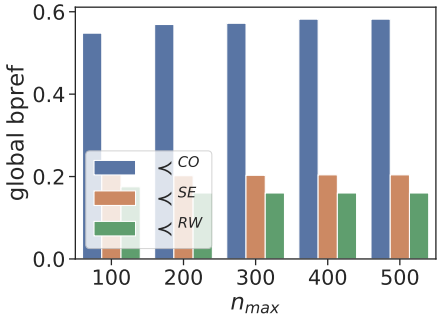
(c) *WikiEdit*



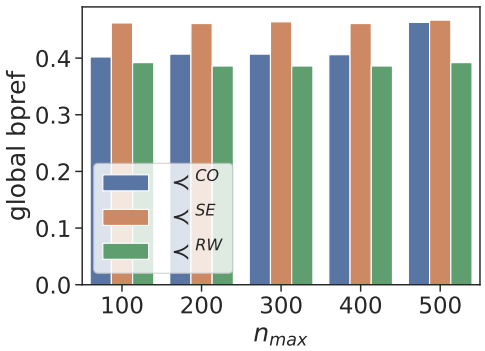
(d) *WikiEdit-exp*



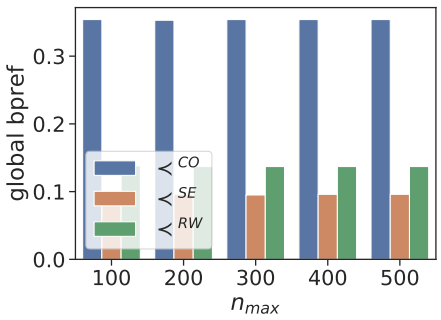
(e) *DKpol-c*



(f) *DKpol-exp-c*



(g) *WikiEdit*



(h) *WikiEdit-exp*

FIGURE 5.5: Trust-class ground-truth evaluation: Global $bpref$, (a)-(d) varying k (with fixed n_{max}) and (e)-(h) varying n_{max} (with fixed k)

prevails over CO in content-based networks (*DKpol-c* and *WikiEdit*). Second, TNI performance always increases when the network information is combined with content information to determine the preference probabilities (i.e., *DKpol-c* vs. *DKpol*, and *DKpol-exp-c* vs. *DKpol-exp*), regardless of the sampling strategy and α setting. Third, concerning the impact of parameter α, the balanced setting (i.e., $\alpha = 0.5$) leads to performance results that are comparable or better than for $\alpha = 0.85$ in the DKpol networks, while an opposite tendency is observed for WikiEdit networks, which are indeed more content-oriented than DKpol ones; analogously, $\alpha = 0.15$ may behave better than the balanced setting on noisy structure-oriented networks like *DKpol-exp*. Fourth, TNI significantly outperforms the competitor methods, at least when equipped with the CO strategy. DDB can behave better than TT (*DKpol-c* and *WikiEdit*), but the opposite holds on the noisy networks: this happens since TT is able to exploit the rich connectivity of expanded networks for inferring new trust links, while DDB considers the local interactions only.

Effect of k and n_{max}. Besides investigating the roles of the sampling stategy and of the α parameter, we also evaluated the impact of k and n_{max} on the TNI performance. To this end, we devised two stages: i) we varied n_{max} from the default 100 up to 500, while keeping k fixed to the default of half of the trust-context size, and ii) we varied k for different percentages of the trust-context size, with n_{max} fixed to 100. α was set to the default 0.5.

Figure 5.5 shows *bpref* results obtained for various sampling strategies. At first sight, it stands out that, in both evaluation stages and for each network dataset, the relative differences between the sampling strategies follow the same trend when varying k (Fig. 5.5a-d) and n_{max} (Fig. 5.5e-h), respectively. Also, our choice of default settings of the two parameters turns out to correspond to *bref* results that are very close to the performance peaks. Overall, this not only suggests relative robustness of TNI to variations of k and n_{max}, but also that the computational burden due to an increase of the values of the parameters can be avoided, since no particularly significant performance gain is guaranteed above specific values (i.e., default values).

Moreover, as already shown in Table 5.4, CO turns out to be the winner strategy for noisy networks (i.e., *DKpol-exp-c*, *WikiEdit-exp*), while SE prevails on other situations.

5.6.2 Trust-network ground-truth evaluation

For the second stage of evaluation, we filtered out the edges with trust scores below a certain threshold, which was set for each entity v as the 25-th percentile of the trust score of entities linked to v. Then, we derived an unweighted trust network to enable comparison with the unweighted reference networks of the CiaoDVD dataset.

Table 5.5 shows precision, recall and F1-score values w.r.t. the ground-truth trust network, for TNI (equipped with different sampling strategies and varying α) and competitors. Looking at the table, we observe that the best scores are always obtained by TNI, mostly with the RW strategy; this shows higher recall than the other strategies, while all three lead to similar performance in terms of precision and F1-score. Concerning precision in particular, the gap between TNI and the competitors is relatively small, and all achieve quite low values in both *CiaoDVD* networks. This is explained since the ground-truth network of CiaoDVD indeed was not derived from interaction data (i.e., a user may trust another one without interacting with her/him), thus the inferred trust network may not be in accord with the ground-truth knowledge. Mid-high values of recall are instead obtained on both networks, with the RW strategy outperforming SE and CO. In particular, TNI with RW or SE (along with

TABLE 5.6: TNI execution times (in seconds)

	TNI with $\prec^{CO}$			TNI with $\prec^{SE}$			TNI with $\prec^{RW}$		
	PDPP	PBR	total	PDPP	PBR	total	PDPP	PBR	total
DKpol	0.93	0.24	1.17	0.92	0.35	1.27	0.89	0.05	0.94
DKpol-c	0.98	0.2	1.18	1.02	0.26	1.28	1.01	0.06	1.07
DKpol-exp	420.25	44.32	464.57	420.05	66.44	486.49	420.63	1.32	421.95
DKpol-exp-c	409.75	52.38	462.13	410.83	86.52	497.35	423.07	2.4	425.47
WikiEdit	4.02	2.32	6.34	4.74	2.4	7.138	4.77	0.32	5.09
WikiEdit-exp	4172.50	110.8	4283.30	4173.96	230.24	4404.20	4176.44	12.32	4188.76
CiaoDVD	38452.02	310.60	38762.62	38454.04	521.52	38975.56	38501.02	43.48	38544.50
CiaoDVD-c	38545.02	322.10	38867.12	38540.53	543.85	39084.38	38543.52	53.48	38597

CO in *CiaoDVD-c*) outperforms the two competitors, despite their bias in producing high trust scores for most edges.

5.6.3 Efficiency evaluation

Table 5.6 shows the execution times of TNI, broken down into the procedures PDPP and PBR, using the default settings. It can be noted that, regardless of the particular network and strategy, most of the total running time is due to the PDPP procedure. Moreover, the RW strategy tends to yield better time performance of TNI, though of the same order of magnitude as for the other two strategies.

We also analyzed the time efficiency of TNI by varying the maximum number of samplings n_{max} from 100 to 500. Figure 5.6 shows time performances on DKpol and WikiEdit networks.[2] In accord with the computational complexity analysis (Sect. 5.4.2), the execution time for SE and CO strategies grows linearly with n_{max}. By contrast, the RW execution time grows much slower or even negligibly: this is explained since the value of n_{max} is checked by the RW strategy to decide if a pair of options does not need to be sampled anymore (Line 4 in Algorithm 2), and this leads the random walk to convergence faster than the other two strategies.

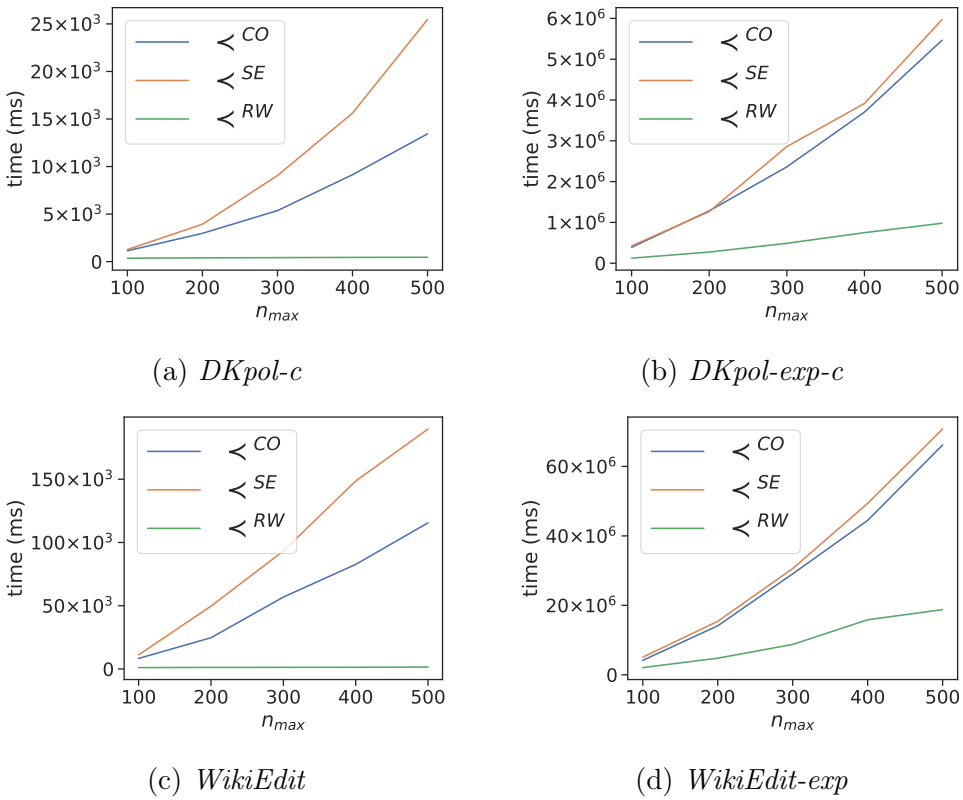
5.6.4 Discussion

We coped with the task of assessing our proposed TNI by designing ground-truth-based stages of evaluation. We believe this design is remarkable as it allowed us to define an all-inclusive approach to the exploitation of ground-truth knowledge for evaluation purposes. Our stages of evaluation of TNI have indeed been defined upon either a notion of trust-class (i.e., cohesive group of mutually-trusted users) or on the availability of a reference trust-network for the input dynamic network. As a consequence, we identified three representative application domains for TNI, with corresponding case studies that refer to relevant scenarios in network analysis.

Upon these premises, experimental results have revealed important findings about the meaningfulness and effectiveness of our proposed method. TNI is capable of inferring a trust network where each entity (i.e., user) observed in the input time-evolving network is associated with higher trust scores to entities in its ground-truth class than to entities of other classes. By contrast, competing methods fail in having this behavior, showing sometimes an opposite trend or even an overly positive-bias (i.e., much more trust links than expected).

Despite having a number of parameters, TNI has shown to be surprisingly robust to their variation; particularly, the sampling strategies (i.e., ranking models) follow similar trends when varying the top-k trusted options for every target entity, and

[2] Platform Linux (Mint 18), with 2.6 GHz Intel Core i7-4720HQ, 16GB RAM



(a) *DKpol-c* (b) *DKpol-exp-c*

(c) *WikiEdit* (d) *WikiEdit-exp*

FIGURE 5.6: TNI runtime performance by varying n_{max}

the number of samplings for each pairwise preference probability distribution (n_{max}). Also, using a balanced setting for α (i.e., the smoothing parameter that controls the contributions of structural and content information from the input network to determine the preference probabilities) has shown to be an appropriate default choice.

From an efficiency viewpoint, TNI running time grows linearly with the number of samplings. Overall, considering a trade-off between impact on the efficiency and impact on effectiveness of TNI, the sampling strategy based on the Copeland's ranking model turned out to be the best choice.

5.7 Chapter review

We introduced the Trust Network Inference problem and proposed a preference-learning-based approach to solve it. Our approach can be regarded as key-enabling for any application that needs to build a trust network associated with a social environment from user interactions observed over time, in order to exploit the inferred trust relatioships in a variety of mining tasks.

Several aspects in our approach are worthy to be further investigated. Different definitions of trust-context and of structural/content affinity functions could easily be integrated into our proposed TNI framework; for instance, as we mentioned earlier in the chapter, the trust-context model could be defined according to various topological structures, such as expanded ego-networks or community structures. Another aspect of interest is to extend our method to build a trust network incrementally in online tasks, i.e., inferring and maintaining/updating a trust network over a stream of interaction networks.

Chapter 6

Optimizing Interactions in Probabilistic Graphs Under Clustering Constraints.

Summary. We study two novel clustering problems in which the pairwise interactions between entities are characterized by probability distributions and conditioned by external factors within the environment where the entities interact. This covers any scenario where a set of actions can alter the entities' interaction behavior. In particular, we consider the case where the interaction conditioning factors can be modeled as cluster memberships of entities in a graph and the goal is to partition a set of entities such as to maximize the overall vertex interactions or, equivalently, minimize the loss of interactions in the graph. We show that both problems are **NP**-hard and they are equivalent in terms of optimality. However, we focus on the minimization formulation as it enables the possibility of devising both practical and efficient approximation algorithms and heuristics. Experimental evaluation of our algorithms, on both synthetic and real network datasets, has shown evidence of their meaningfulness as well as superiority with respect to competing methods, both in terms of effectiveness and efficiency.

6.1 Introduction

Modeling and mining behavioral patterns of users of online as well as offline systems is central to enhance the user engagement and experience in the systems. In this regard, *uncertain graph models* are seen as a powerful tool to capture the inherent uncertainty in user behaviors into a representation of user interaction patterns [71]. A common way of modeling uncertainty in a graph, which we refer to in this work, is to associate each pair of (linked) users with a probability value that expresses the likelihood of *observing* and *quantifying* an interaction between the two users. In this regard, one important aspect is that the modeling of user interactions should also account for exogenous conditions or events that occur within the social environment where the users belong to, which indeed can significantly affect the users' interaction behaviors. For example, delivering a post on a user's page (e.g., Facebook wall) that contains a message of friend recommendation will likely favor or not a meeting between two users, and so their interactions. Intuitively, it is of high interest to identify proper settings of a network system and relating conditions that can maximize the overall user interactions within the system. In this work, we extend the uncertain graph modeling framework to capture *the dependency of interactions on conditioning factors*, in a network system. In particular, we focus on the case when *the interaction behaviors depend on a clustering of the set of users* in a graph, so that the probability

of interaction between any two users varies depending on whether they belong to the same cluster or not. Modeling such interaction conditioning factors in terms of cluster memberships of users arises in several relevant application scenarios. Let us discuss on a couple of them.

Applications. Consider a social-media platform where users produce, exchange and consume content items. Each user is typically associated with a personal profile page. This acts also as an interface for the platform to deliver recommendations and advertisements to a target user u, including those contents produced by other users which u may interact with. The probability that an interaction between two users u and v will occur, in relation to a content item c possibly produced or endorsed by any of them, can depend on whether the two users have been informed or not about c through their corresponding homepages. Clearly, if a grouping of users into communities was available, the platform administrators would likely drive the attention of users towards contents that are produced by members of the same community, according to a homophily effect. On the other hand, any user may also want to seek for relevant contents and similar users outside the boundary of her community, which would also have the effect of mitigating information-bubble issues that may arise inside each community. In this regard, it would be strategic for the administrators to know which links to users and associated contents are worthy to be recommended within other users' pages, in order to incentivate the overall interactions across the platform. Intuitively, this can be translated into the problem of finding a clustering of the set of users such that for each user u in any cluster C_k, a link towards contents produced by the other users in C_k will be posted in the u's profile-page.

Another application scenario corresponds to a team formation task for a collaborative system, like Wikipedia, where users should be grouped into teams to contribute in the editing of different parts of a Wikipedia page. In this context, the likelihood of collaboration between any pair of users will vary in relation to their assignment to the same team. The goal becomes to partition the set of users into teams in order to maximize the total collaboration. The greater the overall collaboration is, the higher the contamination will be, and the probability of successfully accomplishing the task will increase.

Contributions. To the best of our knowledge, we are the first to address the problem of optimizing the overall interaction among a set of entities in a probabilistic graph, subject to the cluster memberships of the entities. In particular, our main contributions are:

- We define the Max-Interaction-Clustering and Min-Inter- action-Loss-Clustering problems for graph entities whose interaction patterns depend on their cluster memberships (Section 6.3). We show that both problems are special instances of the well-studied correlation-clustering framework [4], and we delve into their theoretical properties and complexities.

- Although the two problems are equivalent in terms of optimality, we focus on the minimization problem, as it enables the use of more practical yet efficient algorithms inspired by correlation-clustering theory. To this purpose, we devise both approximation algorithms and heuristics for the Min-Interaction-loss-Clustering problem (Section 6.4).

- Experimental evaluation of our algorithms, on both synthetic and real network datasets, has shown evidence of their meaningfulness as well as superiority with respect to competing methods, both in terms of effectiveness and efficiency (Section 6.5).

The remainder of the chapter is organized as follows. Section 6.2 briefly overviews related work. Section 6.3 introduces our problem formulations, and Section 6.4 describes our developed approximation algorithms and heuristics. In Sections 6.5 we report our experimental evaluation. Section 6.6 concludes the chapter.

6.2 Related Work

Clustering uncertain graphs. The problem we tackle in this work is close to that of clustering uncertain graphs, which is to cluster vertices of a graph whose edges are assigned a probability of existence, according to some (possible-world) semantics [83, 75, 40, 56, 17, 81, 57]. A major difference is that our problem is more general than clustering uncertain graphs since the probabilities of interaction are affected by cluster memberships, which poses additional challenges that we address in this work. Further differences are that (*i*) the classic uncertain-graph model is a special case of interaction graph we consider in this work (where the probability distributions of interaction are binary), and (*ii*) existing methods for clustering uncertain graphs aim to maximize the intra-cluster connectivity and minimize the inter-cluster connectivity, whereas we seek clusterings such that both types of connectivity are maximized.

Community detection in signed graphs. In signed graphs, which have positive and negative signs as a property on the edges (e.g., trust vs. distrust relations), the problem of community detection is to produce a structure whereby many positive (resp. negative) links are observed within (resp. between) communities. For example, [121] considers a directed graph and solves the above problem by optimizing a new notion of modularity that combines linearly the contribution of positive and negative weights, and extending the Potts Model to incorporate negative links. The method in [47] also introduces a generalization of modularity that is able to deal with both positive and negative weights. That definition of modularity is a special case of the one defined in [121], as it can be obtained from the general definition of [121] by properly setting the parameters that control the balance between the importance of present and absent (positive and negative) edges within a community. [32] reformulates the Map Equation to measure the quality of partitions, known as Minimum Description Length (MDL), and extends Constant Potts Model (CPM) to collect a spectrum of partitions from highly simplified to detailed ones, by varying its parameter λ from zero to one. Based on these extensions, the community detection is carried out by minimizing MDL on λ-spectrum.

The aforementioned methods will be considered in our experiments (cf. Section 6.5), given the similarity in the requirements of within- and across-cluster interactions. Nonetheless, we remark that edges in a signed graph can have either positive or negative weight, while in our setting each edge is assigned a probability distribution of the interaction strength between two vertices.

Correlation clustering. Originally introduced by Bansal *et al.* [4], correlation clustering is, given a complete signed graph G where every pair of vertices is labeled either as positive or negative, partition the vertices of G so as to *either* minimize the number of negative pairs within the same cluster plus the positive pairs across different clusters, *or* maximize the positive pairs within the same cluster plus the negative pairs across different clusters. In Sections 6.3.1–6.3.2, we shall discuss the profound relation with our problem formulations.

6.3 Problem definition

We are given a set of users who *interact* with each other, where "interaction" is meant here as referring to any, symmetric or reciprocated, relation between two users (e.g., commenting posts of each other, collaborating for a task, etc.). We assume that the strength of interaction between any two users is represented by a nonnegative real value, however the exact interaction strength is not known beforehand; rather, a set of possible strengths are given, each one being assigned a probability of corresponding to the actual strength. This scenario is here modeled by a probabilistic interaction graph, or simply *interaction graph*, we define as a key notion in this work.

Definition 8 (Probabilistic interaction graph) *A* probabilistic interaction graph *is a triple $\mathcal{G} = (V, E, P)$, with V set of vertices, $E \subseteq V \times V$ set of undirected edges, and $P = \{p_{uv}\}_{(u,v)\in E}$ set of probability distributions, each one defined on a domain $\mathcal{D}(p_{uv}) \subseteq \mathbb{R}_0^+$. For all $(u, v) \in E$ and all $x \in \mathcal{D}(p_{uv})$, $p_{uv}(x)$ is the probability that the strength of the interaction between u and v is equal to x. For any $(u, v) \notin E$, $p_{uv}(0) = 1$ and $p_{uv}(x) = 0$ for any $x \neq 0$.*

Given an interaction graph $\mathcal{G}=(V, E, P)$, a set $\{G=(V, E, w_G)\}_{G\sqsubseteq\mathcal{G}}$ of *deterministic graphs* can be derived as instances of $\mathcal{G}$ — following the literature on uncertain data/graphs [71], these are also called *worlds*. Every instance G that can be derived from $\mathcal{G}$, here denoted as $G \sqsubseteq \mathcal{G}$, is a weighted graph that is defined over the same sets V, E of $\mathcal{G}$, and whose weighting function $w_G : E \to \mathbb{R}_0^+$ assigns a weight to every edge $(u, v) \in E$ so that $w_G(u, v)$ is sampled from $p_{uv} \in P$, i.e., $w_G(u, v) \in \mathcal{D}(p_{uv})$. Note that, as for any $(u, v) \in E : \mathcal{D}(p_{uv}) \subseteq \mathbb{R}_0^+$, a possible world $G \sqsubseteq \mathcal{G}$ may contain edges (u, v) from $\mathcal{G}$ that do not exist in G, i.e., $w_G(u, v) = 0$.

Assuming independence between probability distributions — as usual in the literature on uncertain graphs [83, 75, 8, 69, 101, 17, 70, 71] — the probability of a possible world $G = (V, E, w_G) \sqsubseteq \mathcal{G}$ is:

$$\Pr(G) = \prod_{(u,v)\in E} p_{uv}(w_G(u, v)). \tag{6.1}$$

A key aspect in our study is that an interaction graph $\mathcal{G} = (V, E, P)$ can be provided in terms of two probabilistic graphs, say $\mathcal{G}^+$ and $\mathcal{G}^-$, both defined over V and E, such that the one accounts for interactions within the same clusters of vertices, and the other one for interactions between different clusters of vertices. To this end, let $\mathcal{C} : V \to \mathbb{N}$ denote an injective function that expresses the *cluster-membership* for the vertices in V. Conditionally to the cluster-memberships for the vertices in $\mathcal{G}^+$ and $\mathcal{G}^-$, the two graphs can be "merged" into a single interaction graph we define as follows.

Definition 9 (Clustering-conditional interaction graph) *Let $\mathcal{G}^+ = (V, E, P^+)$ and $\mathcal{G}^- = (V, E, P^-)$ be two interaction graphs defined over the same vertex- and edge-sets, and $\mathcal{C} : V \to \mathbb{N}$ be the cluster-membership function for the vertices. A* clustering-conditional interaction graph *is defined as $\mathcal{G}_\mathcal{C} = (V, E, P_\mathcal{C})$, such that each edge (u, v) of $\mathcal{G}_\mathcal{C}$ is assigned the corresponding probability distribution $p_{uv} \in P^+$ from $\mathcal{G}^+$, if u and v belongs to the same cluster according to $\mathcal{C}$, otherwise the distribution $p_{uv} \in P^-$ from $\mathcal{G}^-$, i.e., $P_\mathcal{C} = \{p_{uv} \in P^+ \mid \mathcal{C}(u) = \mathcal{C}(v)\} \cup \{p_{uv} \in P^- \mid \mathcal{C}(u) \neq \mathcal{C}(v)\}$.*

Upon the above definition, we focus on two optimization problems with complementary yet conceptually equivalent goals, that is, to cluster V so as to either (i) *maximize the expected overall interaction*, or (ii) *minimize the expected overall interaction loss*. These goals lead to two different formulations, which we state in detail next.

6.3.1 Maximizing interaction

Let the overall interaction $f(G)$ of a deterministic graph $G = (V, E, w_G)$ be the sum of the interactions on all edges, i.e.,

$$f(G) = \sum_{(u,v)\in E} w_G(u, v) \tag{6.2}$$

As a consequence, the expected overall interaction $\bar{f}(\mathcal{G})$ of an interaction graph $\mathcal{G} = (V, E, P)$ is defined as:

$$\bar{f}(\mathcal{G}) \;=\; \mathop{\mathbb{E}}_{G\sqsubseteq\mathcal{G}}[f(G)] \;=\; \sum_{G\sqsubseteq\mathcal{G}} f(G)\Pr(G), \tag{6.3}$$

where $\Pr(G)$ is the probability of observing G (Equation (6.1)).

The first problem we tackle in this work is as follows:

Problem 4 (Max-Interaction-Clustering) *Given two interaction graphs $\mathcal{G}^+ = (V, E, P^+)$ and $\mathcal{G}^- = (V, E, P^-)$ sharing the same vertex set and edge set, find a clustering $\mathcal{C}^* : V \to \mathbb{N}$ that maximizes the expected overall interaction of the clustering-conditional interaction graph, i.e.,*

$$\mathcal{C}^* = \operatorname*{argmax}_{\mathcal{C}} \bar{f}(\mathcal{G}_{\mathcal{C}}). \tag{6.4}$$

Connection with Correlation Clustering. Since its introduction, correlation clustering has received a great deal of attention, with a focus on various aspects, such as theoretical results, algorithms, and problem generalizations/variants [100]. To date, the most general formulation of correlation clustering [2] takes as input a set Ω of objects, and two nonnegative weights $\omega^+_{xy}, \omega^-_{xy}$ for every unordered pair $x, y \in \Omega$ of objects. The weights assigned to an object pair (x, y) intuitively express the advantage of putting x and y in the same cluster (ω^+_{xy}) or in separate clusters (ω^-_{xy}). The objective is to partition Ω so as to either *minimize* the sum of the negative weights between objects within the same cluster plus the sum of the positive weights between objects in separate clusters (Min-CC), or *maximize* the sum of the positive weights between objects within the same cluster plus the sum of the negative weights between objects in separate clusters (Max-CC):

Problem 5 (Min-CC [2]) *Given a set Ω of objects, and nonnegative weights $\omega^+_{xy}, \omega^-_{xy} \in \mathbb{R}^+_0$ for all unordered pairs $x, y \in \Omega$ of objects, find a clustering $\mathcal{C} : \Omega \to \mathbb{N}^+$ that minimizes*

$$\sum_{\substack{x,y\in\Omega,\\ \mathcal{C}(x)=\mathcal{C}(y)}} \omega^-_{xy} \;+\; \sum_{\substack{x,y\in\Omega,\\ \mathcal{C}(x)\neq\mathcal{C}(y)}} \omega^+_{xy}. \tag{6.5}$$

Problem 6 (Max-CC [2]) *Given a set Ω of objects, and nonnegative weights $\omega^+_{xy}, \omega^-_{xy} \in \mathbb{R}^+_0$ for all unordered pairs $x, y \in \Omega$ of objects, find a clustering $\mathcal{C} : \Omega \to \mathbb{N}^+$ that maximizes*

$$\sum_{\substack{x,y\in\Omega,\\ \mathcal{C}(x)=\mathcal{C}(y)}} \omega^+_{xy} \;+\; \sum_{\substack{x,y\in\Omega,\\ \mathcal{C}(x)\neq\mathcal{C}(y)}} \omega^-_{xy}. \tag{6.6}$$

Min-CC and Max-CC are equivalent in terms of optimality and complexity class (both **NP**-hard), but have different approximation-guarantee properties, with the latter being easier in this regard.

As a noteworthy result, our Max-Interaction-Clustering problem can be shown to be an instance of Max-CC:

Theorem 1 *Given two interaction graphs $\mathcal{G}^+ = (V, E, P^+)$ and $\mathcal{G}^- = (V, E, P^-)$, solving* Max-Interaction-Clustering *on input $\langle \mathcal{G}^+, \mathcal{G}^- \rangle$ is equivalent to solving* Max-CC *by setting $\Omega = V$, $\omega^+_{uv} = \mathbb{E}[p^+_{uv}]$, $\omega^-_{uv} = \mathbb{E}[p^-_{uv}]$, for all $(u, v) \in E$, and $\omega^+_{uv} = \omega^-_{uv} = 0$, for all $(u, v) \in \overline{E}$ (where $\overline{E} = V \times V \setminus E$).*

Proof. The objective function of Max-Interaction-Clustering (Equation (6.4)) can be rearranged as follows:

$$
\begin{aligned}
\bar{f}(\mathcal{G}_{\mathcal{C}}) &= \mathop{\mathbb{E}}_{G \sqsubseteq \mathcal{G}_{\mathcal{C}}}[f(G)] = \sum_{G \sqsubseteq \mathcal{G}_{\mathcal{C}}} f(G) \Pr(G) = \sum_{G \sqsubseteq \mathcal{G}_{\mathcal{C}}} \left(\sum_{(u,v) \in E} w_G(u, v) \right) \Pr(G) = \\
&= \sum_{G \sqsubseteq \mathcal{G}_{\mathcal{C}}} \left(\sum_{\substack{(u,v) \in E, \\ \mathcal{C}(u) = \mathcal{C}(v)}} w_G(u, v) \right) \Pr(G) \; + \sum_{G \sqsubseteq \mathcal{G}_{\mathcal{C}}} \left(\sum_{\substack{(u,v) \in E, \\ \mathcal{C}(u) \neq \mathcal{C}(v)}} w_G(u, v) \right) \Pr(G) = \\
&= \sum_{\substack{(u,v) \in E, \\ \mathcal{C}(u) = \mathcal{C}(v)}} \underbrace{\left(\sum_{G \sqsubseteq \mathcal{G}_{\mathcal{C}}} w_G(u, v) \Pr(G) \right)}_{= \mathbb{E}[p^+_{uv}]} \; + \sum_{\substack{(u,v) \in E, \\ \mathcal{C}(u) \neq \mathcal{C}(v)}} \underbrace{\left(\sum_{G \sqsubseteq \mathcal{G}_{\mathcal{C}}} w_G(u, v) \Pr(G) \right)}_{= \mathbb{E}[p^-_{uv}]} = \\
&= \sum_{\substack{(u,v) \in E, \\ \mathcal{C}(u) = \mathcal{C}(v)}} \mathbb{E}[p^+_{uv}] \; + \sum_{\substack{(u,v) \in E, \\ \mathcal{C}(u) \neq \mathcal{C}(v)}} \mathbb{E}[p^-_{uv}] \; + \underbrace{\sum_{\substack{(u,v) \in \overline{E}, \\ \mathcal{C}(u) = \mathcal{C}(v)}} \mathbb{E}[p^+_{uv}] \; + \sum_{\substack{(u,v) \in \overline{E}, \\ \mathcal{C}(u) \neq \mathcal{C}(v)}} \mathbb{E}[p^-_{uv}]}_{= 0} = \\
&= \sum_{\substack{x,y \in \Omega, \\ \mathcal{C}(x) = \mathcal{C}(y)}} \omega^+_{xy} \; + \sum_{\substack{x,y \in \Omega, \\ \mathcal{C}(x) \neq \mathcal{C}(y)}} \omega^-_{xy},
\end{aligned}
$$

which corresponds to the objective function of Max-CC. ■

The connection with correlation clustering also unveils the **NP**-hardness of Max-Interaction-Clustering:

Theorem 2 Max-Interaction-Clustering *is* **NP***-hard.*

Proof. (Sketch) The fact of being a special case of Max-CC clearly does not necessarily imply that Max-Interaction-Clustering is **NP**-hard too. However, **NP**-hardness can be shown by reducing from the basic Bansal *et al.*'s variant of correlation clustering on general graphs [4], which corresponds to Max-CC when $(\omega^+_{xy}, \omega^-_{xy}) \in \{(1, 0), (0, 0), (0, 1)\}$. Even such a simpler variant is **NP**-hard, and can easily be observed to correspond to Max-Interaction-Clustering when it holds that $\forall (u, v) \in E : p^+_{uv}(\omega^+_{uv}) = p^-_{uv}(\omega^-_{uv}) = 1$. ■

6.3.2 Minimizing interaction loss

Given two interaction graphs $\mathcal{G}^+ = (V, E, P^+)$, $\mathcal{G}^- = (V, E, P^-)$, let $M(\mathcal{G}^+, \mathcal{G}^-) \in \mathbb{R}^+$ be a constant larger than the maximum interaction strength in $\mathcal{G}^+$ and $\mathcal{G}^-$, i.e.,

$M(\mathcal{G}^+,\mathcal{G}^-) > \max\{x \in \mathcal{D}(p_{uv}) \mid p_{uv} \in P^+ \cup P^-, (u,v) \in E\}$. Based on $M(\mathcal{G}^+,\mathcal{G}^-)$, let the overall interaction loss $\ell(G)$ of a deterministic graph $G = (V, E, w_G)$ be:

$$\begin{aligned}\ell(G) &= \sum_{(u,v)\in E}(M(\mathcal{G}^+,\mathcal{G}^-) - w_G(u,v)) \;+\; |\overline{E}|M(\mathcal{G}^+,\mathcal{G}^-) = \\ &= M(\mathcal{G}^+,\mathcal{G}^-)\binom{|V|}{2} - \sum_{(u,v)\in E} w_G(u,v),\end{aligned} \tag{6.7}$$

and the expected overall interaction loss $\bar{\ell}(\mathcal{G})$ of an interaction graph $\mathcal{G} = (V, E, P)$ be:

$$\bar{\ell}(\mathcal{G}) \;=\; \mathop{\mathbb{E}}_{G\sqsubseteq\mathcal{G}}[\ell(G)] \;=\; \sum_{G\sqsubseteq\mathcal{G}} \ell(G)\Pr(G). \tag{6.8}$$

The minimization version of the problem we tackle in this work is:

Problem 7 (Min-Interaction-loss-Clustering) *Given two interaction graphs $\mathcal{G}^+ = (V, E, P^+)$ and $\mathcal{G}^- = (V, E, P^-)$ sharing the same vertex set and edge set, find a clustering $\mathcal{C}^* : V \to \mathbb{N}^+$ so that*

$$\mathcal{C}^* = \arg\min_{\mathcal{C}} \bar{\ell}(\mathcal{G}_{\mathcal{C}}). \tag{6.9}$$

Such a minimization formulation is equivalent to the maximization one in terms of optimality (and complexity class), since:

$$\begin{aligned}\bar{\ell}(\mathcal{G}) &= \sum_{G\sqsubseteq\mathcal{G}}\left(M(\mathcal{G}^+,\mathcal{G}^-)\binom{|V|}{2} - \sum_{(u,v)\in E} w_G(u,v)\right)\Pr(G) = \\ &= \underbrace{-\sum_{G\sqsubseteq\mathcal{G}}\left(\sum_{(u,v)\in E} w_G(u,v)\right)\Pr(G)}_{=\,-\bar{f}(\mathcal{G})} \;+\; \underbrace{M(\mathcal{G}^+,\mathcal{G}^-)\binom{|V|}{2}}_{\text{constant} > 0}.\end{aligned} \tag{6.10}$$

The result in Theorem 3 immediately follows:

Theorem 3 Min-Interaction-loss-Clustering *is* **NP**-*hard.*

Connection with Correlation Clustering. Similarly to the maximization version, our Min-Interaction-loss-Clustering can be shown to be an instance of Min-CC:

Theorem 4 *Given two interaction graphs $\mathcal{G}^+ = (V, E, P^+)$ and $\mathcal{G}^- = (V, E, P^-)$, solving* Min-Interaction-loss-Clustering *on input $\langle\mathcal{G}^+, \mathcal{G}^-\rangle$ is equivalent to solving* Min-CC *by setting $\Omega = V$, $\omega^+_{uv} = M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]$, $\omega^-_{uv} = M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}]$, for all $(u, v) \in E$, and $\omega^+_{uv} = \omega^-_{uv} = M(\mathcal{G}^+,\mathcal{G}^-)$, for all $(u, v) \in \overline{E}$ (where $\overline{E} = \binom{V}{2} \setminus E$).*

Proof. Let us define the *discounted interaction loss* $\mathcal{G}_{\mathcal{C}}$ which discards the loss contribution due to non-linked pairs, as follows:

$$\mathcal{L}(\mathcal{G}_{\mathcal{C}}) = \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \underbrace{\left(\sum_{G\sqsubseteq\mathcal{G}_{\mathcal{C}}}(M(\mathcal{G}^+,\mathcal{G}^-) - w_G(u,v))\right)\Pr(G)}_{=\,M(\mathcal{G}^+,\mathcal{G}^-)\;-\;\mathbb{E}[p^+_{uv}]} +$$

$$+\sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \underbrace{\left(\sum_{G\sqsubseteq\mathcal{G}_{\mathcal{C}}} (M(\mathcal{G}^+,\mathcal{G}^-)-w_G(u,v))\right)}_{=\, M(\mathcal{G}^+,\mathcal{G}^-)\, -\, \mathbb{E}[p^-_{uv}]} \Pr(G)\,. \tag{6.11}$$

With similar arguments as Theorem 1, it can be shown that:

$$\bar{\ell}(\mathcal{G}_{\mathcal{C}}) = \mathcal{L}(\mathcal{G}_{\mathcal{C}}) + \ |\overline{E}|M(\mathcal{G}^+,\mathcal{G}^-) = \sum_{\substack{x,y\in\Omega,\\ \mathcal{C}(x)=\mathcal{C}(y)}} \omega^-_{xy} \ + \sum_{\substack{x,y\in\Omega,\\ \mathcal{C}(x)\neq\mathcal{C}(y)}} \omega^+_{xy},$$

which corresponds to the objective function of Min-CC. ■

6.4 Algorithms

The connection with correlation clustering forms the basis of algorithm design for our problems. Specifically, our main idea here is to investigate whether the considerable amount of work on correlation-clustering algorithms with proved quality guarantees can be fruitfully exploited in our setting too.

6.4.1 Algorithms for Max-Interaction-Clustering.

In the following we show that the state-of-the-art (constant-factor) approximation algorithms designed for Max-CC keep their guarantees on Max-Interaction-Clustering too. Theorem 1 states that Max-Interaction-Clustering is an instance of Max-CC. Specifically, as the various p^+_{uv}, p^-_{uv} are general, Max-Interaction-Clustering is an instance of Max-CC with weights $(\omega^+_{uv},\omega^+_{uv}) \in \mathbb{R}^+_0 \times \mathbb{R}^+_0$, for all $u,v \in V$. Such a variant of Max-CC is not studied in the literature. The closest variant for which theoretical results have been derived is the one where, for every pair (u,v) of vertices, at most one between ω^+_{uv} and ω^-_{uv} is non-zero, i.e., the variant where $(\omega^+_{uv},\omega^-_{uv}) \in \{(\omega',0),(0,\omega'')\}_{\omega',\omega''\in\mathbb{R}^+_0}$, $\forall u,v \in V$ [24, 113]. For this variant, Swamy [113] devises a 0.7666-approximation algorithm based on a semidefinite-programming, and a further, more practical 0.75-approximation algorithm. Next we show that Swamy's approximation result carries over to the Max-CC variant underlying our Max-Interaction-Clustering problem.

Given two interaction graphs $\mathcal{G}^+ = (V,E,P^+)$ and $\mathcal{G}^- = (V,E,P^-)$, for all $(u,v) \in E$, let $\bar{p}_{uv}$, $\hat{\tau}^+_{uv}$ and $\hat{\tau}^-_{uv}$ be defined as:

$$\bar{p}_{uv} = \min\{\mathbb{E}[p^+_{uv}],\mathbb{E}[p^-_{uv}]\}, \hat{\tau}^+_{uv} = \mathbb{E}[p^+_{uv}] - \bar{p}_{uv}, \hat{\tau}^-_{uv} = \mathbb{E}[p^-_{uv}] - \bar{p}_{uv}.$$

Thus, by definition, $(\hat{\tau}^+_{uv},\hat{\tau}^-_{uv}) \in \{(\omega',0),(0,\omega'')\}_{\omega',\omega''\in\mathbb{R}^+_0}$, $\forall u,v \in V$, like in Swamy's setting. Moreover, the objective function $\bar{f}(\cdot)$ of Max-Interaction-Clustering

can be rewritten as:

$$\begin{aligned}
\bar{f}(\mathcal{G}_{\mathcal{C}}) &= \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \mathbb{E}[p^+_{uv}] + \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \mathbb{E}[p^-_{uv}] \qquad \{\text{Theorem 1}\} \\
&= \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \left(\hat{\tau}^+_{uv} + \bar{p}_{uv}\right) + \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \left(\hat{\tau}^-_{uv} + \bar{p}_{uv}\right) \\
&= \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \hat{\tau}^+_{uv} + \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \hat{\tau}^-_{uv} + \underbrace{\sum_{(u,v)\in E} \bar{p}_{uv}}_{H(\mathcal{G}^+,\mathcal{G}^-)} \\
&= \underbrace{\sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \hat{\tau}^+_{uv} + \sum_{\substack{(u,v)\in E,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \hat{\tau}^-_{uv}}_{:=\, h(\mathcal{G}_{\mathcal{C}})} + \underbrace{H(\mathcal{G}^+,\mathcal{G}^-)}_{\text{constant} \geq 0}. \qquad (6.12)
\end{aligned}$$

As a result, Max-Interaction-Clustering's objective function $\bar{f}(\cdot)$ corresponds to the sum of the objective function of Max-CC (where the weights assigned to every pair (u, v) of vertices are $\hat{\tau}^+_{uv}$ and $\hat{\tau}^-_{uv}$) plus a nonnegative constant. Hence, Max-Interaction-Clustering and Max-CC are equivalent in terms of optimal value. Specifically, let $I_1 = \langle \mathcal{G}^+, \mathcal{G}^- \rangle$ be an instance of Max-Interaction-Clustering, and $I_2 = \langle V, \{\hat{\tau}^+_{uv}\}_{u,v\in V}, \{\hat{\tau}^-_{uv}\}_{u,v\in V} \rangle$ be an instance of Max-CC derived from I_1 by employing the weights defined above. Let also $\mathcal{C}^*_{\bar{f}}$ and $\mathcal{C}^*_h$ be the optimal clusterings for the I_1 instance according to the $\bar{f}(\cdot)$ and $h(\cdot)$ functions, respectively. Finally, let $\widetilde{\mathcal{C}}$ denote the clustering yielded by the given α-approximation algorithm for Max-CC on input I_2 (e.g., aforementioned factor-0.7666 Swamy's algorithm [113]). By definition of approximation algorithm, we know that, for every input: $h(\mathcal{G}_{\widetilde{\mathcal{C}}}) \geq \alpha \times h(\mathcal{G}_{\mathcal{C}^*_h})$, where $\alpha \leq 1$. Therefore, it holds that:

$$h(\mathcal{G}_{\widetilde{\mathcal{C}}}) \geq \alpha \times h(\mathcal{G}_{\mathcal{C}^*_h}) \Leftrightarrow h(\mathcal{G}_{\widetilde{\mathcal{C}}}) + H(\mathcal{G}^+,\mathcal{G}^-) \geq \alpha \times h(\mathcal{G}_{\mathcal{C}^*_h}) + H(\mathcal{G}^+,\mathcal{G}^-)$$

$$\Rightarrow h(\mathcal{G}_{\widetilde{\mathcal{C}}}) + H(\mathcal{G}^+,\mathcal{G}^-) \geq \alpha \times \left(h(\mathcal{G}_{\mathcal{C}^*_h}) + H(\mathcal{G}^+,\mathcal{G}^-)\right)$$

$$\Leftrightarrow \bar{f}(\mathcal{G}_{\widetilde{\mathcal{C}}}) \geq \alpha \times \bar{f}(\mathcal{G}_{\mathcal{C}^*_g}) \Leftrightarrow \bar{f}(\mathcal{G}_{\widetilde{\mathcal{C}}}) \geq \alpha \times \bar{f}(\mathcal{G}_{\mathcal{C}^*_{\bar{f}}}).$$

However, the state-of-the-art approximation algorithms for Max-CC (on general, weighted graphs) correspond to the semidefinite-programming-based ones devised by Swamy [113]. Those algorithms are inefficient and, more importantly, rather impractical, since they are not able to output more than a fixed number of clusters (i.e., six). This is a showstopper in our context, as we are interested in algorithms that are effective and theoretically solid, yet capable of handling large-scale inputs and providing outputs whose quality is recognizable in practice too, not only theoretically.

6.4.2 Algorithms for Min-Interaction-loss-Clustering.

More interesting results instead hold for the minimization version of our problem. Specifically, we derive a rearrangement of Min-Interaction-loss-Clustering's objective function, which unveils that, under mild conditions, the algorithms designed for Min-CC preserve their approximation properties when applied (with minor modifications) to our problem. This is particularly appealing, as Min-CC admits approximation algorithms that do not suffer from the limitations of the maximization

counterpart, i.e., they are efficient and capable of finding general clusterings [2]. For this reason, our algorithm-design process focuses on the minimization version of our problem, and the remainder of this section provides the details of this process.

An approximation algorithm

Ailon *et al.*'s KwikCluster [2] is a well-established algorithm for Min-CC. It iteratively picks an object x (uniformly at random among the unclustered objects), and builds a cluster comprised of x and all unclustered objects y such that $\omega^+_{xy} > \omega^-_{xy}$. KwikCluster is particularly appealing, due to its (*i*) constant-factor (expected) approximation guarantees (i.e., factor-5 or factor-2, depending on the conditions satisfied by the input weights, cf. later in this section), (*ii*) efficiency (i.e., it takes linear time in the number of edges of the input graph), and (*iii*) easiness of implementation. All these aspects make it generally preferable to other algorithms (such as the one by Charikar *et al.* [19]) that have slightly better approximation guarantees, but are less efficient and more difficult to implement.

Theoretical basis. With the above motivations, we investigate possible exploitation of KwikCluster for our Min-Interaction-loss-Clustering, and the theoretical basis for which its appealing features are still valid. In this regard, a major remark is that the constant-factor approximation guarantees of KwikCluster hold for input graphs whose weights on every edge satisfy the probability constraint, i.e., $\omega^+_{xy} + \omega^-_{xy} = 1$, for all $x, y \in \Omega$. Although this is a requirement that does not generally hold for the input to Min-Interaction-loss-Clustering, in the following we show that the objective function of our problem can be manipulated in such a way that the probability constraint *is satisfied under mild conditions*.

Given two interaction graphs $\mathcal{G}^+ = (V, E, P^+)$, $\mathcal{G}^- = (V, E, P^-)$, for every unordered pair $u, v \in V$, let σ_{uv}, τ^+_{uv}, τ^-_{uv} be:

$$\sigma_{uv} = M(\mathcal{G}^+,\mathcal{G}^-) - \left(\mathbb{E}[p^+_{uv}] + \mathbb{E}[p^-_{uv}]\right), \tag{6.13}$$

$$\tau^+_{uv} = \frac{1}{M(\mathcal{G}^+,\mathcal{G}^-)}\left(\mathbb{E}[p^+_{uv}] + \frac{\sigma_{uv}}{2}\right), \quad \tau^-_{uv} = \frac{1}{M(\mathcal{G}^+,\mathcal{G}^-)}\left(\mathbb{E}[p^-_{uv}] + \frac{\sigma_{uv}}{2}\right). \tag{6.14}$$

It is easy to see that $\sigma_{uv} \in [-M(\mathcal{G}^+,\mathcal{G}^-), M(\mathcal{G}^+,\mathcal{G}^-)]$, and τ^+_{uv}, τ^-_{uv} satisfy the above probability constraint, as stated in Lemma 1.

Lemma 1 *It holds that* $\tau^+_{uv}, \tau^-_{uv} \geq 0$ *and* $\tau^+_{uv} + \tau^-_{uv} = 1$, $\forall u, v \in V$.

Function $\bar{\ell}(\cdot)$ of Min-Interaction-loss-Clustering can be rewritten in terms of τ^+_{uv}, τ^-_{uv}, as follows

Lemma 2 *Given two interaction graphs* $\mathcal{G}^+ = (V, E, P^+)$ *and* $\mathcal{G}^- = (V, E, P^-)$, *and a clustering* $\mathcal{C} : V \to \mathbb{N}^+$, *let*

$$g(\mathcal{G}_\mathcal{C}) = \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \tau^-_{uv} + \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \tau^+_{uv}, \quad K(\mathcal{G}^+,\mathcal{G}^-) = \sum_{u,v\in V} \frac{\sigma_{uv}}{2}. \tag{6.15}$$

It holds that $\bar{\ell}(\mathcal{G}_\mathcal{C}) = M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_\mathcal{C}) + K(\mathcal{G}^+,\mathcal{G}^-)$.

Proof. Assuming w.l.o.g. $\mathbb{E}[p^+_{uv}] = \mathbb{E}[p^-_{uv}] = 0$, for all $(u, v) \notin E$, it holds that:

$$
\begin{aligned}
\bar{\ell}(\mathcal{G}_{\mathcal{C}}) =& \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \big(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}]\big) + \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \big(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]\big) \quad \{\text{Theorem 4}\} \\
=& \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \left(M(\mathcal{G}^+,\mathcal{G}^-) - M(\mathcal{G}^+,\mathcal{G}^-)\tau^+_{uv} + \frac{\sigma_{uv}}{2}\right) + \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \left(M(\mathcal{G}^+,\mathcal{G}^-) - M(\mathcal{G}^+,\mathcal{G}^-)\tau^-_{uv} + \frac{\sigma_{uv}}{2}\right) \\
=& M(\mathcal{G}^+,\mathcal{G}^-) \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \frac{1}{M(\mathcal{G}^+,\mathcal{G}^-)} \left(M(\mathcal{G}^+,\mathcal{G}^-) - M(\mathcal{G}^+,\mathcal{G}^-)\tau^+_{uv} + \frac{\sigma_{uv}}{2}\right) + \\
&+ M(\mathcal{G}^+,\mathcal{G}^-) \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \frac{1}{M(\mathcal{G}^+,\mathcal{G}^-)} \left(M(\mathcal{G}^+,\mathcal{G}^-) - M(\mathcal{G}^+,\mathcal{G}^-)\tau^-_{uv} + \frac{\sigma_{uv}}{2}\right) \\
=& M(\mathcal{G}^+,\mathcal{G}^-) \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \left(1 - \tau^+_{uv} + \frac{\sigma_{uv}}{2M(\mathcal{G}^+,\mathcal{G}^-)}\right) + M(\mathcal{G}^+,\mathcal{G}^-) \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \left(1 - \tau^-_{uv} + \frac{\sigma_{uv}}{2M(\mathcal{G}^+,\mathcal{G}^-)}\right) \\
=& M(\mathcal{G}^+,\mathcal{G}^-) \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \underbrace{\big(1-\tau^+_{uv}\big)}_{=\,\tau^-_{uv}} + M(\mathcal{G}^+,\mathcal{G}^-) \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \underbrace{\big(1-\tau^-_{uv}\big)}_{=\,\tau^+_{uv}} + \underbrace{\sum_{u,v\in V} \frac{\sigma_{uv}}{2}}_{=\,K(\mathcal{G}^+,\mathcal{G}^-)} \\
=& M(\mathcal{G}^+,\mathcal{G}^-) \underbrace{\left(\sum_{\substack{u,v\in V,\\ \mathcal{C}(u)=\mathcal{C}(v)}} \tau^-_{uv} + \sum_{\substack{u,v\in V,\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} \tau^+_{uv} \right)}_{=\,g(\mathcal{G}_{\mathcal{C}})} + K(\mathcal{G}^+,\mathcal{G}^-). \qquad \blacksquare
\end{aligned}
$$

Moreover, in Lemma 3, we state that constant-factor approximation guarantees for Min-CC carry over to our problem.

Lemma 3 *If $K(\mathcal{G}^+,\mathcal{G}^-) \geq 0$ (Equation (6.15)), then any α-approximation algorithm for* Min-CC *is an α-approximation algorithm for* Min-Interaction-loss-Clustering*, for every constant $\alpha > 1$.*

Proof. Let $I_1 = \langle \mathcal{G}^+, \mathcal{G}^- \rangle$ be an instance of Min-Interaction-loss-Clustering, and $I_2 = \langle V, \{\tau^+_{uv}\}_{u,v\in V}, \{\tau^-_{uv}\}_{u,v\in V} \rangle$ be an instance of Min-CC derived from I_1 by employing the weights defined in Equation (6.14). Let also $\mathcal{C}^*_{\bar{\ell}}$ and $\mathcal{C}^*_g$ be the optimal clusterings for the I_1 instance according to the $\bar{\ell}(\cdot)$ and $g(\cdot)$ functions, respectively. Finally, let $\widetilde{\mathcal{C}}$ denote the clustering yielded by the given α-approximation algorithm for Min-CC on input I_2.

The goal is to demonstrate that, for every I_1, I_2, $\bar{\ell}(\mathcal{G}_{\widetilde{\mathcal{C}}}) \leq \alpha \times \bar{\ell}(\mathcal{G}_{\mathcal{C}^*_{\bar{\ell}}})$. First, it is straightforward to note that $g(\cdot)$ corresponds to Min-CC's objective function. By

Algorithm 8 MIL

Input: Interaction graphs $\mathcal{G}^+ = (V, E, P^+)$, $\mathcal{G}^- = (V, E, P^-)$
Output: A clustering $\mathcal{C}$ of V
1: compute τ^+_{uv}, τ^-_{uv} for all $(u, v) \in E$ {Equation (6.14)}
2: $\mathcal{C} \leftarrow \emptyset$, $V' \leftarrow V$
3: **while** $V' \neq \emptyset$ **do**
4: pick a pivot vertex $u \in V'$ uniformly at random
5: $\mathcal{C}_u \leftarrow \{u\} \cup \{v \in V' \mid (u, v) \in E, \tau^+_{uv} > \tau^-_{uv}\}$
6: add cluster $\mathcal{C}_u$ to $\mathcal{C}$ and remove all vertices in C_u from V'
7: **end while**

definition of approximation algorithm, $g(\mathcal{G}_{\widetilde{\mathcal{C}}}) \leq \alpha \times g(\mathcal{G}_{\mathcal{C}^*_g})$, therefore it holds that:

$$
\begin{aligned}
& g(\mathcal{G}_{\widetilde{\mathcal{C}}}) \leq \alpha \times g(\mathcal{G}_{\mathcal{C}^*_g}) \\
\Leftrightarrow & M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\widetilde{\mathcal{C}}}) \leq \alpha \times M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\mathcal{C}^*_g}) \\
\Leftrightarrow & M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\widetilde{\mathcal{C}}}) + K(\mathcal{G}^+,\mathcal{G}^-) \leq \alpha \times M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\mathcal{C}^*_g}) + K(\mathcal{G}^+,\mathcal{G}^-) \\
\Rightarrow & M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\widetilde{\mathcal{C}}}) + K(\mathcal{G}^+,\mathcal{G}^-) \leq \alpha \times M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\mathcal{C}^*_g}) + \alpha \times K(\mathcal{G}^+,\mathcal{G}^-) \\
\Leftrightarrow & \underbrace{M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\widetilde{\mathcal{C}}}) + K(\mathcal{G}^+,\mathcal{G}^-)}_{= \bar{\ell}(\mathcal{G}_{\widetilde{\mathcal{C}}})\ \{\text{Lemma 2}\}} \leq \alpha \times \underbrace{\Big(M(\mathcal{G}^+,\mathcal{G}^-) \times g(\mathcal{G}_{\mathcal{C}^*_g}) + K(\mathcal{G}^+,\mathcal{G}^-)\Big)}_{= \bar{\ell}(\mathcal{G}_{\mathcal{C}^*_g})\ \{\text{Lemma 2}\}} \\
\Leftrightarrow & \bar{\ell}(\mathcal{G}_{\widetilde{\mathcal{C}}}) \leq \alpha \times \bar{\ell}(\mathcal{G}_{\mathcal{C}^*_g}) \ \Leftrightarrow\ \bar{\ell}(\mathcal{G}_{\widetilde{\mathcal{C}}}) \leq \alpha \times \bar{\ell}(\mathcal{G}_{\mathcal{C}^*_{\bar{\ell}}}),
\end{aligned}
$$

where the first equivalence step holds since $M(\mathcal{G}^+,\mathcal{G}^-) > 0$, the second one holds because of the assumption $K(\mathcal{G}^+,\mathcal{G}^-) \geq 0$, the third step holds as $\alpha > 1$, and the last step holds since, based on Lemma 2, the optimum of $g(\cdot)$ corresponds to the optimum of $\bar{\ell}(\cdot)$. ■

The MIL algorithm. Lemmas 1–3 provide the theoretical support and motivation for the first algorithm we propose for our Min-Interaction-loss-Clustering problem, named MIL, whose pseudocode is shown in Algorithm 8. Given two interaction graphs $\mathcal{G}^+$, $\mathcal{G}^-$, MIL simply builds an instance of Min-CC as per Equation (6.14), and applies the KwikCluster algorithm on it.

Proposition 3 *MIL takes $\mathcal{O}(|V|+|E|)$ time.*

Proof. The vertex-sampling step (Line 4 of Algorithm 8) can be implemented so as to take $\mathcal{O}(|V|)$ time overall, by preliminarily generating a random permutation of V (e.g., via $\mathcal{O}(|V|)$-time *Fisher-Yates shuffle* algorithm), and picking vertices u according to the ordering of that permutation. Also, the computation of τ^+_{uv}, τ^-_{uv} (Line 1 of Algorithm 8) can be restricted to the linked (u, v) pairs; in fact, for $(u, v) \notin E$, it holds that $\tau^+_{uv} = \tau^-_{uv}$, thus, in the next cluster-building step (Line 5 of Algorithm 8) the vertices v such that $(u, v) \notin E$ can be discarded. This makes the weight-computation and cluster-building take $\mathcal{O}(|E|)$ time overall. ■

Approximation guarantees. Thanks to Lemmas 1–3, the MIL algorithm can be shown to achieve expected factor-5 approximation guarantees for Min-Interaction-loss-Clustering if $K(\mathcal{G}^+,\mathcal{G}^-) \geq 0$:

Theorem 5 *If $K(\mathcal{G}^+,\mathcal{G}^-) \geq 0$, Algorithm 8 is a randomized expected 5-approximation algorithm for Problem 7.*

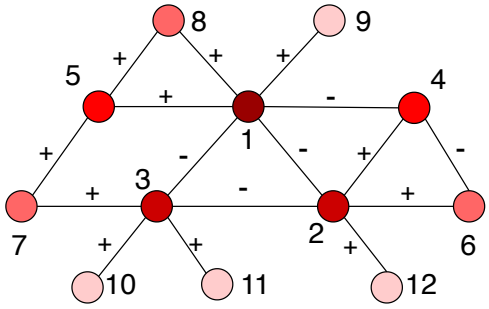


FIGURE 6.1: MIL algorithm: Effect of sampling pivots uniformly at random in general graphs.

PROOF. The weights τ^+_{uv}, τ^-_{uv} satisfy the probability constraint (Lemma 1). Thus, running the KwikCluster algorithm (i.e., Lines 3–7 of Algorithm 8) on a MIN-CC instance with τ^+_{uv}, τ^-_{uv} weights is proved to achieve expected 5-approximation guarantees for MIN-CC [2]. According to Lemma 3, If $K(\mathcal{G}^+,\mathcal{G}^-) \geq 0$, such a factor-5 approximation carries over to Problem 7. ■

Condition for approximation guarantees. The condition for MIL to be a 5-approximation algorithm for MIN-INTERACTION-LOSS-CLUSTERING is that the constant $K(\mathcal{G}^+,\mathcal{G}^-)$ (Equation (6.15)) is nonnegative. Here we show that this is a rather mild assumption, which is expected to hold for real-world interaction graphs. In fact:

$$
\begin{aligned}
K(\mathcal{G}^+,\mathcal{G}^-) \quad &= \sum_{u,v\in V} \frac{\sigma_{uv}}{2} \quad = \sum_{(u,v)\in E} \frac{\sigma_{uv}}{2} \quad + \sum_{(u,v)\notin E} \frac{\sigma_{uv}}{2} \\
&= \sum_{(u,v)\in E} \left(\frac{M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}] - \mathbb{E}[p^-_{uv}]}{2} \right) + \frac{M(\mathcal{G}^+,\mathcal{G}^-)}{2} \left(\binom{|V|}{2} - |E| \right) \geq 0 \\
&\Leftrightarrow \sum_{(u,v)\in E} \left(\frac{\mathbb{E}[p^+_{uv}] + \mathbb{E}[p^-_{uv}]}{2} \right) \leq \frac{M(\mathcal{G}^+,\mathcal{G}^-)}{2} |E| + \frac{M(\mathcal{G}^+,\mathcal{G}^-)}{2} \left(\binom{|V|}{2} - |E| \right) \\
&\Leftrightarrow \sum_{(u,v)\in E} \left(\mathbb{E}[p^+_{uv}] + \mathbb{E}[p^-_{uv}] \right) \leq M(\mathcal{G}^+,\mathcal{G}^-) \binom{|V|}{2}. \qquad (6.16)
\end{aligned}
$$

Thus, as $\mathbb{E}[p^+_{uv}], \mathbb{E}[p^-_{uv}] \leq M(\mathcal{G}^+,\mathcal{G}^-)$, the worst case to have the condition in the above Equation (6.16) satisfied is when $\mathbb{E}[p^+_{uv}] = \mathbb{E}[p^-_{uv}] = M(\mathcal{G}^+,\mathcal{G}^-)$, for all $(u,v) \in E$. This means that, in the worst case, $K(\mathcal{G}^+,\mathcal{G}^-)$ is guaranteed to be nonnegative if $|E| \leq \binom{|V|}{2}/2$, i.e., if the number of edges in the input interaction graphs is no more than half of the number of all unordered pairs of vertices. Note this relates to sparseness, which is typical in real-world graphs.

Stronger approximation guarantees. If the input weights, apart from satisfying the probability constraint, also obey the triangle inequality, then the KwikCluster algorithm for MIN-CC is shown to achieve better approximation guarantees, i.e., 2 instead of 5 [2]. In our setting this means that, if the weights defined in Equation (6.14) are such that $\tau^-_{uv} \leq \tau^-_{uz} + \tau^-_{zv}$, for all $u,v,z \in V$, then the proposed MIL algorithm becomes a 2-approximation algorithm for MIN-INTERACTION-LOSS-CLUSTERING.

To this purpose, let us denote with Δ^+_{uv} the difference $\mathbb{E}[p^+_{uv}] - \mathbb{E}[p^-_{uv}]$, for any $u,v \in V$; also, let $\Delta^-_{uv} := -\Delta^+_{uv}$. It can first be noted that, for any $u,v,z \in V$, whenever the triangle inequality $\tau^-_{uv} \leq \tau^-_{uz} + \tau^-_{zv}$ holds, then there exists an equivalent inequality in terms of the expectation differences, up to the constant $M(\mathcal{G}^+,\mathcal{G}^-)$.

Lemma 4 *Given two interaction graphs* $\mathcal{G}^+ = (V, E, P^+)$ *and* $\mathcal{G}^- = (V, E, P^-)$*, it holds that* $\tau^-_{uv} \leq \tau^-_{uz} + \tau^-_{zv} \Leftrightarrow \Delta^-_{uv} \leq \Delta^-_{uz} + \Delta^-_{zv} + M(\mathcal{G}^+,\mathcal{G}^-)$*, with* $u, v, z \in V$*.*

Proof. By definition (Equation (6.14)), $\tau^-_{uv} = \frac{1}{M(\mathcal{G}^+,\mathcal{G}^-)}(\mathbb{E}[p^-_{uv}] + \frac{\sigma_{uv}}{2})$, where $\sigma_{uv} = M(\mathcal{G}^+,\mathcal{G}^-) - (\mathbb{E}[p^+_{uv}] + \mathbb{E}[p^-_{uv}])$, thus it holds:

$$\begin{aligned}
&\tau^-_{uv} \leq \tau^-_{uz} + \tau^-_{zv} \Leftrightarrow \mathbb{E}[p^-_{uv}] + \frac{\sigma_{uv}}{2} \leq \mathbb{E}[p^-_{uz}] + \frac{\sigma_{uz}}{2} + \mathbb{E}[p^-_{zv}] + \frac{\sigma_{zv}}{2} \\
&\Leftrightarrow \mathbb{E}[p^-_{uv}] + M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}] \leq \mathbb{E}[p^-_{uz}] + M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uz}] + \\
&\quad + \mathbb{E}[p^-_{zv}] + M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{zv}] \\
&\Leftrightarrow \Delta^-_{uv} \leq \Delta^-_{uz} + \Delta^-_{zv} + M(\mathcal{G}^+,\mathcal{G}^-), \text{with } u, v, z \in V.l
\end{aligned}$$

■

The following Lemma 5 states the condition for stronger approximation guarantees, which requires that the difference $\mathbb{E}[p^+_{uv}] - \mathbb{E}[p^-_{uv}]$ lies in the range $[0, M(\mathcal{G}^+,\mathcal{G}^-)/2]$.

Lemma 5 *Given interaction graphs* $\mathcal{G}^+ = (V, E, P^+)$*,* $\mathcal{G}^- = (V, E, P^-)$*, if* $\Delta^+_{uv} \in [0, M(\mathcal{G}^+,\mathcal{G}^-)/2]$*, then* $\tau^-_{uv} \leq \tau^-_{uz} + \tau^-_{zv}$*, for any* $u, v, z \in V$*.*

Proof. By Lemma 4, it follows that:

$$\begin{aligned}
&\tau^-_{uv} \leq \tau^-_{uz} + \tau^-_{zv} \Leftrightarrow \Delta^-_{uv} + \Delta^+_{uz} + \Delta^+_{zv} \leq M(\mathcal{G}^+,\mathcal{G}^-) \\
&\Leftrightarrow \Delta^-_{uv} + \Delta^+_{uz} + \Delta^+_{zv} \leq 0 + \frac{M(\mathcal{G}^+,\mathcal{G}^-)}{2} + \frac{M(\mathcal{G}^+,\mathcal{G}^-)}{2} \\
&\Leftarrow \Delta^-_{uv} \leq 0, \quad \Delta^+_{uv} \leq \frac{M(\mathcal{G}^+,\mathcal{G}^-)}{2}, \forall u, v \in V,
\end{aligned}$$

which corresponds to $\Delta^+_{uv} \in [0, M(\mathcal{G}^+,\mathcal{G}^-)/2]$, as $\Delta^+_{uv} = -\Delta^-_{uv}$. ■

Theorem 6 *If* $K(\mathcal{G}^+,\mathcal{G}^-) \geq 0$ *and* $\Delta^+_{uv} \in [0, M(\mathcal{G}^+,\mathcal{G}^-)/2]$*,* $\forall u, v \in V$*, Algorithm 8 is a randomized expected 2-approximation algorithm for Problem 7.*

Thus, the stronger approximation guarantees of MIL hold if the expected interaction between any two users u and v when they are put in the same cluster is higher than the expected interaction when they are part of different clusters, and the former does not exceed the latter by more than half of the maximum interaction. This is actually not a strict condition, since it can be observed in application scenarios (especially $\Delta^+_{uv} \geq 0$).

Relation with correlation-clustering theory. Correlation-clustering (in)approximability result states that Min-CC on general (i.e., not necessarily complete) graphs is **APX**-hard, with best known approximation factor $\mathcal{O}(\log |V|)$ [19]. By constrast, in Theorems 5–6, we have shown that our Min-Interaction-loss-Clustering problem has constant-factor approximation guarantees for general instances of our problem, and such results are obtained by adapting correlation-clustering algorithms.

The above would apparently contradict the theory on correlation clustering. However, this is not the case, for the following reasons. First, although the original input interaction graphs we deal with are general (i.e., they may have missing edges), the way how we formulate our Min-Interaction-loss-Clustering problem, through the $M(\mathcal{G}^+,\mathcal{G}^-)$ constant (Equation (6.7)), guarantees that the actual graphs processed by the Min-Interaction-loss-Clustering algorithms are complete, i.e., they have

Algorithm 9 D-MIL

Input: Interaction graphs $\mathcal{G}^+ = (V, E, P^+)$, $\mathcal{G}^- = (V, E, P^-)$
Output: A clustering $\mathcal{C}$ of V

compute τ^+_{uv}, τ^-_{uv} for all $(u, v) \in E$ {Equation (6.14)}
$\mathcal{C} \leftarrow \emptyset$, $V' \leftarrow V$
while $V' \neq \emptyset$ **do**
  compute $d_{V'}(u) = |\{v \in V' \mid (u, v) \in E\}|$, for all $u \in V'$
  sample a pivot vertex $u \in V'$ with probability proportional to $d_{V'}(u)$
  $\mathcal{C}_u \leftarrow \{u\} \cup \{v \in V' \mid (u, v) \in E, \tau^+_{uv} > \tau^-_{uv}\}$
  add cluster $\mathcal{C}_u$ to $\mathcal{C}$ and remove all vertices in C_u from V'
end while

an edge with a positive weight between every pair of vertices. Second, the actual edge weights handled by the Min-Interaction-loss-Clustering algorithms (Equation (6.14)) are not arbitrary. Indeed, they are derived from the original weights with an ad-hoc rearrangement that guarantees the appealing properties we show above (i.e., fulfilment of the probability constraint and the fact that constant-factor guarantees for correlation clustering carry over to our problem). Such a rearrangement is a nice peculiarity of our problem, which is not possible in general: that is the main reason why this result is not in contrast with the inapproximability of Min-CC on general graphs.

Enhanced pivot-sampling strategy

The proposed MIL basically resembles the correlation-clustering KwikCluster algorithm [2] on a graph with ad-hoc-defined edge weights. However, KwikCluster is explicitly designed for complete graphs, whereas the input graphs for our Min-Interaction-loss-Clustering problem are general at first. Clearly, KwikCluster could be modified to handle general graphs, but this may lead to ineffectiveness. More specifically, we recall that the edge reweighting adopted in our MIL algorithm (Equation (6.14)) makes the input graph complete by assigning an equal positive and negative weight (equal to 1/2) to those vertex pairs that do not share an edge in the original graph. This way, putting those non-linked pairs in the same cluster or in different clusters does not make any difference in terms of objective-function value of the resulting solution. This fact is overlooked by KwikCluster, which, being designed for complete graphs, samples pivots uniformly at random, without taking into account how many neighbors a candidate pivot has in the original graph. This may raise inaccuracies, as shown in the next example.

Example 3 *Figure 6.1 shows an interaction graph where (u, v) edges are labeled according to what distribution prevails on the other in terms of expected value, i.e., "+" if $\mathbb{E}[p^+_{uv}] > \mathbb{E}[p^-_{uv}]$, "–" otherwise, while vertices are colored according to their degrees, i.e., the darker the shade, the more the number of edges adjacent to it. The optimal solution of* Min-Interaction-loss-Clustering *on this example consists of clusters $\{3, 7, 10, 11\}$, $\{2, 4, 6, 12\}$, $\{1, 5, 8, 9\}$. It is apparent that this optimal clustering may be found by the MIL algorithm if darker-colored vertices (i.e., vertices 1, 2, 3) are selected as pivots. Nevertheless, as MIL samples pivots uniformly at random, it is likely that one of the lighter-colored vertices (that are more than the darker-colored ones) becomes instead a pivot in the first place. This may lead to ineffective clusterings. For instance, assume vertex 10 is selected as a very first pivot. Even assuming*

that the next pivots are the darker-colored vertices 1 and 2, the ultimate clustering will be $\{3,10\}$, $\{1,5,8,9\}$, $\{2,4,6,12\}$, $\{11\}$, $\{7\}$, *which is far from the optimal one.*

The above example shows that, on general graphs, sampling pivots according to their degrees may be more appropriate than uniform sampling. This is the main intuition behind the second algorithm we propose in this work, which is termed D-MIL and whose outline is reported as Algorithm 9.

Proposition 4 *D-MIL takes* $\mathcal{O}(|E|\log|V|)$ *time.*

Proof. Sampling a vertex u with probability proportional to its (current) $d_{V'}(u)$ degree (cf. line 4 in Algorithm 2) can be implemented with a priority queue Q with priorities $d_{V'}(u) \times rnd$, where rnd is a random number. At the beginning, all vertices V are added to Q. Pivots are sampled following the order upon which vertices are extracted from Q, discarding vertices that have been assigned to clusters beforehand. Initializing Q and extracting all vertices from it takes $\mathcal{O}(|E|+|V|\log|V|)$ time. Building all the clusters takes $\mathcal{O}(|E|)$ time overall, as each cluster requires accessing the neighbors of the pivot $\mathcal{O}(1)$ times. Updating the degrees and the priorities of vertices in Q after a cluster has been built (and removed) takes $\mathcal{O}(|E|\log|V|)$ time. As a result, the overall time complexity of D-MIL is $\mathcal{O}(|E|\log|V|)$. ■

Hill climbing

The proposed MIL and D-MIL algorithms can be further improved by performing an a-posteriori hill-climbing step on the clusterings yielded by them. In particular, the idea is to consider relocating vertices in other clusters, as long as the resulting clustering has a better objective-function value. Such relocation steps may be efficiently implemented by incrementally computing marginal objective-function losses/gains. Specifically, given a clustering $\mathcal{C}$, let $\mathcal{C}'$ be the clustering obtained from $\mathcal{C}$ by moving a vertex u from cluster $C_u \in \mathcal{C}$ to a cluster $C'_u \neq C_u$. Taking into account the rearrangement of the $\bar{\ell}(\cdot)$ objective function stated in Theorem 4, removing u from C_u leads to a *decrease* in the $\bar{\ell}(\cdot)$ function equal to:

$$\sum_{v \in C_u \setminus \{u\}} \left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}]\right) \quad + \sum_{v \in V \setminus C_u} \left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]\right).$$

At the same time, adding u to C'_u leads to an *increase* of $\bar{\ell}(\cdot)$ equal to:

$$\sum_{v \in C'_u} \left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}]\right) \quad + \sum_{v \in V \setminus C'_u} \left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]\right).$$

Combining the expressions above, and denoting $\Delta^+_{uv} = \mathbb{E}[p^+_{uv}] - \mathbb{E}[p^-_{uv}]$, $\Delta^-_{uv} = -\Delta^+_{uv}$, we obtain:

$$\bar{\ell}(\mathcal{G}_{\mathcal{C}'}) = \bar{\ell}(\mathcal{G}_{\mathcal{C}}) + \sum_{v \in C_u \setminus \{u\}} \underbrace{\left(\left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]\right) - \left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}]\right)\right)}_{= \mathbb{E}[p^+_{uv}] - \mathbb{E}[p^-_{uv}] \,=\, \Delta^+_{uv}} +$$
$$+ \sum_{v \in C'_u} \underbrace{\left(\left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^+_{uv}]\right) - \left(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]\right)\right)}_{= \mathbb{E}[p^-_{uv}] - \mathbb{E}[p^+_{uv}] \,=\, \Delta^-_{uv}} +$$

$$+ \sum_{v \in V \setminus \{C_u \cup C'_u\}} \underbrace{\big(\big(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]\big) - \big(M(\mathcal{G}^+,\mathcal{G}^-) - \mathbb{E}[p^-_{uv}]\big)\big)}_{=0}$$

$$= \bar{\ell}(\mathcal{G}_\mathcal{C}) + \sum_{v \in C_u \setminus \{u\}} \Delta^+_{uv} + \sum_{v \in C'_u} \Delta^-_{uv} \quad = \quad \bar{\ell}(\mathcal{G}_\mathcal{C}) + \sum_{\substack{v \in C_u \setminus \{u\}, \\ (u,v) \in E}} \Delta^+_{uv} + \sum_{\substack{v \in C'_u, \\ (u,v) \in E}} \Delta^-_{uv}, \tag{6.17}$$

where the last equivalence holds as, for vertices $v : (u, v) \notin E$, $\mathbb{E}[p^+_{uv}] = \mathbb{E}[p^-_{uv}]$, and, then, $\Delta^+_{uv} = \Delta^-_{uv} = 0$. The hill-climbing step consists in iteratively picking a vertex u and a cluster $C'_u \neq C_u$ that minimize Eq. (6.17), and moving u from C_u to C'_u. This local-search process goes on until either no movement leading to a decrease in the $\bar{\ell}(\cdot)$ function exists, or a certain number of iterations I has been hit. The process is outlined as Algorithm 10.

Algorithm 10 HillClimbing

Input: Interaction graphs $\mathcal{G}^+ = (V, E, P^+)$, $\mathcal{G}^- = (V, E, P^-)$; A clustering $\mathcal{C}$ of V; An integer $I > 0$
Output: A clustering $\mathcal{C}'$ of V

$\mathcal{C}' \leftarrow \mathcal{C}$
for all $i = 1, \ldots, I$ **do**
  for every $u \in V$ let $C_u \in \mathcal{C}'$ the cluster of $\mathcal{C}'$ where u belongs to
  pick $u \in V$ and cluster $C'_u \in \mathcal{C}'$ ($C'_u \neq C_u$) that minimize Eq. (6.17)
  $\mathcal{C}'' \leftarrow$ clustering obtained from $\mathcal{C}'$ by moving u from C_u to C'_u
  if $\bar{\ell}(\mathcal{G}_{\mathcal{C}''}) < \bar{\ell}(\mathcal{G}_{\mathcal{C}'})$ **then**
    $\mathcal{C}' \leftarrow \mathcal{C}''$
  end if
end for

Proposition 5 *Hill-climbing for MIL and D-MIL takes $\mathcal{O}(I(|V| + |E|))$ time, with I number of iterations.*

Proof. Computing Equation (6.17) for all vertices takes $\mathcal{O}(|V| + |E|)$ time, as, for every vertex u, it requires processing u's neighbors only. Hence, denoting by I the number of iterations of the process, the overall time complexity of Algorithm 10 is $\mathcal{O}(I(|V| + |E|))$. ■

Final remark on approximation guarantees.

Here we summarize the approximation guarantees of all the proposed algorithms. Algorithm 8 achieves constant-factor approximation guarantees under mild conditions (either factor-5 or factor-2, see Theorems 5–6). Algorithm 9 does not come instead with any guarantees as of now: the study of its approximation properties is indeed an interesting open problem that we defer to future work. However, we remark that one can still have both the guarantees of Algorithm 8 without sacrificing the practical benefits of Algorithm 9: simply run both the algorithms and take the best one (in terms of objective-function value) among the two yielded solutions. This way, the approximation guarantees of Algorithm 8 would be preserved. As far as hill climbing procedure, instead, being a post-processing strategy that can only improve the outputs of Algorithm 8 or Algorithm 9, it does not alter the approximation properties of those

TABLE 6.1: Summary of real networks used in our evaluation: original data (cols. 2-5) and preprocessed data (cols. 6-7)

	$\|V\|$	$\sum_{t=1}^{T}\|E_t\|$	T	edge semantics	$\|E\|$	$\%\{\Delta_{uv}^{+}>0\}$
Amazon	2 146 057	22 728 036	115	co-rating	22 507 680	50
DBLP	1 824 701	11 865 584	80	co-authorship	8 344 615	52
Epinions	120 492	33 412 111	25	co-rating	24 994 363	50
HighSchool	327	47 589	1212	face-to-face	5 818	69
Last.fm	992	4 342 951	77	co-listening	369 973	50
PrimarySchool	242	55 043	390	face-to-face	8 317	66
ProsperLoans	89 269	3 343 271	307	economic	3 330 022	50
StackOverflow	2 433 067	16 200 209	51	Q/A	15 786 816	49
Wikipedia	343 860	18 086 734	101	co-editing	10 519 921	50
WikiTalk	2 863 439	10 335 318	192	communication	8 146 544	54

algorithms (i.e., it achieves guarantees if applied to Algorithm 8's solutions, while no guarantees hold for the combo Algorithm 9 + hill climbing).

We hereinafter denote with suffix $\_R$ the combo algorithms obtained by executing Algorithm 10 in cascade of Algorithm 8 (MIL_R) or Algorithm 9 (D-MIL_R).

6.5 Experimental evaluation

Data. We considered real-world networks as well as data produced by selected random network generation models. More specifically, we used 10 real-world, undirected, unweighted and timestamped networks, available from the KONECT project, except PrimarySchool and HighSchool from SocioPatterns, and StackOverflow from SNAP.[1] Table 6.1 summarizes main structural characteristics of the real-world networks used in our evaluation.

Each of the input temporal networks is treated as a sequence of undirected snapshot graphs $\langle G_1,\ldots,G_T\rangle$, where each $G_t=(V,E_t)$ $(t=1..T)$ models the vertex interactions at time t. The topology of the two graphs was derived by "flattening" the temporal network, i.e., $(u,v)\in E$ if u and v are linked in at least one graph from $\langle G_1,\ldots,G_T\rangle$. For each pair $u,v\in V$, if $(u,v)\notin E$ we assume that the two vertices will have no interaction with probability one, otherwise (i.e., $(u,v)\in E$) we define the distributions p_{uv}^{+}, p_{uv}^{-} as:

$$p_{uv}^{+}(w)=\frac{\Pr[w_G(u,v)=w\wedge\mathcal{C}(u)=\mathcal{C}(v)]}{\Pr[\mathcal{C}(u)=\mathcal{C}(v)]} \tag{6.18}$$

$$p_{uv}^{-}(w)=\frac{\Pr[w_G(u,v)=w\wedge\mathcal{C}(u)\neq\mathcal{C}(v)]}{\Pr[\mathcal{C}(u)\neq\mathcal{C}(v)]} \tag{6.19}$$

for $w\in\mathcal{D}(p_{uv})$, with $G\sqsubseteq\mathcal{G}_{\mathcal{C}}$ possible world induced by $\mathcal{C}$ from $\mathcal{G}$.

To estimate the above probabilities, we first derived a clustering solution on each graph from $\langle G_1,\ldots,G_T\rangle$, by initially assigning each vertex to a singleton cluster (i.e., starting from a solution totally biased towards the distributions in P^{-}), then iteratively performing agglomerative hierarchical clustering based on the minimization of a criterion function defined as the absolute value of the difference between the sum of the number of edges internal to each cluster and the sum of the number of edges external to each cluster. Although simple, this criterion function is better suited to

[1] http://konect.cc/, http://www.sociopatterns.org/datasets/, http://snap.stanford.edu/

our setting than classic community-detection approaches, such as *modularity*-based optimization criteria, which compares the actual within-community connectivity with the expected one based on a null model.

Once obtained the clustering solution on each G_t, we finally estimated $p^+_{uv}(w)$, resp. $p^-_{uv}(w)$, as the fraction of the timestamped graphs where u and v shared the same cluster, resp. were not in the same cluster, that corresponds to the interaction strength equal to w. The intuition for the definition of $p^+_{uv}(w)$ is that the more frequently u and v were grouped into the same cluster and their observed strength of interaction was w, the higher the probability that they will interact with strength w if they would be assigned to the same group; analogously for the functions p^-_{uv}. In our evaluation, we considered binary distribution functions; in this regard, note that the last column in Table 6.1 denotes the percentage of edges (u, v), in each network, such that $\mathbb{E}[p^+_{uv}] > \mathbb{E}[p^-_{uv}]$.

Concerning the synthetic data, we focused on two well-known random-graph models, namely Barabasi-Albert (hereinafter BA) and Watts-Strogatz (hereinafter WS) models. For the BA model, we varied the number of edges to attach with a new vertex, denoted as m, and for the WS model, we varied the distance (i.e., number of steps) within which two vertices will be connected, denoted as *neigh*. For both BA and WS models, we generated networks with 1000 vertices. For the BA model, we varied m from 0 to 1000, with steps of 5, for a total of 200 BA networks generated; analogously, for the WS model, we varied *neigh* from 0 to 1000/2=500, for a total of 100 WS networks generated. The expected values of interaction were randomly generated (uniformly) between 0 and 1.

Evaluation goals and competitors. We evaluated the *interaction loss*, the *size of the clustering* produced, and the *runtime performance* of the proposed methods, i.e., MIL, D-MIL, MIL_R, and D-MIL_R. All criteria measurements reported correspond to averages over 100 runs. We set the number of iterations I to 8, which experimentally revealed to be a good trade-off for balancing the three criteria. Moreover, we compared our proposed methods against three selected methods for community detection in signed graphs, namely CPM [121], GJA [47], and CPMap [32] (cf. Sect. 6.2). Since all such methods require only one weight, either positive or negative, for each edge, we set any weight to be the highest expected value between the positive and negative distribution (i.e., $\max\{\mathbb{E}[p^+_{uv}], \mathbb{E}[p^-_{uv}]\}$), changing the sign of the weight in case the maximum corresponds to the negative distribution. Since [121] deals with directed networks, our evaluation networks were modified by replacing each edge with two reciprocal directed edges. Also, the objective function of the methods in [121] and [47] corresponds to that used in the Louvain modularity-optimization-based method. For all methods, we used the default parameter setting.

6.5.1 Results on real data

Interaction loss. Table 6.2 reports the values of the discounted interaction loss (cf. Eq. (6.11)). As expected, the clustering solutions of the enhanced methods (i.e., MIL_R and D-MIL_R) show consistently lower loss than the solutions produced by MIL and D-MIL. Also, we observe that, on all datasets, D-MIL outperforms MIL and, in turn, D-MIL_R outperforms MIL_R, which confirms our initial hypothesis that the degree-based heuristic should be preferred on real-world networks. Notably, considering the total average of loss values over all networks (last row in the table), the percentage loss-decrease values obtained by MIL_R are 37% and 28% against MIL and D-MIL, respectively, while the values obtained by D-MIL_R are 39%, 30% and 3% against MIL, D-MIL, and MIL_R, respectively.

TABLE 6.2: Average loss values.

	MIL	MIL_R	D-MIL	D-MIL_R	CPM [121]	GJA [47]	CPMap [32]
Amazon	4.80×10^6	3.82×10^6	4.49×10^6	3.69×10^6	4.38×10^6	4.33×10^6	$\mathbf{3.66 \times 10^6}$
DBLP	3.94×10^6	3.17×10^6	3.70×10^6	3.01×10^6	$\mathbf{2.55 \times 10^6}$	2.89×10^6	2.81×10^6
Epinions	12.92×10^6	4.71×10^6	9.06×10^6	$\mathbf{4.70 \times 10^6}$	9.80×10^6	8.82×10^6	5.06×10^6
High School	4.59×10^3	3.50×10^3	4.44×10^3	3.35×10^3	4.29×10^3	3.43×10^3	$\mathbf{3.29 \times 10^3}$
Last.fm	164.67×10^3	$\mathbf{150.25 \times 10^3}$	163.35×10^3	$\mathbf{150.25 \times 10^3}$	161.53×10^3	160.66×10^3	151.60×10^3
Primary School	6.95×10^3	5.01×10^3	6.80×10^3	$\mathbf{4.92 \times 10^3}$	6.48×10^3	6.27×10^3	5.46×10^3
Prosper Loans	1.82×10^6	1.30×10^6	1.81×10^6	$\mathbf{1.28 \times 10^6}$	$\mathbf{1.28 \times 10^6}$	1.30×10^6	1.39×10^6
Stack-Overflow	12.39×10^6	8.83×10^6	11.91×10^6	$\mathbf{8.65 \times 10^6}$	9.90×10^6	9.26×10^6	10.81×10^6
Wikipedia	6.74×10^6	5.31×10^6	6.44×10^6	$\mathbf{5.26 \times 10^6}$	5.84×10^6	5.84×10^6	5.83×10^6
WikiTalk	6.29×10^6	3.72×10^6	5.41×10^6	$\mathbf{3.38 \times 10^6}$	3.68×10^6	5.13×10^6	NA
tot. average	4.91×10^6	3.10×10^6	4.30×10^6	$\mathbf{3.01 \times 10^6}$	3.76×10^6	3.77×10^6	3.30×10^6

TABLE 6.3: Average clustering sizes.

	MIL	MIL_R	D-MIL	D-MIL_R	CPM [121]	GJA [47]	CPMap [32]
Amazon	1.51×10^6	1.36×10^6	1.47×10^6	1.34×10^6	1.17×10^6	1.03×10^6	1.34×10^6
DBLP	986.02×10^3	614.86×10^3	858.93×10^3	557.90×10^3	354.03×10^3	506.72×10^3	393.38×10^3
Epinions	76.81×10^3	47.54×10^3	65.59×10^3	47.51×10^3	16.73×10^3	16.68×10^3	65.31×10^3
High School	45.26	8.16	37.66	6.38	9.00	7.00	8.00
Last.fm	57.04	37.64	42.10	36.94	3.00	4.00	37.00
Primary School	16.44	1.20	15.12	1.04	5.00	5.00	2.00
Prosper Loans	39.60×10^3	3.75×10^3	26.06×10^3	3.70×10^3	1.54×10^3	1.13×10^3	7.49×10^3
Stack-Overflow	1.74×10^6	308.66×10^3	1.27×10^6	237.36×10^3	106.58×10^3	13.78×10^3	188.44×10^3
Wikipedia	276.14×10^3	157.77×10^3	246.29×10^3	168.80×10^3	113.64×10^3	108.82×10^3	209.68×10^3
WikiTalk	2.77×10^6	381.88×10^3	1.99×10^6	485.66×10^3	351.73×10^3	1.69×10^6	NA
tot. average	7.40×10^5	2.87×10^5	5.93×10^5	2.84×10^5	2.11×10^5	3.37×10^5	2.45×10^5

Number of clusters. Table 6.3 also shows the size of the clusterings produced by the various methods. D-MIL always yields a smaller number of clusters than MIL. This happens since, by pivoting over vertices with higher degree, it is more likely to sample vertices having a larger number of incident edges such that $p^+_{uv} > p^-_{uv}$. Also, note that MIL and D-MIL tend to produce more clusters than MIL_R and D-MIL_R, up to 157% and 160%, respectively, of percentage size-increase. This is not surprising since the reduction of loss is related to a decrease in the clustering size.

Time performance. Table 6.4 reports the average time performance of the various methods. For MIL_R and D-MIL_R, we show details about the optimization phase time (i.e., Algorithm 10).[2] Consistently with the computational complexity analysis (Sect. 6.4), D-MIL tends to perform worse than MIL, and so D-MIL_R against MIL_R. Nevertheless, in PrimarySchool, D-MIL_R runtime is found to be slightly better than MIL_R: this happens since, despite the two methods converge to almost the same local optimum, D-MIL_R starts from a solution which is closer to the final solution as compared to the one produced by MIL_R (cf. Table 6.2), thus requiring a fewer number of optimization steps (10% decrease), and hence execution time.

6.5.2 Results on synthetic data

We analyzed loss, clustering size and time performance of the proposed methods, averaged over 100 network-generation runs. Each of the assessment criteria was measured by varying the m parameter for BA networks and the *neigh* parameter for WS networks.

Interaction loss. Figures 6.2(a)-(b) show the percentage loss-decrease of D-MIL over MIL, and of D-MIL_R over MIL_R. In agreement with the results obtained on real-world networks, the pairwise loss variation is relatively low, for either pair of methods,

[2]Experiments were carried out on a Ubuntu 18.04.2 LTS machine with Intel Xeon(R) Gold 5118 CPU @ 2.30GHz × 48 processor and 256GB ram

TABLE 6.4: Execution times (in seconds)

	MIL	MIL_R	opt. time	D-MIL	D-MIL_R	opt. time	CPM [121]	GJA [47]	CPMap [32]
Amazon	8.63	347.77	339.14	97.40	427.28	329.88	2 248.9	1 020 122.23	669.114
DBLP	6.11	189.63	183.52	71.15	251.24	180.09	1 570.41	147 159.68	601.044
Epinions	5.90	327.27	321.38	18.90	348.11	329.21	797.71	34 998.9	592.901
High School	0.00	0.04	0.04	0.01	0.04	0.03	0.2	0.19	2.716
Last.fm	0.03	3.48	3.45	0.14	3.72	3.58	7.73	21.54	10.467
PrimarySchool	0.00	0.06	0.05	0.01	0.05	0.04	0.125	0.1	3.698
ProsperLoans	0.70	48.74	48.04	4.31	52.06	47.75	179.78	30 152.47	116.59
StackOverflow	7.88	319.67	311.79	105.68	397.39	291.72	2 465.76	1 140 054.23	1519.943
Wikipedia	2.41	150.03	147.62	19.05	160.69	141.64	826.93	189 345.74	316.438
WikiTalk	13.92	203.49	189.56	129.10	300.68	171.58	1 165.01	650 282.4	NA

as long as the network is sparse (i.e., lower m or *neigh* values); more specifically, the percentage loss-decrease of D-MIL w.r.t. MIL is just 0.4% and 0.15% for BA and WS networks, respectively, while corresponding values for D-MIL_R w.r.t. MIL_R are further lower (i.e., below 0.1% for BA and 0.05% for WS). In all cases, the loss variation becomes negligible already for mid regimes of the x-axis. Also, for WS networks having rewiring probability lower than 0.5 (results not shown), the pairwise loss variation would be negligible even for low values of *neigh* (i.e., higher sparsity).

Number of clusters. The clustering size (Figs. 6.2(c)-(d)) decreases as the number of edges increases with the value of m or *neigh*. MIL always yields a larger number of clusters than the other methods, especially on BA networks still with highest values of m, followed by D-MIL and the enhanced methods, which produce almost the same number of clusters. In general, the difference among the methods is emphasized for sparser networks and decreases as the networks tend to become almost complete.

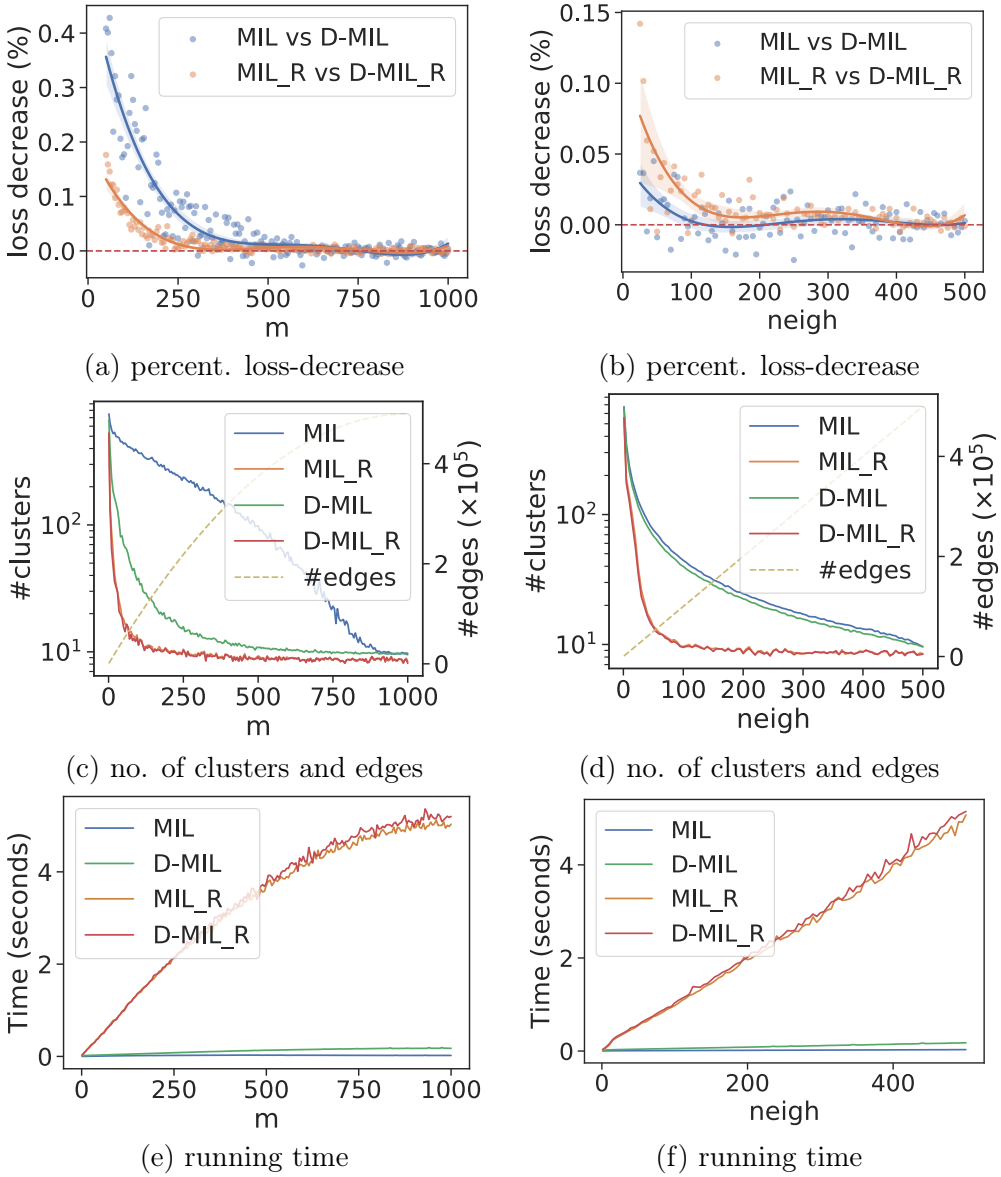
Time performance. Figures 6.2(e)-(f) show the running times of the methods. Like for real networks (cf. Table 6.4), MIL is the fastest method, immediately followed by D-MIL, showing to be very robust as the number of edges (and hence, density) of the network increases, i.e., as m and *neigh* parameter values increase for BA and WS networks, respectively. On the contrary, the enhanced methods achieve higher runtime, with D-MIL_R being slightly slower than MIL_R; nonetheless, they scale linearly on WS networks, and sublinearly on BA networks, with the density of the network.

6.5.3 Evaluation with competing methods

On real networks, considering the interaction-loss values reported in the last three groups of columns in Table 6.2, it is worth noticing that our D-MIL_R and MIL_R outperform all competing methods in most cases, with average percentage loss-decreases of 20% for D-MIL_R, resp. 18% for MIL_R, against both CPM and GJA, and 10% for D-MIL_R, resp. 8% for MIL_R, against CPMap. In terms of clustering size, GJA generally produces the lowest number of clusters (6 cases out of 10), though it holds the opposite on average due to the performance on WikiTalk, while CPMap generates solutions with higher size than the others (7 cases out of 10).

Concerning execution times, GJA and CPMap are the slowest and the fastest method, respectively, among the competitors. Remarkably, CPMap is always outperformed by all of our MLI methods, with a minimum gap (w.r.t. D-MIL_R) of 119% time-increase.

Figures 6.3(a)-(b) show the percentage loss decrease of D-MIL_R against each competitor, which is always positive. Both CPM and GJA produce fewer clusters



(a) percent. loss-decrease (b) percent. loss-decrease

(c) no. of clusters and edges (d) no. of clusters and edges

(e) running time (f) running time

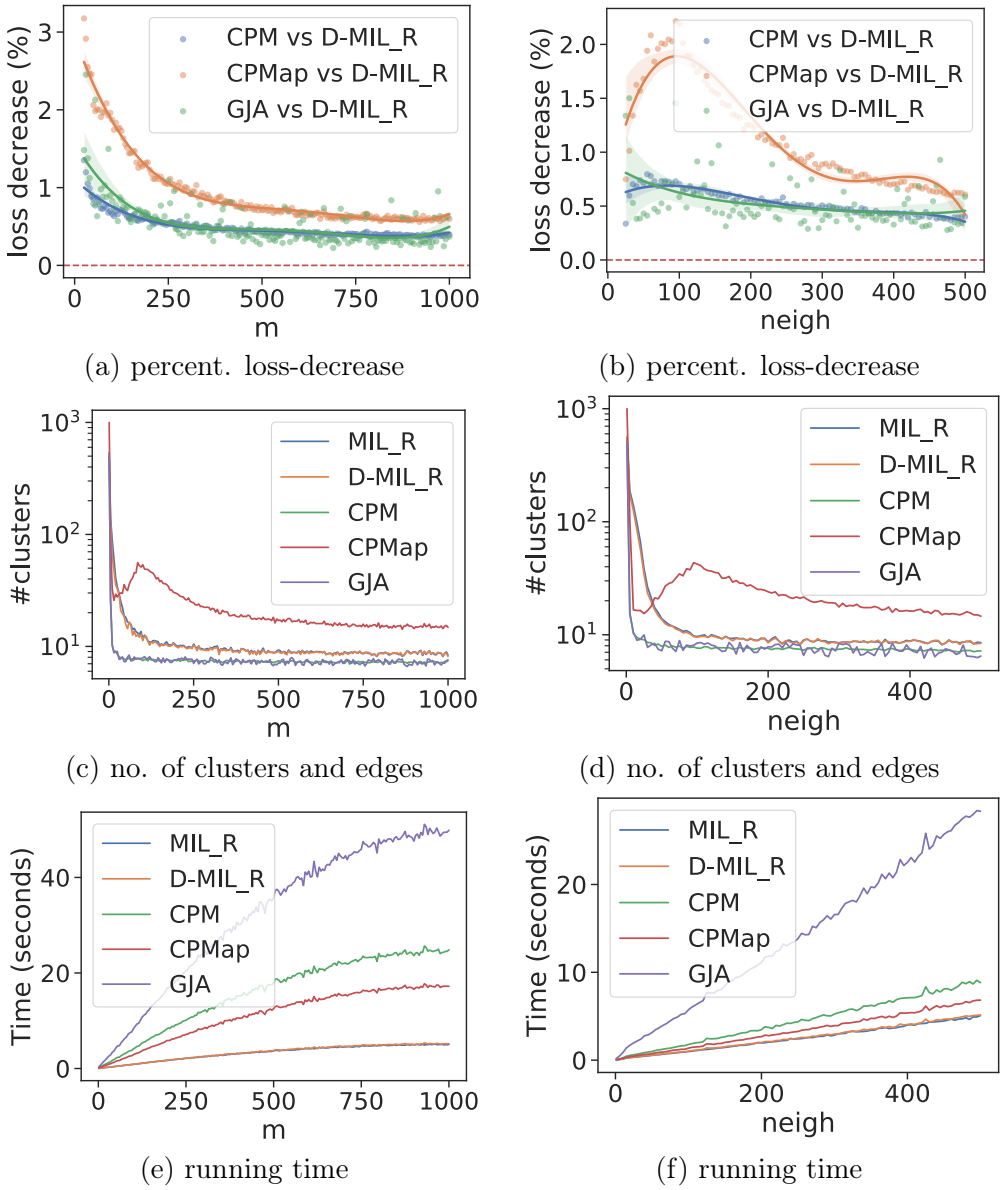
FIGURE 6.2: Results on BA networks (left side) and on WS networks, with rewiring probability 0.5 (right side).

than the other methods, whereas CPMap, except for low m and *neigh*, yields the highest number of clusters (c.f. Figs. 6.3(c)-(d)). CPMap is the fastest method among competitors, followed by CPM and GJA (c.f. Figs. 6.3(e)-(f)). All competitors are anyway outperformed by MIL_R and D-MIL_R.

6.6 Chapter review

We introduced the problem of optimizing the overall interaction in probabilistic graphs under clustering constraints. We theoretically characterized the problem and devised both approximation algorithms and heuristics, whose effectiveness, efficiency and superiority w.r.t. competing methods was assessed in the experiments.

As future work, we plan to extend the problem formulation in order to capture overlapping clusters as well as consider the case when the probability distributions



(a) percent. loss-decrease

(b) percent. loss-decrease

(c) no. of clusters and edges

(d) no. of clusters and edges

(e) running time

(f) running time

FIGURE 6.3: Results on BA networks (left side) and on WS networks, with rewiring probability 0.5 (right side).

of interaction are not given but only samples coming from that distributions can be observed.

Chapter 7

Conclusion

This thesis has focused on mining and learning entities' behaviour patterns in complex graph data. We dealt with different graph data model (e.g. multilayer, temporal and uncertain graphs), each describing in different way the intrinsic multifaced nature of the represented real world systems. The main objective of this research project has been developing models and methods to discover, on complex graph data, hidden and strong relationships between entities, such as *group associations* and *trust relationships*, which clearly exist in real world scenarios.

Concerning the discovery of groups (or clustering) in complex graph data, this thesis explored it from two different perspectives, depending on what kind of clusterings we look for. Following the widely accepted definition of community [96], the first perspective, adopted in the research line presented in Chapter 3 and Chapter 4 in the context of multilayer and temporal graphs respectevely, aims to find communities which are densely linked internally and sparsely connected externally. The second perspective, introduced in Chapter 6 in the context of maximizing interactions (conditioned by cluster memberships) in probabilistic graphs, looks for clusters that induce high connectivity both internally and externally.

In Chapter 3 we introduced a new framework for consensus community detection in multilayer networks. This was designed to identify communities whose nodes are internally connected by many edges possibly of different types, and are externally connected by few edges of different types. Moreover, by exploiting parameter-free generative models for graph pruning, this framework overcomes the dependency on a user-specified threshold for the global denoising of the co-association graph. The experimental evaluation of the proposed M-EMCD* algorithm, designed to enhance the modularity-optimization process w.r.t. existing EMCD [114] method, confirmed the beneficial effect of using model-based filtering methods and also showed its superiority on state-of-the-art multilayer community detection.

Another contribution of this thesis is represented by CreDENCE (cf. Chapter 4), a CMAB-based method for the novel sequential problem of identifying, for each snapshot in a temporal network, a dynamic consensus community structure. This is designed to encompass the whole information available in the sequence of observed temporal snapshots of a network in order to be representative of the knowledge available from community structures at the different time steps. The novel concept of dynamic consensus community structure embraces the widely accepted notion of community - in temporal networks this translates into a group of nodes which are internally connected by many edges, and are externally connected by few edges in the most recent snapshots. Unlike existing approaches, it has been designed to be able to embed long-term changes in the community formation as well as to capture short-term effects and newly observed community structures.

In Chapter 6 we introduced a new uncertain graph model where entities' interaction patterns depend on their cluster memberships. This covers any scenario, such as

recommendation in social-media platforms and team formation tasks, where a set of actions over entities, modeled as cluster assignments, can alter the entities' interaction behaviour. Upon this model, the novel problems of finding a clustering such as to maximize (resp. minimize) the overall interactions (resp. interaction loss) have been introduced. In other words, the goal is to find the clusters which induce, according to the novel uncertain graph model, high connectivity both internally and externally. The theoretical characterization of the two problems have shown their connection with the correlation clustering problem which inspired the introduced approximation algorithms and heuristics, namely MIL, D-MIL, MIL_R and D-MIL_R.

Concerning trust relationships, in Chapter 5 we introduced the Trust Network Inference problem and proposed a preference-learning-based approach to solve it. The proposed solution contributes to research in this area since the actual lack of methods for inferring a trust network from social interactions. For the practitioners, the introduced approach can be regarded as key-enabling for any application that needs to build a trust network associated with a social environment from user interactions observed over time, in order to exploit the inferred trust relatioships in a variety of mining tasks.

Future Research

The research described in this thesis can be further extended in several directions.

Research on dynamic consensus community detection in temporal networks, presented in Chapter 4, adopted the CMAB paradigm to manage the exploration-exploitation trade-off which, in that context, translated into the balancing over time between the need for embedding long-term changes observed in the community formation and the need for capturing short-term effects and newly observed community structures. (C)MAB algorithms usually come with some theoretical guarantees on their performances if some properties on the input instance (e.g. the existance of approximation algorithms for optimizing the adopted reward function) are satisfied. However, in this thesis the application of CMAB paradigm to the dynamic consensus community detection problem was not focused on theoretical analysis. This research line may be extended to provide insights on the theoretical properties of the problem and algorithms when some assumptions on the input (e.g. evolution rate of the network) are made. A further goal is to learn our model parameters to best fit the community structure and evolution in a given temporal network.

The contribution of this thesis to the novel problem of optimizing interactions in probabilistic graphs is actually at an early stage; thus, the improvements in this regard are manifold. A first goal to be accomplished is to extend the problem formulation in order to capture overlapping clusters as well as clusters with specific size bounds since some application scenarios, as the one mentioned in Chapter 6, requires specific properties on the desired clusters. A further goal is to consider the more realistic case when the probability distributions of interaction are not given but only samples coming from that distributions can be observed over time.

Research on trust network inference may be extended as follows. Different definitions of trust-context and of structural/content affinity functions could easily be integrated into our proposed TNI framework; for instance, as we mentioned earlier in Chapter 5, the trust-context model could be defined according to various topological structures, such as expanded ego-networks or community structures. Another aspect of interest is to extend our method to build a trust network incrementally in online tasks, i.e., inferring and maintaining/updating a trust network over a stream of interaction networks.

Bibliography

[1] P. Agarwal, R. Verma, A. Agarwal, and T. Chakraborty. "DyPerm: Maximizing Permanence for Dynamic Community Detection". In: *Proc. Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD)*. 2018, pp. 437–449.

[2] N. Ailon, M. Charikar, and A. Newman. "Aggregating inconsistent information: Ranking and clustering". In: *Journal of the ACM (JACM)* 55.5 (2008), 23:1–23:27.

[3] Javed A Aslam and Scott E Decatur. "General bounds on statistical query learning and PAC learning with noise via hypothesis boosting". In: *Information and Computation* 141.2 (1998), pp. 85–118.

[4] N. Bansal, A. Blum, and S. Chawla. "Correlation Clustering". In: *Machine Learning* 56.1 (2004), pp. 89–113.

[5] M. Berlingerio, M. Coscia, and F. Giannotti. "Finding and Characterizing Communities in Multidimensional Networks". In: *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 2011, pp. 490–494.

[6] M. Berlingerio, F. Pinelli, and F. Calabrese. "ABACUS: frequent pattern mining-based community discovery in multidimensional networks". In: *Data Min. Knowl. Discov.* 27.3 (2013), pp. 294–320.

[7] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre. "Fast unfolding of communities in large networks". In: *J. Stat. Mech.* 10 (2008), P10008.

[8] F. Bonchi, F. Gullo, A. Kaltenbrunner, and Y. Volkovich. "Core decomposition of uncertain graphs". In: *Proc. ACM KDD Conf.* 2014, pp. 1316–1325.

[9] Z. Borbora, M. A. Ahmad, K. Z. Haigh, J. Srivastava, and Z. Wen. "Exploration of Robust Features of Trust Across Multiple Social Networks". In: *Proc. IEEE Conf. on Self-Adaptive and Self-Organizing Systems (SASOW)*. 2011, pp. 27–32.

[10] Oualid Boutemine and Mohamed Bouguessa. "Mining community structures in multidimensional networks". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 11.4 (2017), pp. 1–36.

[11] Paul S Bradley and Usama M Fayyad. "Refining initial points for k-means clustering." In: *Proc. of Int. Conf. on Machine Learning (ICML)*. Vol. 98. Citeseer. 1998, pp. 91–99.

[12] U. Brandes, P. Kenis, J. Lerner, and D. van Raaij. "Network analysis of collaboration structure in Wikipedia". In: *Proc. of World Wide Web Conf. (WWW)*. 2009, pp. 731–740.

[13] P. Brodka, S. Saganowski, and P. Kazienko. "GED: the method for group evolution discovery in social networks". In: *Social Netw. Analys. Mining* 3.1 (2013), pp. 1–14.

[14] C. Buckley and E. M. Voorhees. "Retrieval evaluation with incomplete information". In: *Proc. ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR)*. 2004, pp. 25–32.

[15] R. Busa-Fekete, B. Szörényi, W. Cheng, P. Weng, and E. Hüllermeier. "Top-k Selection based on Adaptive Sampling of Noisy Preferences". In: *Proc. of Int. Conf. on Machine Learning (ICML)*. 2013, pp. 1094–1102.

[16] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. "A comprehensive survey of graph embedding: Problems, techniques, and applications". In: *IEEE Transactions on Knowledge and Data Engineering* 30.9 (2018), pp. 1616–1637.

[17] M. Ceccarello, C. Fantozzi, A. Pietracaprina, G. Pucci, and F. Vandin. "Clustering Uncertain Graphs". In: *PVLDB* 11.4 (2017), pp. 472–484.

[18] T. Chakraborty, S. Srinivasan, N. Ganguly, A. Mukherjee, and S. Bhowmick. "Permanence and community structure in complex networks". In: *ACM Trans. Knowl. Discov. Data* 11.2 (2016), p. 14.

[19] M. Charikar, V. Guruswami, and A. Wirth. "Clustering with qualitative information". In: *JCSS* 71.3 (2005), pp. 360–383.

[20] W. Chen, Y. Wang, and Y. Yuan. "Combinatorial Multi-Armed Bandit: General Framework and Applications". In: *Proc. of Int. Conf. on Machine Learning (ICML)*. 2013, pp. 151–159.

[21] Michele Coscia. "Multidimensional network analysis". PhD thesis. Ph. D. thesis, Universitá Degli Studi Di Pisa, Dipartimento di Informatica, 2012.

[22] J. Crawford and T. Milenkovic. "ClueNet: Clustering a temporal network based on topological similarity rather than denseness". In: *PLOS ONE* 13.5 (2018), pp. 1–25.

[23] N. Dakiche, F. B.-S. Tayeb, Y. Slimani, and K. Benatchba. "Tracking community evolution in social networks: A survey". In: *Inf. Process. Manag.* 56.3 (2019), pp. 1084–1102.

[24] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. "Correlation clustering in general weighted graphs". In: *TCS* 361.2-3 (2006), pp. 172–187.

[25] Navid Dianati. "Unwinding the hairball graph: Pruning algorithms for weighted complex networks". In: *Physical Review E* 93 (2016), p. 012304.

[26] M. E. Dickison, M. Magnani, and L. Rossi. *Multilayer Social Networks*. UK: Cambridge University Press, 2016.

[27] Thang N Dinh, Xiang Li, and My T Thai. "Network clustering via maximizing modularity: Approximation algorithms and theoretical limits". In: *2015 IEEE International Conference on Data Mining*. IEEE. 2015, pp. 101–110.

[28] Manlio De Domenico, Andrea Lancichinetti, Alex Arenas, and Martin Rosvall. "Identifying modular flows on multilayer networks reveals highly overlapping organization in interconnected systems". In: *Phys. Rev. X* 5 (2015), p. 011027.

[29] Manlio De Domenico, Vincenzo Nicosia, Alexandre Arenas, and Vito Latora. "Structural reducibility of multilayer networks." In: *Nature communications* 6 (2015), p. 6864.

[30] Carlotta Domeniconi and Muna Al-Razgan. "Weighted cluster ensembles: Methods and analysis". In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 2.4 (2009), pp. 1–40.

[31] Carlotta Domeniconi, Dimitrios Gunopulos, Sheng Ma, Bojun Yan, Muna Al-Razgan, and Dimitris Papadopoulos. "Locally adaptive metrics for clustering high dimensional data". In: *Data Mining and Knowledge Discovery* 14.1 (2007), pp. 63–97.

[32] P. Esmailian and M. Jalili. "Community detection in signed networks: the role of negative ties in different scales". In: *Scientific reports* 5 (2015), p. 14339.

[33] X. Fan, D. He, and J. Bi. "Trustworthiness and Untrustworthiness Inference with Group Assignment". In: *Proc. Int. Conf. on Web Services (ICWS)*. 2018, pp. 389–404.

[34] X. Z. Fern and C. E. Brodley. "Solving Cluster Ensemble Problems by Bipartite Graph Partitioning". In: *Proc. of Int. Conf. on Machine Learning (ICML)*. 2004, p. 36.

[35] T. La Fond, G. Sanders, C. Klymko, and H. Van Emden. "An ensemble framework for detecting community changes in dynamic networks". In: *Proc. IEEE Conf. on High Performance Extreme Computing (HPEC)*. 2017, pp. 1–6.

[36] Santo Fortunato and Marc Barthelemy. "Resolution limit in community detection". In: *Proceedings of the national academy of sciences* 104.1 (2007), pp. 36–41.

[37] A. Fred. "Finding Consistent Clusters in Data Partitions". In: *Proc. Work. on Multiple Classifier Systems*. 2001, pp. 309–318. ISBN: 978-3-540-48219-2.

[38] A. L. N. Fred and A. K. Jain. "Data clustering using evidence accumulation". In: *Object recognition supported by user interaction for service robots*. Vol. 4. 2002, 276–280 vol.4.

[39] R.E. Funderlic and C.D. Meyer. "Sensitivity of the stationary distribution vector for an ergodic Markov chain". In: *Linear Algebra and its Applications* 76 (1986), pp. 1 –17. ISSN: 0024-3795.

[40] Yu G., Chunpeng G., Gao C., and Ge Y. "Effective and Efficient Clustering Methods for Correlated Probabilistic Graphs". In: *IEEE TKDE* 26.5 (2014), pp. 1117–1130.

[41] Y. Gai, B. Krishnamachari, and R. Jain. "Combinatorial Network Optimization With Unknown Variables: Multi-Armed Bandits With Linear Rewards and Individual Observations". In: *IEEE/ACM Trans. Netw.* 20.5 (2012), pp. 1466–1478.

[42] W. Gao, W. Luo, and C. Bu. "Adapting the TopLeaders algorithm for dynamic social networks". In: *The Journal of Supercomputing* (2017).

[43] V. Gemmetto, A. Cardillo, and D. Garlaschelli. "Irreducible network backbones: unbiased graph filtering via maximum entropy". In: *arXiv* (2017).

[44] M. Giatsoglou and A. Vakali. "Capturing Social Data Evolution Using Graph Clustering". In: *IEEE Internet Computing* 17.1 (2013), pp. 74–79.

[45] A. Gionis, H. Mannila, and P. Tsaparas. "Clustering Aggregation". In: *Proc. of IEEE Int. Conf. on Data Engineering (ICDE)*. 2005, pp. 341–352.

[46] J.A. Golbeck. "Computing and Applying Trust in Web-based Social Networks". PhD thesis. College Park, MD, USA, 2005.

[47] S. Gómez, P. Jensen, and A. Arenas. "Analysis of community structure in networks of correlated data". In: *Physical Review E* 80.1 (2009), p. 016114.

[48] F. C. Graham, A. Tsiatas, and W. Xu. "Dirichlet PageRank and Ranking Algorithms Based on Trust and Distrust". In: *Internet Mathematics* 9.1 (2013), pp. 113–134.

[49] D. Greene, D. Doyle, and P. Cunningham. "Tracking the evolution of communities in dynamic social networks". In: *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 2010, pp. 176–183.

[50] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral. "Modularity from fluctuations in random graphs and complex networks". In: *Physical Review E* 70.2 (2004), p. 025101.

[51] F. Gullo, A. Tagarelli, and S. Greco. "Diversity-Based Weighting Schemes for Clustering Ensembles". In: *Proc. SDM*. 2009, pp. 437–448.

[52] Francesco Gullo, Carlotta Domeniconi, and Andrea Tagarelli. "Projective clustering ensembles". In: *Data Mining and Knowledge Discovery* 26.3 (2013), pp. 452–511.

[53] G. Guo, J. Zhang, D. Thalmann, and N. Yorke-Smith. "ETAF: An Extended Trust Antecedents Framework for Trust Prediction". In: *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 2014, pp. 540–547.

[54] Y. Gur, A. J. Zeevi, and O. Besbes. "Stochastic Multi-Armed-Bandit Problem with Non-stationary Rewards". In: *Proc. Conf. on Neural Information Processing Systems (NIPS)*. 2014, pp. 199–207.

[55] Z. Gyöngyi, H. Garcia-Molina, and J. O. Pedersen. "Combating Web Spam with TrustRank". In: *Proc. of Int. Conf. on Very Large Data Bases (VLDB)*. 2004, pp. 576–587.

[56] Z. Halim, M. Waqas, and S. F. Hussain. "Clustering large probabilistic graphs using multi-population evolutionary algorithm". In: *Inf. Sci.* 317 (2015), pp. 78–95.

[57] K. Han, F. Gui, X. Xiao, J. Tang, Y. He, Z. Cao, and H. Huang. "Efficient and Effective Algorithms for Clustering Uncertain Graphs". In: *PVLDB* 12.6 (2019), pp. 667–680.

[58] O. Hanteer, L. Rossi, D. Vega D'Aurelio, and M. Magnani. "From Interaction to Participation: The Role of the Imagined Audience in Social Media Community Detection and an Application to Political Communication on Twitter". In: *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 2018, pp. 531–534.

[59] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[60] J. He and D. Chen. "A fast algorithm for community detection in temporal network". In: *Physica A: Statistical Mechanics and its Applications* 429 (2015), pp. 87–94.

[61] Pa. Wagenseller III, F. Wang, and W. Wu. "Size Matters: A Comparative Analysis of Community Detection Algorithms". In: *IEEE Trans. Comput. Social Systems* 5.4 (2018), pp. 951–960.

[62] Lucas GS Jeub, Michael W Mahoney, Peter J Mucha, and Mason A Porter. "A local perspective on community structure in multilayer networks". In: *arXiv preprint arXiv:1510.05185* (2015).

[63] W. Jiang, G. Wang, and J. Wu. "Generating trusted graphs for trust evaluation in online social networks". In: *Future Generation Comp. Syst.* 31 (2014), pp. 48–58.

[64] P. Jiao, W. Wang, and D. Jin. "Constrained common cluster based model for community detection in temporal and multiplex networks". In: *Neurocomputing* 275 (2018), pp. 768–780.

[65] A. Jøsang, E. Gray, and M. Kinateder. "Simplification and analysis of transitive trust networks". In: *Web Intelligence and Agent Systems* 4.2 (2006), pp. 139–161.

[66] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. "Multilevel hypergraph partitioning: Applications in VLSI domain". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 7.1 (1999), pp. 69–79.

[67] George Karypis and Vipin Kumar. "A fast and high quality multilevel scheme for partitioning irregular graphs". In: *SIAM Journal on scientific Computing* 20.1 (1998), pp. 359–392.

[68] M. N. Katehakis and A. F. Veinott Jr. "Multi-Armed Bandit Problem: Decomposition and Computation". In: *Mathematics of Operations Research* 12.2 (1987), pp. 262–268.

[69] A. Khan, F. Bonchi, A. Gionis, and F. Gullo. "Fast Reliability Search in Uncertain Graphs". In: *Proc. EDBT Conf.* 2014, pp. 535–546.

[70] A. Khan, F. Bonchi, F. Gullo, and A. Nufer. "Conditional Reliability in Uncertain Graphs". In: *IEEE TKDE* 30.11 (2018), pp. 2078–2092.

[71] A. Khan, Y. Ye, and L. Chen. *On Uncertain Graphs*. Synthesis Lectures on Data Management. Morgan & Claypool, 2018.

[72] J. Kim and J.-G. Lee. "Community Detection in Multi-Layer Graphs: A Survey". In: *SIGMOD Record* 44.3 (2015), pp. 37–48.

[73] Jungeun Kim, Jae-Gil Lee, and Sungsu Lim. "Differential flattening: A novel framework for community detection in multi-layer graphs". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.2 (2016), pp. 1–23.

[74] M. Kivela, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter. "Multilayer networks". In: *Journal of Complex Networks* 2.3 (2014), pp. 203–271.

[75] G. Kollios, M. Potamias, and E. Terzi. "Clustering Large Probabilistic Graphs". In: *IEEE TKDE* 25.2 (2013), pp. 325–336.

[76] Z. Kuncheva and G. Montana. "Community Detection in Multiplex Networks using Locally Adaptive Random Walks". In: *Proc. ASONAM*. 2015, pp. 1308–1315.

[77] A. Lancichinetti and S. Fortunato. "Consensus clustering in complex networks". In: *Sci. Rep.* 2 (2012), p. 336.

[78] D. LaSalle and G. Karypis. "Multi-threaded modularity based graph clustering using the multilevel paradigm". In: *J. Parallel Distrib. Comput.* 76 (2015), pp. 66–80.

[79] J. Leskovec, D. P. Huttenlocher, and J. M. Kleinberg. "Governance in Social Media: A Case Study of the Wikipedia Promotion Process". In: *Proceedings of the International AAAI Conference on Web and Social Media*. 2010.

[80] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. "Statistical properties of community structure in large social and information networks". In: *Proceedings of the 17th international conference on World Wide Web*. 2008, pp. 695–704.

[81] Y. Li, X. Kong, C. Jia, and J. Li. "Clustering Uncertain Graphs with Node Attributes". In: *Proc. ACML Conf.* 2018, pp. 232–247.

[82] H. Liu, E. Lim, H. W. Lauw, M. Le, A. Sun, J. Srivastava, and Y. A. Kim. "Predicting trusts among users of online communities: an epinions case study". In: 2008, pp. 310–319.

[83] L. Liu, R. Jin, C. C. Aggarwal, and Y. Shen. "Reliable Clustering on Uncertain Graphs". In: *Proc. IEEE ICDM Conf.* 2012, pp. 459–468.

[84] James MacQueen et al. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Vol. 1. 14. Oakland, CA, USA. 1967, pp. 281–297.

[85] Matteo Magnani, Obaida Hanteer, Roberto Interdonato, Luca Rossi, and Andrea Tagarelli. "Community Detection in Multiplex Networks". In: *arXiv preprint arXiv:1910.07646* (2019).

[86] D. Mandaglio, A. Amelio, and A. Tagarelli. "Consensus Community Detection in Multilayer Networks Using Parameter-Free Graph Pruning". In: *Proc. Pacific-Asia Conf. on Advances in Knowledge Discovery and Data Mining (PAKDD)*. 2018, pp. 193–205.

[87] P. Massa and P. Avesani. "Controversial Users Demand Local Trust Metrics: An Experimental Study on Epinions.com Community". In: *Proc. Nat.l Conf. on Artificial Intelligence (AAAI)*. 2005, pp. 121–126.

[88] P. Massa and P. Avesani. "Controversial Users Demand Local Trust Metrics: An Experimental Study on Epinions.com Community". In: *Proc. of AAAI Conf. on Artificial Intelligence (AAAI)*. 2005, pp. 121–126.

[89] Rossana Mastrandrea, Tiziano Squartini, Giorgio Fagiolo, and Diego Garlaschelli. "Enhanced reconstruction of weighted networks from strengths and degrees". In: *New Journal of Physics* 16 (2014).

[90] A. E. Mislove. "Online social networks: measurement, analysis, and applications to distributed information systems". PhD thesis. Rice University, 2009.

[91] Raul J Mondragon, Jacopo Iacovacci, and Ginestra Bianconi. "Multilink communities of multiplex networks". In: *PloS one* 13.3 (2018), e0193821.

[92] Hervi Moulin. *Axioms of cooperative decision making*. 15. Cambridge university press, 1991.

[93] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela. "Community structure in time-dependent, multiscale, and multiplex networks". In: *Science* 328.5980 (2010), pp. 876–878.

[94] S. Nepal, W. Sherchan, and C. Paris. "STrust: A Trust Model for Social Networks". In: 2011, pp. 841–846.

[95] M. E. J. Newman. "Fast Algorithm for Detecting Community Structure in Networks". In: *Phys. Rev. E* 69 (2004).

[96] M. E. J. Newman and M. Girvan. "Finding and evaluating community structure in networks". In: *Phys. Rev. E* 69.2 (2004), p. 026113.

[97] Nam Nguyen and Rich Caruana. "Consensus clusterings". In: *Seventh IEEE international conference on data mining (ICDM 2007)*. IEEE. 2007, pp. 607–612.

[98] F. Å. Nielsen. "A New ANEW: Evaluation of a Word List for Sentiment Analysis in Microblogs". In: *Proc. of the ESWC2011 Workshop on 'Making Sense of Microposts'*. 2011, pp. 93–98.

[99] F. J. Ortega, J. A. Troyano, F. L. Cruz, C. G. Vallejo, and F. Enríquez. "Propagation of trust and distrust for the detection of trolls in a social network". In: *Computer Networks* 56.12 (2012), pp. 2884–2895.

[100] D. Pandove, S. Goel, and R. Rani. "Correlation clustering methodologies and their fundamental results". In: *Expert Systems* 35.1 (2018).

[101] P. Parchas, F. Gullo, D. Papadias, and F. Bonchi. "Uncertain Graph Processing through Representative Instances". In: *ACM TODS* 40.3 (2015), 20:1–20:39.

[102] Alex Pothen, Horst D Simon, and Kang-Pu Liou. "Partitioning sparse matrices with eigenvectors of graphs". In: *SIAM journal on matrix analysis and applications* 11.3 (1990), pp. 430–452.

[103] Filippo Radicchi, Jose J. Ramasco, and Santo Fortunato. "Information filtering in complex weighted networks". In: *Physical Review E* 83 (2011), p. 046101.

[104] Jörg Reichardt and Stefan Bornholdt. "Detecting fuzzy community structures in complex networks with a Potts model". In: *Physical Review Letters* 93.21 (2004), p. 218701.

[105] G. Rossetti. "RDyn: graph benchmark handling community dynamics". In: *Journal of Complex Networks* (2017).

[106] G. Rossetti, L. Pappalardo, D. Pedreschi, and F. Giannotti. "Tiles: an online algorithm for community discovery in dynamic social networks". In: *Machine Learning* 106.8 (2017), pp. 1213–1241.

[107] Martin Rosvall and Carl T. Bergstrom. "Maps of random walks on complex networks reveal community structure". In: *Proceedings of the National Academy of Sciences* 105.4 (2008), pp. 1118–1123.

[108] M Ángeles Serrano, Marián Boguná, and Alessandro Vespignani. "Extracting the multiscale backbone of complex weighted networks". In: *Proceedings of the National Academy of Sciences* 106.16 (2009), pp. 6483–6488.

[109] W. Sherchan, S. Nepal, and C. Paris. "A survey of trust in social networks". In: *ACM Comput. Surv.* 45.4 (2013), 47:1–47:33.

[110] A. Strehl and J. Ghosh. "Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions". In: *J. Mach. Learn. Res.* 3 (2003), pp. 583–617.

[111] A. Strehl and J. Ghosh. "Cluster ensembles—a knowledge reuse framework for combining multiple partitions". In: *J. Mach. Learn. Res.* 3.Dec (2002), pp. 583–617.

[112] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*. Vol. 135. Cambridge: MIT press, 1998.

[113] C. Swamy. "Correlation Clustering: maximizing agreements via semidefinite programming". In: *Proc. ACM-SIAM SODA Conf.* 2004, pp. 526–527.

[114] Andrea Tagarelli, Alessia Amelio, and Francesco Gullo. "Ensemble-based community detection in multilayer networks". In: *Data Min. Knowl. Discov.* 31.5 (2017), pp. 1506–1543.

[115] M. Takaffoli, R. Rabbany, and O. R. Zaïane. "Community evolution prediction in dynamic social networks". In: *Proc. IEEE/ACM Int. Conf. on Advances in Social Networks Analysis and Mining (ASONAM)*. 2014, pp. 9–16.

[116] J. Tang, H. Gao, H. Liu, and A. Das Sarma. "eTrust: understanding trust evolution in an online world". In: *Proc. ACM KDD Conf.* 2012, pp. 253–261.

[117] L. Tang and H. Liu. *Community Detection and Mining in Social Media*. Synthesis Lectures on Data Mining and Knowledge Discovery. Morgan & Claypool Publishers, 2010.

[118] L. Tang, X. Wang, and H. Liu. "Community detection via heterogeneous interaction analysis". In: *Data Min. Knowl. Discov.* 25 (2012), pp. 1–33.

[119] L. Tang, X. Wang, and H. Liu. "Uncovering Groups via Heterogeneous Interaction Analysis". In: *Proc. ICDM*. 2009, pp. 503–512.

[120] Nazanin Afsarmanesh Tehrani and Matteo Magnani. "Partial and Overlapping Community Detection in Multiplex Social Networks". In: *Social Informatics - 10th International Conference, SocInfo 2018, St. Petersburg, Russia, September 25-28, 2018, Proceedings, Part II*. Ed. by Steffen Staab, Olessia Koltsova, and Dmitry I. Ignatov. Vol. 11186. Lecture Notes in Computer Science. Springer, 2018, pp. 15–28.

[121] V. A. Traag and J. Bruggeman. "Community detection in networks with positive and negative links". In: *Physical Review E* 80.3 (2009), p. 036115.

[122] B. Viswanath, A. Mislove, M. Cha, and P. Krishna Gummadi. "On the evolution of user interaction in Facebook". In: *Proc. ACM Workshop on Online Social Networks (WOSN)*. 2009, pp. 37–42.

[123] Z. Wang, Z. Li, G. Yuan, Y. Sun, X. Rui, and X. Xiang. "Tracking the evolution of overlapping communities in dynamic social networks". In: *Knowledge-Based Systems* 157 (2018), pp. 81–97.

[124] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. "Overlapping community detection in networks: The state-of-the-art and comparative study". In: *Acm computing surveys (csur)* 45.4 (2013), pp. 1–35.

[125] K. Yang, Q. Guo, and J.-G. Liu. "Community detection via measuring the strength between nodes for dynamic networks". In: *Physica A: Statistical Mechanics and its Applications* 509 (2018), pp. 256–264.

[126] Y. Yao, H. Tong, F. Xu, and J. Lu. "Subgraph Extraction for Trust Inference in Social Networks". In: *Encyclopedia of Social Network Analysis and Mining, 2nd Edition*. 2018.

[127] A. Zakrzewska and D. A. Bader. "Tracking local communities in streaming graphs with a dynamic algorithm". In: *Social Netw. Analys. Mining* 6.1 (2016), 65:1–65:16.

[128] H. Zhang, C.-D. Wang, J.-H. Lai, and P. S. Yu. "Modularity in Complex Multilayer Networks with Multiple Aspects: A Static Perspective". In: *CoRR* abs/1605.06190 (2016).