CLUSTERING ATTRIBUTED MULTI-GRAPHS

Andreas Papadopoulos

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Cyprus

Recommended for Acceptance

by the Department of Computer Science

May, 2017

APPROVAL PAGE

Doctor of Philosophy Dissertation

CLUSTERING ATTRIBUTED MULTI-GRAPHS

Presented by

Andreas Papadopoulos

Research Supervisor ______________________
George Pallis

Committee Member ______________________
Marios D. Dikaiakos

Committee Member ______________________
Constantinos S. Pattichis

Committee Member ______________________
Nick Bassiliades

Committee Member ______________________
Fragkiskos Papadopoulos

University of Cyprus

May, 2017

DECLARATION PAGE

Doctor of Philosophy Dissertation

CLUSTERING ATTRIBUTED MULTI-GRAPHS

The present doctoral dissertation was submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy of the University of Cyprus. It is a product of original work of my own, unless otherwise mentioned through references, notes, or any other statements.

Andreas Papadopoulos

University of Cyprus

May, 2017

ΟΜΑΔΟΠΟΙΗΣΗ ΑΝΤΙΚΕΙΜΕΝΩΝ ΣΕ ΔΙΚΤΥΑ ΠΛΗΡΟΦΟΡΙΩΝ

Ανδρέας Παπαδόπουλος

Πανεπιστήμιο Κύπρου, 2017

Ένα δικτύο πληροφοριών (information networks) μπορεί να μοντελοποιηθεί αποτελεσματικά ως ένας πολυγράφος ιδιοτήτων (attributed multi-graph). Σε ένα πολυγράφο ιδιοτήτων μια κορυφή (vertex) αντιπροσωπεύει ένα αντικείμενο. Κάθε κορυφή χαρακτηρίζεται από κάποιες τιμές που αντιστοιχούν στις ιδιότητες του αντικειμένου που αντιπροσωπεύει. Οι συνδέσεις αναπαριστούν τις σχέσεις που έχουν τα αντικείμενα.

Η ομαδοποίηση (clustering) αποσκοπεί στη διαίρεση των αντικειμένων σε ομάδες βάση διαφόρων κριτηρίων. Σε πραγματικά δίκτυα πληροφοριών, κάθε χαρακτηριστικό των αντικειμένων, π.χ. ιδιότητες και τύποι συνδέσεων, περιέχει διαφορετική πληροφορία, και ορισμένα από αυτά τα χαρακτηριστικά μπορεί να μην είναι χρήσιμα στην διαδικασία ομαδοποίησης. Επομένως, πρέπει να προσδιορίσουμε πόσο σημαντική είναι η κάθε ιδιότητας και ο κάθε τύπος ακμής. Όταν η διαδικασία ομαδοποίησης λαμβάνει υπόψη το πόσο σημαντικά είναι τα χαρακτηριστικά των αντικειμένων επιτυγχάνει αποτελέσματα υψηλής ποιότητας.

Πολλές υπάρχουσες μέθοδοι ομαδοποίησης αντικειμένων/κόμβων σε γράφους ιδιοτήτων θεωρούν ότι οι ιδιότητες των αντικειμένων είναι το ίδιο σημαντικές ή αγνοούν

την ύπαρξη συνδέσεων πολλαπλών τύπων. Επίσης, ανακαλύπτουν ομάδες που χαρακτηρίζονται από ομοιογένεια χαρακτηριστικών και είναι πυκνά συνδεδεμένες (densely connected components). Ωστόσο, η αναγνώριση ομάδων αντικειμένων που μοιράζονται παρόμοιες συνδέσεις είναι επίσης σημαντική.

Προτείνουμε μια συλλογή καινοτόμων μεθόδων για την εύρεση ομάδων σε ένα δίκτυο πληροφοριών που μοντελοποιείται ως ένας πολυγράφος ιδιοτήτων. Οι προτεινόμενες μέθοδοι μπορούν να εκμεταλλευτούν την υπολογιστική ισχύ των σύγχρονων πολυπύρηνων υπολογιστών έτσι ώστε να είναι ικανές να χειριστούν μεγάλα σύνολα δεδομένων. Προτείνουμε συναρτήσεις ενοποιημένης ομοιότητας ή απόστασης που συνδυάζουν αποτελεσματικά τα χαρακτηριστικά των κορυφών και σχεδιάζουμε μηχανισμούς στάθμισης. Οι προτεινόμενες μέθοδοι ομαδοποίησης προσδιορίζουν τη σημαντικότητα κάθε χαρακτηριστικού (ιδιότητας και τύπου ακμών) των αντικειμένων χρησιμοποιώντας αυτούς τους μηχανισμούς στάθμισης, και έτσι εξισορροπούν και συνδυάζουν αποτελεσματικά όλα τα χαρακτηριστικά των κορυφών. Το αποτέλεσμα είναι η βελτίωση της ποιότητας της ομαδοποίησης με βάση διάφορα ευρέως αποδεκτά μέτρα αξιολόγησης αλγορίθμων ομαδοποίησης. Ο στόχος μας είναι να μεγιστοποιήσουμε την ενοποιημένη ομοιότητα μεταξύ των κορυφών που ανήκουν στην ίδια ομάδα. Τα μέλη μιας ομάδας πρέπει να έχουν υψηλή παρόμοια συνδεσιμότητα (similar connectivity), δηλαδή να σχετίζονται/συνδέονται με τις ίδιες κορυφές. Επίσης, πρέπει να χαρακτηρίζονται από κοντινές τιμές ιδιοτήτων (χαμηλή εντροπία). Είμαστε από τους πρώτους που βρίσκουμε ομάδες με παρόμοια συνδεσιμότητα σε πολυγράφους ιδιοτήτων.

Επιπλέον, αξιοποιούμε τις προτεινόμενες μεθόδους για την επίλυση ενός πρακτικού ζητήματος. Συγκεκριμένα, αναπτύσσουμε ένα σύστημα το οποίο προσφέρει αξιόπιστες συστάσεις (recommendations) σε ευρωπαϊκούς οργανισμούς για καινούριες συνεργασίες. Για να το επιτύχουμε αυτό ομαδοποιούμε τους οργανισμούς που έχουν συμμετάσχει σε έργα που χρηματοδοτήθηκαν από την Ευρωπαϊκή Ένωση στο πλαίσιο του προγράμματος Horizon 2020. Στην συνέχεια, προτείνουμε μια μέθοδο που αναλύει τα αποτελέσματα της ομαδοποίησης για να εξάγει πιθανές συνεργασίες. Οι οργανισμοί και οι ερευνητές μπορούν να χρησιμοποιήσουν το σύστημά μας για να για να εντοπίσουν νέους συνεργάτες. Από όσο γνωρίζουμε, το σύστημα μας είναι το πρώτο που προσφέρει τέτοιες υπηρεσίες στην κοινότητα.

CLUSTERING ATTRIBUTED MULTI-GRAPHS

Andreas Papadopoulos

University of Cyprus, 2017

An attributed multigraph is a structure that efficiently represents real world networks. In an attributed graph, a vertex represents an object. A vertex is characterized by some attributes corresponding to the object's properties. Edges capture the objects' relationships.

Clustering aims to partition the objects into groups, namely clusters, based on various criteria. In real world networks each object property, i.e. attribute and edge-type, contains different information. Some of these properties may be irrelevant to the clustering task. Hence, we must identify the significance of each attribute and edge-type. Clustering process must consider the significance of vertex properties to achieve high-quality results.

Many existing attributed graph clustering methods assume the vertex properties are equally important, or they ignore that many edge types exist. Also, they discover clusters characterized by attribute homogeneity that form densely connected components. Yet, identifying clusters of objects that share similar connections is also important.

We propose a collection of novel methods to detect clusters in an attributed multigraph. Proposed methods can exploit the computational power of modern

multicore architectures. Hence, they can handle large datasets. We propose unified similarity or distance functions that efficiently combine the various vertex properties. We additionally design weighting mechanisms. The proposed methods identify the importance of each vertex property using these mechanisms. They so balance and combine the vertex properties efficiently. The result is the improvement of clustering quality in terms of various evaluation measures. Our goal is to maximize the unified similarity among vertices in the same cluster. Cluster members must have high similar connectivity, i.e. relate/connect to the same vertices. Also, they must be characterized by close attribute values (low entropy). To the best of our knowledge, we are among the first to optimize similar connectivity.

Moreover, we leverage proposed methods to solve a practical issue. That is, how to offer reliable, evidence-based recommendations to European organizations. To do so, we cluster the European research activities network. That is, the network of all organizations that participated to projects funded by the European Union. We propose a clustering-based recommendation method that analyzes the clustering results to provide recommendations. Organization and researchers can use our system to establish new collaborations. To the best of our knowledge, this is the first system to offer such services to the community.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank to my Ph.D. advisors, Professors George Pallis and Marios D. Dikaiakos, for supporting me during the years of my doctoral studies. George Pallis is someone you will never forget once you meet him. He is a respectable and affable scholar, who has provided me with unreserved support and invaluable guidance in every step during my doctoral studies. Marios Dikaiakos has been supportive and gave me the most insightful guidance. The enlightening discussions with both of them during my doctoral studies are priceless. They not only guide me to qualify as a Ph.D., but also give me valuable pieces of advice for the future success. I would also like to thank the members of my Ph.D. committee Professors Constantinos S. Pattichis, Nick Bassiliades and Fragkiskos Papadopoulos for those constructive comments on my dissertation.

I would also like to take the opportunity to gratefully acknowledge the support I have received from: the Cyprus state scholarship foundation, the Telerehabilitation project (co-financed by the European Union and national funds of Greece and Cyprus, Cross Border Cooperation Programme Greece Cyprus 2007-2013), the EU Commission in terms of the PaaSport 605193 FP7 project (FP7-SME-2013), and the Department of Computer Science at University of Cyprus.

I am deeply grateful to the supervisor of my bachelor's and master's theses, Professor Dimitrios Katsaros, who introduced me to research and encouraged me to apply for a Ph.D.. My appreciation also goes to my colleges and friends (too many to list here but they know who they are) for their support and the extensive fruitful discussions we had during my studies at University of Cyprus. I would also like to acknowledge Andreas Andreou for implementing the web interface of the recommended system.

Finally, words cannot express my gratitude to my family for believing in me and their constant support and encouragement. This thesis would not have been possible without the support and encouragement of my wife, Ivi, who is always beside me, days and nights since my bachelor studies, helping me overcome various hurdles.

I dedicate this thesis to my family, my wife, Ivi, and my kids, Stephanos and Angelos, for their unconditional love, support and encouragement.

CREDITS

1. Andreas Papadopoulos, George Pallis, and Marios D. Dikaiakos, "Weighted clustering of attributed multi-graphs," Springer journal on Computing, pages 1-28, 2016.

2. Andreas Papadopoulos, Dimitrios Rafailidis, George Pallis, and Marios D. Dikaiakos, "Clustering attributed multi-graphs with information ranking," in Database and Expert Systems Applications, Lecture Notes in Computer Science. Springer International Publishing, Volume 9261, Pages: 432-446, ISBN: 978-3-319-22848-8, 2015.

3. Andreas Papadopoulos, George Pallis, and Marios D. Dikaiakos, "Identifying clusters with attribute homogeneity and similar connectivity in information networks," in IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Volume 1, Pages: 343-350, ISBN: 978-1-4799-2902-3, 2013.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Nowadays many datasets are becoming publicly available. Yet, a dataset on its own is unlikely to be useful since users need more information. Data mining extracts useful, and before unknown, knowledge from a dataset. This thesis focuses on the data mining task of clustering. Clustering is an unsupervised data mining technique which aims at finding groups/clusters of related data [3, 42]. An unsupervised technique does not need apriori knowledge of labeled samples. Clusters may be the input to other algorithms for further analysis. Cluster analysis provides insights on the data.

Traditionally, datasets are represented as vectors, (objects are modeled as points) or graphs (objects are modeled as graph vertices and object-relationships are modeled as edges). In many real-world datasets from various domains such as biology, telecommunications, software engineering and social networking, both types of information are exposed. For instance, a social network consists of people connected to each other by relationships such as friendships and family, while each person is characterized by his profile, i.e. his interests, gender, education.

Generally speaking, *an attributed multi-graph can represent any real world information network*. In an attributed graph, vertices represent homogeneous objects. Each vertex is characterized by some attributes corresponding to the object's properties. Edges capture the objects' relationships. Thus, attributed graphs represent both structural and attribute properties of the objects. An attributed multi-graph is an attributed graph where a pair of vertices may be connected by more than one edges. Hence, it also captures multiple relationships, i.e. interactions, among objects. For instance, to model a co-authorship network we represent each author by a vertex. Some attributes like 'area of interest' and 'number of publications' characterize each vertex. The edges represent co-authorship and friendship relationships. Co-authorship and friendship are two different types of relationships.

1.1 Clustering Attributed Multi-graphs

The identification of clusters in an attributed multi-graph is a challenging problem with a lot of interesting applications. For instance, we cluster a social network to offer targeted advertisements or recommend friendships; a co-authorship network to suggest new collaborations or identify influential authors; the clients of a mobile telephony company to provide recommendations on cell towers placement, and so on. Clustering attributed multi-graphs faces the following challenges:

Nowadays, *networks can be huge.* The World Wide Web has at least nine billion pages. Each page is described by its context such as topic, author, publish date e.t.c. and links to other pages. Online social networks, such as Facebook and Twitter, count more than two billion users. Biologists are gathering more and more genomic data at lower costs [6, 94]. These huge amounts of data demand efficient clustering methods capable of extracting useful information.

Moreover, it is not clear *how to balance the structural and the attribute information.* A naive approach is to assume equal importance between them [101]. Although it simplifies things, it is not realistic. As mentioned by Moser and Ester: "*often vertex attributes and edges contain complementary information, i.e. neither the relationships can be derived from the vertex attributes nor vice versa. In such scenarios, the simultaneous use of both data types promises more meaningful and accurate results*" [61].

In real-world networks, *relationships and attributes contain different information.* Some of these may be irrelevant and introduce noise to the clustering task [24, 80]. For example, when clustering a co-authorship network we can argue that the attribute 'area of interest' is more important than authors gender. Similarly, co-author relations are more important than authors friendships and gender.

In addition, structural information is often used to detect densely connected components [24, 90]. Yet, identifying clusters with similar connectivity may be more important than identifying dense structures [5, 8]. For instance, in a social network, unconnected people may have a lot of common friends. Although they do not know each other, they have *high similar connectivity.* Groups of people with such similarities can be used for friend recommendations. Similarly, software files from the same software package do not form densely connected component [44]. Hence, identifying densely connected components on a software files graph is rather useless [44, 55, 97].

Another important property of real-world networks is that objects are usually not completely well separated, i.e. an object may be related-similar to objects in multiple clusters[4]. Thus, we should allow multiple memberships. *Fuzzy clustering* identifies overlapping clusters in which an object belongs with a membership probability. Membership probabilities can provide insights on the data, i.e. importance of an object in a cluster [33, 48].

1.2 Contributions of this Thesis

This thesis focuses on the problem of clustering information networks modeled as attributed multi-graphs and introduces novel methods that efficiently tackle the aforementioned challenges. We summarize the major contributions of this thesis as follows.

We introduce a data model that efficiently represents an attributed multi-graph and we formerly define the clustering problem. We approach solutions to the problem of attributed multi-graph clustering through the *definition and optimization (either minimization or maximization) of specific objective functions.* The goal is to maximize the unified similarity between objects in the same cluster. Unified similarity is computed by functions that combine both structural and attribute properties of the vertices. Due to the high complexity of the problem (NP-Hard) we adopt techniques that are not ideal but converge quickly to local optima and yield high-quality clustering. Particularly, we adopt gradient descent and spectral clustering techniques.

We survey related work on clustering attributed graphs and multi-graphs. Specifically, we give a summary and qualitative comparison of state-of-the-art attributed graph clustering methods. Also, we outline several non-trivial challenges of clustering attributed multi-graphs.

We develop a collection of novel methods to detect clusters in an attributed multi-graph. Proposed methods can exploit the computational power of modern multi-core architectures. Hence, they can handle large datasets. We propose unified similarity or distance functions that efficiently combine the various vertex properties. We additionally design weighting mechanisms. The proposed methods identify the importance of each vertex property using these mechanisms. They so balance and combine the vertex properties efficiently. The result is the improvement of clustering quality in terms of various evaluation measures. Our goal is to maximize the unified similarity among vertices in the same cluster. Cluster members must have *high similar connectivity*, i.e. relate/connect to the same vertices. Also, they must be characterized by *close attribute values* (low entropy). To the best of our knowledge, we are among the first to optimize similar connectivity instead of measures such as density and modularity that are being traditionally optimized.

We firstly present a heuristic approach, HASCOP (Homogeneous Attributes and Similar COnnectivity Patterns), which optimizes both attribute homogeneity and similar connectivity. HASCOP identifies the importance of vertex properties based on a scoring mechanism. Particularly, an edge-type or an attribute is considered more important and is given a higher score if vertices in current clusters interconnect by edges of this edge-type or share the same values for the specific attribute, respectively. HASCOP applies to attributed multi-graphs and automatically identifies the different importance of both edge-types and attributes. A key advantage of HASCOP is that it does not need any user-specified input such as the number of clusters.

Secondly, we present CAMIR (Clustering Attributed Multi-graphs with Information Ranking) which is based on spectral clustering. Initially, we rank the vertex properties according to the similarity of the resulted clustering if used independently. We avoid calculating all possible clusterings by adopting a co-regularization technique. Based on the properties ranking we construct a unified similarity matrix on which we apply a spectral clustering technique to generate the final clusters. CAMIR weight identification process is decoupled from the clustering task and, thus for a given dataset, it can take place only once. Because CAMIR follows a spectral clustering approach, it identifies clusters of arbitrary shapes and sizes.

Thirdly, we present CLAMP (CLustering Attributed Multi-graPhs) that also optimizes attribute homogeneity and similar connectivity. CLAMP considers simultaneously the individual importance of the attributes and edge-types as well as the balance between the sets of attributes and edges, by assigning them different weights that are identified during the clustering process in an automatic manner. CLAMP is to the best of the authors' knowledge, the first to perform fuzzy clustering on weighted directed attributed multi-graphs with heterogeneous attributes.

We developed an experimental framework for attributed graph clustering, which includes implementing attributed graph and multigraph clustering algorithms; collecting and generating real-world and synthetic datasets; and evaluating acquired clusterings. Our extensive experimental evaluation on synthetic datasets and a diverse collection of real-world information networks (bibliography items, software packages, research and innovation projects funded by the European Union) against the state-of-the-art attributed graph clustering approaches: (a) confirms that using weighting scheme improves results quality; and (b) demonstrates the efficiency and effectiveness of the proposed approaches in terms of similar connectivity and attribute homogeneity under various scenarios. Moreover, to the best of our knowledge, we are the first to study clustering results on the network of organizations participated in projects funded by the European Union.[1]

Moreover, we leverage proposed methods to solve a practical issue. That is, how to offer reliable, evidence-based recommendations to European organizations. We propose a clustering-based recommendation method for the European research

[1]The dataset is available online at EU Open Data Portal - `http://open-data.europa.eu`

activities network. That is the network of all organizations participated in R&D projects funded by the European Union under Horizon 2020 programme. [2] We integrate the proposed recommendation approach in a web-based system. Our system imports updates of the European research activities network from the web automatically. It then applies the proposed recommendation method and analyses the results to provide recommendations. Organization and researchers can use our system to identify possible partners to establish new collaborations. To the best of our knowledge, this is the first system to offer such services to the community.

[2]Horizon 2020 is the biggest EU Research and Innovation programme ever with nearly €80 billion of funding available over 7 years (2014 to 2020). `https://ec.europa.eu/programmes/horizon2020/`

Chapter 2

Motivation

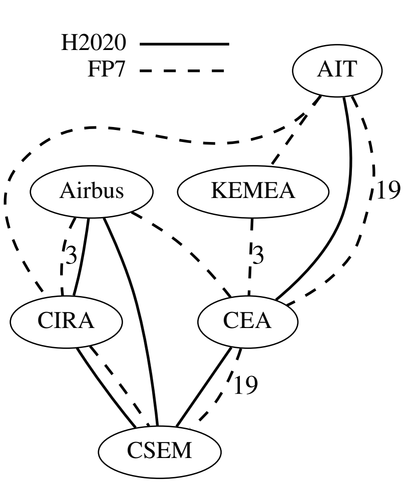
An attributed multi-graph is an expressive data structure able to represent real world information networks from various domains. Clustering objects on real world networks is being widely used for a lot of purposes. Most of the times the goal of clustering depends on the application and the network.

Despite the plethora of practical real-world applications, clustering attributed multi-graphs is an interesting challenging problem. For example, more than 130000 research projects have been funded by the European Union under H2020 and FP7 programmes. Figure (1) presents an exemplary attributed multigraph modeling a sample of the network. A vertex is characterized by some attributes (Table (1)) and represents an organization that participated to projects funded by the European Union. Collaborations on FP7 and H2020 projects are modeled by weighted edges of different type in Figure (1a).

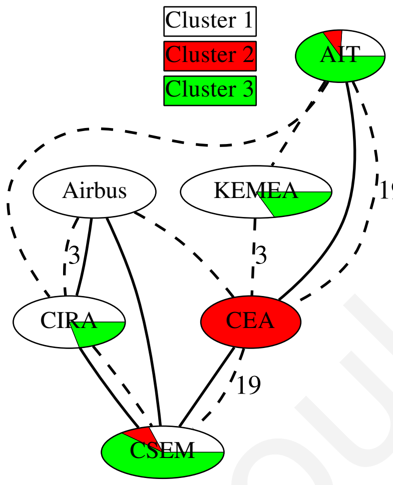
Clustering this collaborations network is practically useful in recommending new partnerships and in detecting outliers. Yet, we have to exploit the information from both the heterogeneous organizations' attributes and the multiple collaborations.

Recently, there has been a lot of research in the area of attributed graph clustering [13, 19] aiming to combine structural and attribute properties and consequently improve clustering quality [34, 76]. However, these methods [8, 23, 91, 93] cannot be directly applied to an attributed multi-graph such as the network of Figure (1), because they either ignore the multiple types of edges and/or deal with only one attribute type. To overcome this, these methods must project an attributed multi-graph to an attributed graph with homogeneous attributes. Still, a projection results in information loss and consequently limits clustering accuracy. For example, by discretizing the attribute 'number of H2020 projects' to two bins we cannot distinguish if an organization participated in 1, 7 or 9 projects. Similarly, by transforming the multi-graph to a graph, i.e. by summing the weights of multiple edges, we discard collaborations.

Moreover, majority of existing methods [23, 91, 93] detect densely connected components while identification of clusters of objects having similar connectivity may be of even greater importance [8, 63]. For instance, in Figure (1a) AIT and CSEM share common partners (high similar connectivity) and close attributes. Since these two organizations have not collaborated in the past we can recommend a new collaboration. However, they are not a densely connected component and thus would not be clustered together. Specifically, methods that seek to identify densely



(a) Organizations Collaboration Network. Weights denote the number of collaborations (unlabeled edges have weight 1).



(b) Fuzzy Clustering of the Network

Figure 1: A sample of the European research activities network modeled as attributed multi-graph. Each vertex represents an organization characterized by the attributes shown in Table 1, and Figure (a) shows the organizations collaboration network. A weighted edge indicates the number of collaborations. Figure (b) depicts a fuzzy clustering of the organizations into three clusters (white, red, green). A vertex may belong to multiple clusters according to a probability represented by a circle share.

connected components would identify the clusters: {Airbus, CIRA, CSEM} and {AIT, KEMEA, CEA}. However, these clusters do not provide many insights, especially for recommending new collaborations since organizations in the same cluster have already collaborated.

Another important property of the collaboration network in Figure (1) is that organizations are usually not completely well separated, i.e. an organization may be related-similar to organizations in multiple clusters. However, many of existing methods such as PICS [8], SA-Cluster [23] and BAGC [91] perform hard clustering instead of allowing multiple memberships, i.e. fuzzy clustering that identifies

Table 1: Organization Attributes

Short Name	**Name**	**FP7 projects**	**H2020 projects**	**Country**
Airbus	Airbus SAS	14	1	France
AIT	Austrian Institute of Technology	137	9	Austria
KEMEA	Center for Security Studies	32	0	Greece
CIRA	Centro Italiano Ricerche Aerospaziali	44	1	Italy
CEA	Commissariat a l'energie atomique et aux energies alternatives	611	39	France
CSEM	Swiss Center for Electronics and Microtechnology	96	7	Switzerland

overlapping clusters in which an object belongs with a membership probability.[1] Figure (1b) demonstrates a fuzzy clustering by CLAMP proposed method where each vertex slice represents the object's probability belonging to the respective cluster. For example, we observe that KEMEA is highly related to Airbus and CIRA, because of cluster 1; and it is also related to AIT and CSEM, because of cluster 3. Thus, we can recommend some of these organizations as partners to KEMEA by leveraging the membership probabilities.

Moreover, object properties, i.e. attributes and edge-types, contain different information and some may be irrelevant to the clustering task. For instance, to cluster the collaboration network of Figure (1) the attributes 'number of FP7' and 'number of H2020' projects are important, while attributes 'number of employees' and 'year of establishment' may introduce noise. One way to capture this challenging

[1]Similarly, overlapping clustering assigns an object to multiple clusters with binary memberships [21, 93]. Though, membership probabilities provide more information, i.e. importance of an object in a cluster [33, 48].

aspect is to assign proper weights to each object property (attribute and edge-type). By doing so each object property is considered differently in the clustering. However, tuning such weights is a difficult task requiring apriori knowledge or costly preprocessing of the information network under study.

In this thesis we focus on the problem of clustering information networks modeled as attributed multi-graphs, and we introduce novel methods that efficiently tackle the aforementioned challenges and automatically tune the weights.

2.1 Roadmap

To this end, Chapter 3 introduces the data model and the notations used throughout the thesis, and formally defines the attributed multi-graph clustering problem. Chapter 4 surveys related work. Chapters 5, 6, 7 describe the proposed methods, which are extensively evaluated in Chapter 8 on both real-world and synthetic datasets. Chapter 9 demonstrates a use-case and confirms the applicability of proposed methods to real-world problems. Finally, Chapter 10 concludes this thesis and presents some future research directions.

Chapter 3

Problem Statement and Preliminary Concepts

In this section, we introduce our notations, present the attributed multi-graph data model used in this thesis, and define the problem clustering attributed multi-graphs. Due to the high complexity of the problem we adopt approximation techniques which we also discuss in this section. These techniques converge quickly to high quality clustering.

Our notation is presented in Table 2. Following standard notations, sets are denoted by calligraphic upper case letters, e.g., $\mathcal{A}$; matrices by plain upper case letters, e.g., A; functions by lower case calligraphic letters, e.g., $\mathscr{a}$; and set elements by lower case letters, e.g., a.

3.1 Data Structure

An **information network** can be modeled as a directed weighted attributed multi-graph G where:

- $\mathcal{V} = \{v_i\}$ is the set of vertices

Table 2: Notations.

Symbol	**Description**
$\mathcal{V}$	Set of vertices $\mathcal{V} = \{v_i\}$. $\lvert\mathcal{V}\rvert$ is the number of vertices
$\mathcal{E}_t$	Set of edges of type t. $\mathcal{E}_t = \{(v_i, v_j) : v_i, v_j \in \mathcal{V}\}$
$\omega_t : \mathcal{E}_t \to (0, 1]$	Function that returns the weight of the edges of type t
$\mathcal{A}$	Set of attributes $\mathcal{A} = \{\alpha\}$
$\mathcal{D}_\alpha$	The range of attribute α, i.e. all possible values of attribute α
$a_\alpha(v_i)$	Value of vertex v_i on attribute α
$\mathcal{T}$	Set of edge-types, with $\mathcal{T} = \{t\}$
$\mathcal{P} = \{\mathcal{A} \cup \mathcal{T}\}$	Set of all vertex properties (attributes and edge-types)
$w : \mathcal{P} \to (0, 1]$	Function that returns the importance/weight of a vertex property p
K	The number of clusters, $K \geq 2$
$\mathcal{C}_j$	Cluster j, $1 \leq j \leq K$
Θ	A $\lvert\mathcal{V}\rvert \times K$ matrix where $\theta_{i,j}$ is the probability of vertex v_i belonging to cluster $\mathcal{C}_j$

- $\mathcal{T} = \{t\}$ is the set of edge types

- $\mathcal{E}^t = \{e_{ij}^t = (v_i, v_j, t) : v_i, v_j \in \mathcal{V}, t \in \mathcal{T}\}$ is the set of edges of type t (for undirected graphs $e_{ij} = e_{ji}$)

- $\mathcal{E} = \bigcup_{t\in\mathcal{T}}(\mathcal{E}^t)$ is the set of all edges

- $w : \mathcal{E} \to (0, 1]$ is the function that returns the edge weights, i.e. $w(e_{ij}^t)$ is the weight of the edge from v_i to v_j of type t

- $\mathcal{A} = \{a\}$ is the set of all attributes which are numerical or categorical

- $a_\alpha : \mathcal{V} \to \mathcal{D}_\alpha$ is the function that returns the value of attribute α for each vertex. Each vertex is characterized by $\lvert\mathcal{A}\rvert$ attribute values, one for each attribute α.

Furthermore, let $\mathcal{P}$ be set of all vertex properties, i.e. $\mathcal{P} = \{\mathcal{A} \bigcup \mathcal{T}\}$, and $w : \mathcal{P} \rightarrow (0, 1]$ be the function that returns the weight of an edge type t or an attribute a. Weights represent the importance of the edge-types or attributes for the clustering process.

3.2 Clustering Problem

The problem we focus is to identify groups of related ("similar") objects in an attributed multi-graph. Formally, given an attributed multi-graph the goals are to:

- Learn the importance of the various vertex properties, that is, compute a weight $w(p)$ for each vertex property $p \in \mathcal{P}$;
- Compute a $|\mathcal{V}| \times K$ matrix Θ where K is the number of clusters and $\Theta_{i,j}$ is the probability of vertex v_i belonging to cluster $\mathcal{C}_j$ with respect to the constraint $\sum_{j=1}^{K} \Theta_{i,j} = 1$ for all vertices.

so that vertices clustered together have: (a) have high weighted similarity in terms of structure; and (b) low diversity in their attribute values.

In our work, we model the problem of attributed multi-graph clustering as a problem of *defining and optimizing specific objective functions.* The goal is to maximize the unified similarity or equivalently minimize the unified distance between objects in the same cluster. Unified distance/similarity is a function that efficiently combines both the structural and attribute properties of the vertices and captures the desiderata of the final clustering. In its simplest form, the clustering goal is to optimize the following multi-objective function:

$$\arg \min_{K,\Theta\in\mathbb{R}^{|\mathcal{V}|\times K},w} \sum_{i=1}^{|\mathcal{V}|}\sum_{k=1}^{K}\Theta_{i,k}\cdot\left(\sum_{j=1}^{|\mathcal{V}|}\Theta_{j,k}\cdot d(v_i,v_j,w)\right) \tag{1}$$

where $d(v_i, v_j, w)$ is the unified distance of vertices v_i and v_j with respect to the importance w of vertex properties.

Finding the global optimum of Equation (1) is computationally difficult (NP-hard). For example, the detection of maximal cliques or optimization of the commonly used modularity measure on plain graphs is proven to be NP-hard [18, 31]. The same holds for clustering attributed multi-graphs; where the importance of vertex properties is an additional parameter to be optimized. To tackle the high complexity of the problem we: (a) represent a cluster by a virtual vertex which summarizes the properties of the cluster members, i.e. is characterized by $|\mathcal{A}|$ attribute values and connects to vertices by edges of $|\mathcal{T}|$ types[1] ; (b) add additional constraints to the models, i.e. number of clusters is given as input or a vertex is placed in only one cluster; and (c) adopt techniques that are not optimal but converge quickly to local optima and yield high-quality clustering. In this thesis we adopt gradient descent [16] and spectral clustering [75] techniques, which are used by many clustering algorithms like K-means [57] and NCut [75]. Below, we provide an overview of these techniques.

[1]Hence, a cluster $\mathcal{C}_k$ is valid parameter to vertex distance/similarity functions.

3.3 Background

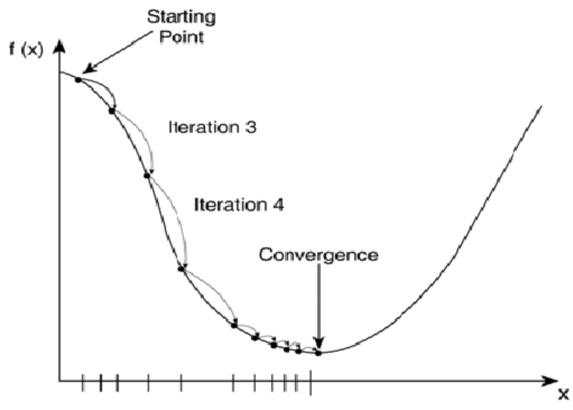
This section describes the gradient descent [16] and spectral clustering [75] techniques that we used to minimize or maximize the defined objective functions.

3.3.1 Gradient Descent

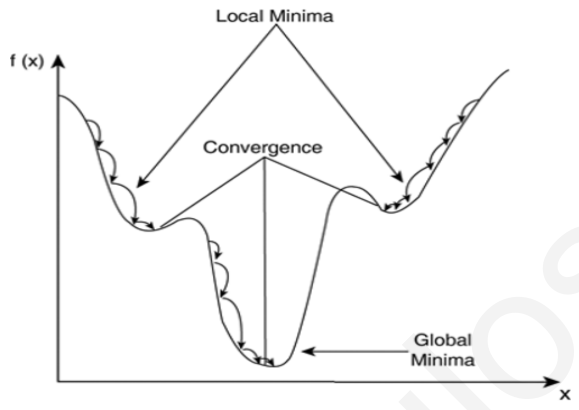
Gradient descent is an optimization technique to find a local minimum of a function [16, 35]. Particularly, at each iteration the algorithm takes one step proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. In case of a multi-variate function, the technique alternates between the dimensions (variables) considering at each step the others as fixed. An illustration of gradient descent is depicted in Figure (2).[2] In Figure (2a) the function to be minimized is assumed to be defined on a two-dimensional space, and to have only one minimum. We see that gradient descent leads us to the minimum of the function, that is, to the point where the value of the function is minimal. In Figure (2b), we see that gradient descent technique is sensible to the starting point of the descent, because the function has three local minima and a different solution is achieved depending on the starting point.

The advantage of the gradient descent technique is that it works in spaces of any number of dimensions. Although in theory it might takes many iterations to converge with a required accuracy, in practice, it converges quickly to local optima [16, 35], and it is used by many clustering algorithms like the K-means.

[2]Source: `http://www.yaldex.com/game-development/1592730043_ch18lev1sec4.html`



(a) Minimization of a function with one minimum



(b) Minimization of a function with three minimums.

Figure 2: Illustration of Gradient Descent

We note that gradient descent is not tightly coupled to clustering, and it is being widely used in many optimization problems. Clustering algorithms that adopt gradient descent differ in how they model the problem and consequently which objective function they aim to optimize.

3.3.2 Spectral Clustering

Spectral clustering takes as input a similarity matrix S. The similarity matrix S, also referred as affinity matrix, is a $|\mathcal{V}| \times |\mathcal{V}|$ matrix where $S_{i,j}$ denotes the similarity of vertices (or data points) v_i and v_j.[3] Spectral clustering makes use of the eigenvalues (spectrum) of the similarity matrix to perform dimensionality reduction before clustering in fewer dimensions. The key idea is to achieve graph partitioning by performing eigendecomposition of a graph Laplacian matrix derived from S. Laplacian matrix is a matrix representation of a graph that can be used to find many useful properties of a graph, such as the number of spanning trees [56].

[3]For a simple graph it can be the adjacency matrix. Simple graph is an undirected graph without multiple edges and loops.

Spectral clustering uses the top K eigenvectors of the Laplacian to define a K-dimensional eigenspace in which vertices (or data points) are easily separable to clusters [27]. Laplacian matrix L for a simple graph is computed using the following equation [25]:

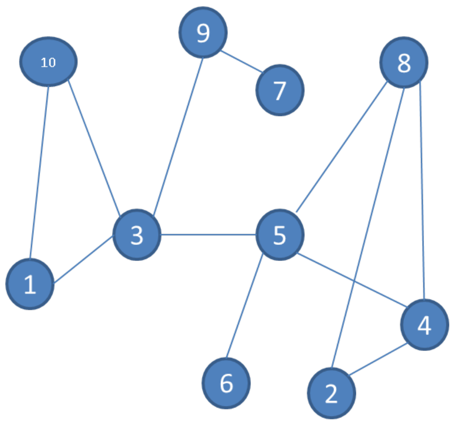
$$L_{i,j} = \begin{cases} deg(v_i) & \text{if } i = j \\ -1 & \text{if } i \neq j \text{ and } (v_i, v_j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $deg(v_i)$ is the degree of vertex v_i. The Laplacian of a simple graph is used to minimize the total cost of the edges among clusters.

Spectral clustering methods differ in how they define and construct the Laplacian matrix L of the similarity matrix S and thus which eigenvectors are selected to construct a K-dimensional eigenspace, aiming to exploit special properties of different matrix formulations [32]. Ulrike von Luxburg's tutorial [56] includes examples of different Laplacians' constructions. For instance, Normalized Laplacian matrix given by Equation (3) is used to minimize the total cost of the edges crossing the cluster boundaries, normalized by the degree of each cluster [75].

$$L_{i,j} = \begin{cases} 1 & \text{if } i = j \\ \frac{-1}{\sqrt{deg(v_i) \cdot deg(v_j)}} & \text{if } i \neq j \text{ and } (v_i, v_j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The basic idea of the spectral clustering technique is as follows. Given a $|\mathcal{V}| \times |\mathcal{V}|$ matrix S where $S_{i,j}$ is the similarity of v_i and v_j:



(a) Simple Graph

Adjacency Matrix	1	2	3	4	5	6	7	8	9	10
1			1							1
2				1				1		
3					1				1	1
4					1				1	
5						1		1		
6										
7									1	
8										
9										
10										

(b) Adjacency Matrix

Laplacian Matrix	1	2	3	4	5	6	7	8	9	10
1	1		-0.354							-0.5
2		1		-0.408				-0.408		
3	-0.354		1		-0.25				-0.354	-0.354
4		-0.408		1	-0.289			-0.333		
5			-0.25	-0.289	1	-0.5		-0.289		
6					-0.5	1				
7							1		-0.707	
8		-0.408		-0.333	-0.289			1		
9			-0.354				-0.707		1	
10	-0.5		-0.354							1

(c) Laplacian Matrix

Figure 3: Example of Laplacian Matrix for a Simple Graph

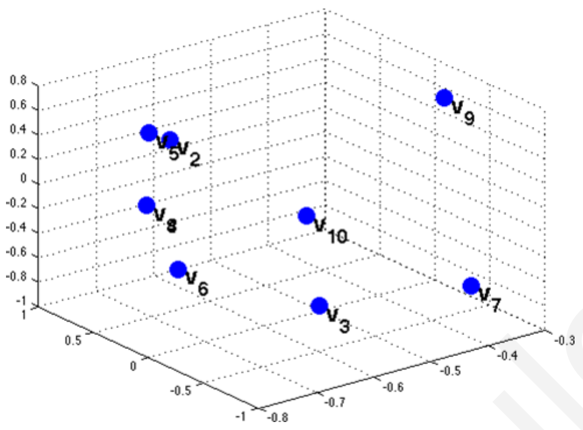
1. Construct the Graph Laplacian L from S. In case of a simple graph, S can be the adjacency matrix (Figure (3)).

2. Perform eigendecomposition on L. That is, solve the eigenvalue problem: $L \cdot u = \lambda \cdot u$ where λ is an eigenvalue and u its corresponding eigenvector.

3. Construct a $|\mathcal{V}| \times K$ matrix U by selecting the top K eigenvectors corresponding to the top K eigenvalues. These eigenvectors define a K-dimensional eigenspace. The columns of U are the top K eigenvectors of the Laplacian matrix L. The rows of U are the embeddings of the $|\mathcal{V}|$ vertices in the eigenspace,

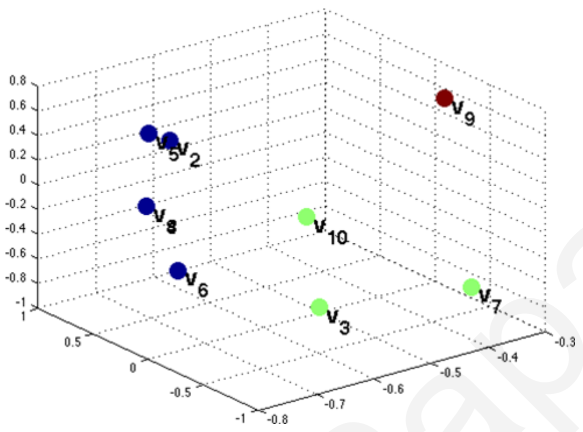
	Top 3 eigenvectors		
	U1	U2	U3
1	-0.659	-0.705	0.263
2	-0.620	0.747	0.241
3	-0.595	-0.486	-0.640
4	-0.668	0.711	-0.221
5	-0.723	0.395	0.566
6	-0.669	0.414	-0.617
7	-0.332	-0.486	-0.808
8	-0.668	0.711	-0.221
9	-0.379	-0.491	0.784
10	-0.659	-0.705	0.263

(a) Eigenvectors

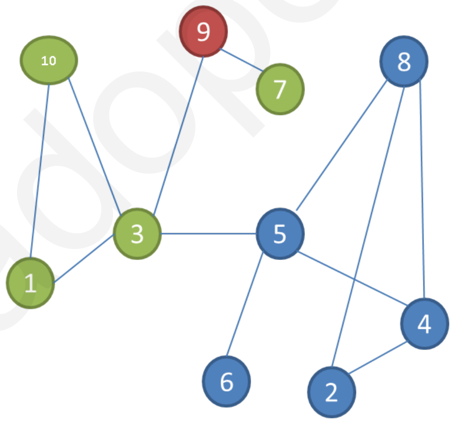


(b) Eigen-space

Figure 4: Example of Eigen decomposition



(a) Clusters in the eigenspace



(b) Project back to the original graph

Figure 5: Example of eigen-space clusters to real graph clusters

which compared to the initial metric space $\mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ of S is a lower dimensional space, since $K \ll |\mathcal{V}|$ in the clustering problem (Figure (4)).

4. Identify clusters in the eigenspace using, for instance, K-means algorithm (Figure (5a)).

5. Project back to the original space (Figure (5b)).

By definition spectral clustering requires the similarity matrix to be symmetric which implies only undirected graphs. Approximations that allow application of spectral clustering on directed graphs have been proposed [60]. The main advantage of spectral clustering is the fact that it identifies clusters of arbitrary shapes and sizes. That is, because similar objects are projected closely in the eigenspace independently of their representation in the initial space.[4]

[4]For instance, in Figure (4) vertices v_1 and v_{10} are projected to the same point in the 3-dimensional eigenspace.

Chapter 4

Related Work

Many methods have been proposed in the context of graphs and multi-graphs clustering [1, 33, 66, 73, 83, 99]. However, these methods ignore vertex attributes and cannot be directly applied to attributed graphs or multi-graphs.

Recently, attributed graph clustering has received much attention [7, 13, 19]. The representative approaches are based on unified distance functions [13, 23, 24, 101] or model definitions [80, 90, 91, 93]. Thus, we categorize attributed graph clustering algorithms into two types:

- Distance-based
- Model-based

4.1 Distance-based Attributed Graph Clustering

Distance-based algorithms adopt or define a unified similarity/distance measure that combines both structural and attribute information of the vertices. Based on this measure various approaches are followed to identify clusters in which vertices are characterized by high similarity or equivalently low distance.

The challenges of distance-based algorithms for clustering attributed graphs are, firstly, the design of such a unified distance measure that takes into account both the structural and the attribute structure and, secondly, the optimization of its parameters, i.e. weights reflecting the importance of each vertex property. To tackle these challenges the majority of distance-based methods fall into one of the categories described in the following paragraphs.

4.1.1 Centroid-based

Centroid-based approaches typically follow K-means or K-medoids approach [47]. That is, they initially choose K cluster prototypes and iterate through the following steps: (i) assign vertices to their closest or most similar cluster according to the unified distance or similarity measure, respectively; and (ii) update cluster prototypes. The clustering process completes when no more changes occur to cluster prototypes or vertex memberships. Since after some iterations results improvement is limited these algorithms usually terminate if at the end of an iteration changes are insignificant, i.e. below a small fraction, usually referred as 'accepted error' or 'convergence delta' denoted as ϵ and δ respectively.

For instance, SA-Cluster [101] is a centroid-based approach that uses random walk distance on an augmented graph in order to calculate the unified distance between two vertices (or a vertex and a cluster). The augmented graph is the original graph without attributes enriched with new vertices each of which represents an attribute value. Then, a weighted edge from a graph vertex to an attribute vertex is added if the vertex is characterized by the value that is represented by the specific vertex. Its weight depends on the importance of the attribute represented by the attribute vertex. By enriching the graph, vertices which share the same attributes are closer and there is a path of two hops, through the attribute vertex, between them. To efficiently compute the random walk distances, authors further proposed two extended versions: an incremental distance computation in Inc-Cluster [102] and an approximate distance computation in SA-Cluster-Opt [23].

4.1.2 Hierarchical

Approaches in this category, namely hierarchical, aim to form a hierarchy of clusters. Specifically, they iteratively either merge or split clusters based on a unified similarity/distance function. The clustering process terminates when a new merge/split is not possible, i.e. a cluster of one object cannot be further split, or a cluster cannot be merged with other cluster [76]. Alternatively, a stopping criteria may be imposed to stop before completing the clusters hierarchy. Usually, these criteria are based on cost functions that evaluate the results at each iteration, i.e., if a new split/merge does not improve the cost function the clustering process terminates.

PICS [8] is a hierarchical clustering method that uses the total encoding cost in bits as stopping criteria. Initially, all vertices are in one cluster and at each iteration it splits the cluster with the maximum per attribute entropy followed by a split based on the adjacency matrix. It halts when a new splitting does not lower the total encoding cost.

4.1.3 Spectral Clustering

Spectral clustering technique projects the attributed graph into a lower dimensional space by performing eigendecomposition on its unified Laplacian matrix. The Laplacian is defined differently by each method, aiming to select different eigenvectors and thus identify clusters with different properties (see Section 3.3.2). In this lower dimensional space, known as eigenspace, the K-means clustering algorithm is adopted to generate the final clusters [38, 51, 54, 75]. Clusters produced by methods in this category have arbitrary shapes and sizes.

4.1.4 Graph Transformation

The attributed graph is projected to a weighted graph in which the weights of the edges represent a combination of attribute and structural similarities. Usually, the edges are sampled based on various criteria to avoid having a complete graph,[1] i.e. edges with weight less than a threshold are removed. Consequently, they apply a clustering algorithm for plain (without attributes) weighted graphs [68, 70, 78].

For instance, CODICIL [70] adds a new edge from each vertex to its most attribute-similar vertices. On the new graph, it computes the similarity of two

[1]A complete graph is a graph where there is an edge between every pair of vertices.

vertices as the overlap of their neighbor sets and prunes the edges such that a vertex connects only to its most similar vertices according to a threshold. Any graph clustering algorithm is then applied to the new graph.

4.2 Model-based Attributed Graph Clustering

In model-based (also known as distribution-based) schemes, probabilistic distribution models are used to cluster the data. Specifically, model-based methods assume that the data are generated by a mixture of underlying probability distributions [89, 90, 93]. Hence, their purpose it to calculate the parameters of these distributions to optimize inter-cluster similarities. A cluster is generally modeled by its connection (e.g. Bernoulli) and attribute distributions (e.g. exponential, Gaussian). Each probability distribution that characterizes a cluster assigns a probability to each vertex belonging to it, according to its probability density function (pdf). Since vertices in the same cluster are most likely belonging to the same distribution, vertices are categorized into the cluster they follow its distributions. The attributed graph is a finite mixture model of these clusters [82, 84].

For example, BAGC [90] assumes that each cluster is generated by a Bernoulli distribution modeling the edges among its members, and by a set of multinomial distributions, one for each attribute. It uses the Bayesian inference to estimate the parameters of the distributions. Bayesian inference is an inference method in which Bayes' rule is used to update the probability estimate for a hypothesis as additional evidence is learned. The approach has been generalized to weighted attributed graphs in [91]. CESNA [93] is another model-based approach that assumes edges and attributes of vertices in the same cluster follow Bernoulli distributions.

The key problem of model-based clustering methods is known as over-fitting. Over-fitting occurs when the model captures random error or noise instead of the actual relationships of the objects. A excessively complex model is prone to over-fitting, and as a result it has poor performance. To avoid it, constraints are added to the model. The most widely used constraint is the predefinition of the number of clusters [89].

4.3 Discussion

Concluding this chapter, we provide a summary and qualitative comparison of some of the current approaches for clustering attributed graphs, and we outline several non-trivial challenges.

An overview of attributed graph clustering methods is depicted in Table 3. There are many aspects to consider upon selecting which method to use. For example: (a) the applicability of a specific method to the network under study, i.e. some methods do not handle weighted attributed graphs; (b) the handling of numerical and categorical attributes; and (c) the characteristics of the final clustering are important.

In addition, the different importance of edge-types and attributes is generally ignored. Weighting mechanisms to reflect the different importance of various properties and improve results quality have been encapsulated in both traditional data

Table 3: Comparison of Attributed Graph Clustering Methods. *1. For undirected unweighted graphs PICS can also identify densely connected components. *2. Clustering Results: H=Hard, O=Overlapping, F=Fuzzy. *3. CODICIL uses any traditional graph clustering algorithm to generate the final clusters.

	Model Based			**Distance Based**		
	BAGC [90]	**CESNA** [93]	**GenClus** [80]	**CODICIL** [70]	**PICS** [8]	**SA-Cluster** [101]
	Graph Properties					
Directed	✓	✓	✓	✓	✓	✓
Weighted			✓	✓		✓
Multi-graph			✓			
	Structural Properties					
Edge-type importance			✓			
Densely Connected Components	✓	✓	✓	✓	✓^{*1}	✓
Similar Connectivity				✓	✓	
	Attribute Properties					
Attribute importance		✓				✓
Numerical			✓			
Categorical	✓	✓	✓	✓	✓	✓
Incomplete			✓			
	Algorithm Design					
Parameter free					✓	
Clustering Results^{*2}	H	O	F	All^{*3}	H	H

clustering [40] and attributed graph clustering methods [23, 87]. GenClus that considers the multiple edge-types and their different importance allows only one edge (of a specific type) between two vertices while it does not deal with different importance of attributes. GenClus is the only approach that, by design, handles incomplete and numerical attributes as well. Numerical attributes though can be discretized. CESNA and SA-Cluster assign different weights to attributes thus incorporating their different importance in the clustering process. CESNA weights attributes separately for each cluster, thus it identifies clusters in attribute subspaces of the network. Weighting schemas have been also adopted to balance the sets of attributes and edges [19]. However, these methods either do not apply to attributed multi-graphs, do not concern simultaneously the different importance of structural-attribute properties and the different importance of the sets of edges-attributes, or they do not automatically compute the weights.

A characteristic of the majority of the above approaches for clustering attributed graphs is the requirement of user specified parameters such as the number of clusters, thresholds or similarity functions. These parameters though are often unknown, hard to be defined and do not adapt well to different application domains, thus requiring fine-tuning for each individual application/dataset. PICS has the advantage of being parameter-free. In contrast to all other methods, PICS identifies clusters that exhibit similar connectivity patterns and attribute homogeneity. It is noted that similar connectivity does not imply that the algorithm is parameter-free. For undirected unweighted graphs, PICS also identifies densely connected components. However, PICS does not consider either the fact that different types of directed edges exist in such networks or the different importance of the attributes.

The concept of similar connectivity has been studied on unattributed unweighted graphs [79, 88]. SCAN [88] exploits the neighborhood of vertices to partition the

Table 4: Time Complexity of Attributed Graph Clustering Algorithms. It is noted that the number of iterations (R) differs for each method. Also, the number of clusters (K) for PICS is not constant.

Algorithm	Time Complexity
BAGC	$\approx O\left(R \cdot (K + \|\mathcal{V}\| + \|\mathcal{E}\| + \sum_{\forall\alpha} \|\mathcal{D}_\alpha\|)\right)$
GenClus	$\approx O(R \cdot (R_1 \cdot K \cdot (\|\mathcal{V}\| \cdot \|\mathcal{P}\| + \|\mathcal{E}\|) + R_2 \cdot t^{2.376}))$
PICS	$\approx O(R \cdot K^2 \cdot (\|\mathcal{V}\| \cdot \|\mathcal{P}\| + \|\mathcal{E}\|))$
SA-Cluster	$\approx O(R \cdot \|\mathcal{V}\|^3)$
SA-Cluster-Opt	$\approx O(R \cdot (3 \cdot \|\mathcal{V}\| + \log_2 l))$ where l is the length limit of the random walks
CESNA	$\approx O(R \cdot \left(\|\mathcal{E}\| + \|\mathcal{V}\| \cdot \sum_{\forall\alpha} \|\mathcal{D}_\alpha\|)\right)$
CODICIL	$\approx O(\|\mathcal{V}\|^2 \cdot (\|\mathcal{P}\| + \log \|\mathcal{V}\|) + O(\|\mathcal{V}\| \log \|\mathcal{V}\|)$

network such that vertices sharing many neighbors are grouped into the same cluster. Although SCAN optimizes similar connectivity, it is sensible to a threshold parameter: the minimum number of common neighbors. To overcome the issue of selecting the threshold parameter of SCAN, gSkeletonClu [79], which additionally aims to find hubs and outliers, has been proposed. Also, the concept of similar connectivity is close to the concept of block models. Block models is a generative model for generating undirected unweighted graphs. The idea is to divide the vertices into hard clusters, or "blocks", where all vertices in the same cluster have the same pattern of connections to vertices in other clusters. Stochastic block model (also referred to as planted partition model [26]) weakens this idea: the probability of an edge between two vertices just depends on which clusters they belong to, and is independent across edges [1, 4, 46]. Although these approaches and models are useful, they do not apply to attributed multi-graphs.

Furthermore, the runtime of the method is a crucial aspect. Table 4 presents the worst-case time complexity of the above methods. It is noted that the number of iterations R and the number of clusters K is not the same for all the algorithms. We have experimentally noticed that CESNA and BAGC, although are model based approaches, are the fastest currently available methods for clustering attributed graphs, followed by SA-Cluster-Opt and PICS. We note that for SA-Cluster-Opt we do not include the time for augmenting the graph because the graph augmentation occurs only once before the algorithm is applied. GenClus requires more time because it is based on EM algorithm, deals with multiple edge types and identify their importance.

Moreover, the use of augmented graphs (SA-Cluster) can lead to an explosion of graph size, increasing significantly the requirements in memory and time. CESNA does not augment the original graph, but handles non-weighted attributed graphs. CODICIL can be applied to weighted graphs but the graph transformation process is relatively slow. CODICIL has the advantage that once the original graph is transformed a lot of algorithms can be applied on it.

We conclude this section by noting the existence of many open areas in the field of attributed graph clustering. Particularly, we observe among others that: (a) the existence of more than one edge type is largely ignored; (b) the identification of clusters characterized by similar connectivity has been airily considered; (c) although the different importance of attributes and edge-types have been studied by various approaches, none of these considers both simultaneously.

Following chapters present some work done towards supporting this thesis. Particularly, three methods for clustering attributed multi-graphs, namely HASCOP, CAMIR, and CLAMP, have been proposed. All of them consider the existence of multiple edge-types as well as the different importance of attributes and edge-types simultaneously. Although these three methods attack the same problem, each of them optimizes a different objective function aiming to optimize different cluster properties by exploiting various characteristics of the network under study.

Chapter 5

Homogeneous Attributes and Similar COnnectivity Patterns - HASCOP

HASCOP (Homogeneous Attributes and Similar COnnectivity Patterns) [63] is a heuristic parameter-free distance-based attributed multi-graph clustering method that clusters the network such that *both attribute and structural similarities are maximized.* It also automatically identifies the importance of each attribute and edge-type during the clustering process.

HASCOP defines a heuristic function that combines both structural and attribute properties of the vertices according to their identified importance, by assigning different weights to each of them. At each iteration, these weights are updated according to the new clustering results using a scoring mechanism similar to SACluster. An attribute is considered more important if in the present clustering configuration vertices in the same clusters share the same value for the specific attribute. An edge type is considered more important if vertices in the same cluster are inter-connected by this edge type. Initially, every vertex is considered to belong into one cluster. Iteratively clusters are merged, and edge-type and attribute weights are adjusted until convergence. Clustering process stops if at the end of an iteration the number of clusters is not further reduced. Resulted clustering configuration exhibits attribute homogeneity and similar connectivity.

Formally, with respect to the clustering problem presented in Section 3.2, given an attributed multi-graph G, HASCOP goals are:

- Identify the importance of each edge type and calculate its weight $w(t)$, where $w(t) \in (0,1]$ and $\sum\limits_{\forall t \in \mathcal{T}} w(t) = 1$.
- Based on the importance of each attribute, assign each of them a weight $w(\alpha)$, such that $w(\alpha) \in (0,1]$ and $\sum\limits_{\forall \alpha \in \mathcal{A}} w(\alpha) = 1$.
- Find a fuzzy clustering $\Theta_{|\mathcal{V}| \times K}$, where $\Theta_{i,j}$ is the probability of v_i belonging to cluster C_j subject to $\sum\limits_{j=1}^{K} \Theta_{i,j} = 1$ for all vertices.

In other words, the goal is to identify the importance of each edge-type and attribute, and cluster an attributed multi-graph such that vertices in the same cluster exhibit **both** *a similar connectivity pattern* and *attribute coherence*. In the following paragraphs we give an overview of the similar connectivity and attribute coherence, followed by HASCOP objective function and optimization process.

5.1 Distance Measures

5.1.1 Similar Connectivity

A set of vertices connecting to the same vertices are characterized by similar connectivity. For the sake of simplicity we limit the discussion in the following paragraphs to only one type of edges.

Two vertices v_i, v_j exhibit a similar connectivity pattern if $\mathcal{N}[v_i]$ and $\mathcal{N}[v_j]$ highly overlap, where $\mathcal{N}[v_i]$ is the closed out-neighbourhood of vertex v_i. Formally, the closed out-neighbourhood of a vertex is defined as [20]:

$$\mathcal{N}[v_i] = \{v_i\} \cup \{u : (v_i, u) \in \mathcal{E}\} \tag{4}$$

If $\mathcal{N}[v_i]$ and $\mathcal{N}[v_j]$ highly overlap then v_i and v_j link to common vertices. It is noted that *vertices having similar connectivity may not be directly connected to each other.* Thus, the density metric of the final clustering may not be close to one because clusters are not necessary densely connected components. The intuition is that vertices related to common vertices should form a cluster even though they are not inter-connected. For instance, in the case of a social network, if some people have a lot of common friends (and common attributes) but they do not know each other, they should form a group, and such groups are covered as well.

Furthermore, it is desired that the similar connectivity pattern of two vertices, which link to common vertices, to be higher if they are also connected to each other. For example, a group of people having common friends should belong to the same clusters with higher probability if they are also friends with each other. To address this, we choose the *closed* neighborhood of a vertex. That is, the set containing the vertices that v_i connects to *and* v_i itself, as is given by Equation (4).

5.1.2 Attribute Coherence

Two vertices have attribute coherence if their distance is low, that is their attribute vectors are close in $\mathbb{R}^{|\mathcal{A}|}$. The intuition is that vertices described by close attribute values should have high attribute coherence.

5.2 HASCOP Clustering Model

5.2.1 Overview

HASCOP fuzzy clustering model for attributed multi-graphs makes use of a similarity function denoted as *sim* that combines both the structural and attribute properties of the vertices according to their importance. Hence, similarly to fuzzy clustering algorithms, for each vertex we want to maximize the weighted (by its memberships) sum of its similarities with the clusters it belongs. Equally, the goal is to maximize the following objective function:

$$O(\Theta) = \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{K} \Theta_{i,j} \cdot sim(v_i, \mathcal{C}_j) \tag{5}$$

where $\Theta_{i,j}$ is the probability that vertex v_i belongs to cluster $\mathcal{C}_j$, and $sim(v_i, \mathcal{C}_j)$ is a heuristic function that returns the similarity of vertex v_i and cluster $\mathcal{C}_j$ based on the different importance of edge-types and attributes.

A cluster is considered to be a regular vertex in the attributed multi-graph (characterized by $|\mathcal{A}|$ attributes and connected to vertices by $|\mathcal{T}|$ types of edges) that represents its members. The weights of a cluster's outgoing edges are given as the weighted average of the outgoing edges of its members. The attribute centroid of a cluster is given by the attributes that characterize most of its members, i.e. the mode of the cluster [41]. We defer discussion in Sections 5.2.3 and 5.2.4.

HASCOP maximizes the objective function given by Equation (5) as follows. Initially each vertex is categorized in a cluster by itself. Iteratively clusters are being updated (some are eliminated) and edge-type and attribute weights are adjusted. An iteration starts with categorizing the vertices into the updated clusters according to their similarity and the updated weights. An edge type or an attribute is considered more important and is given higher weight at the next iteration if vertices in current clusters inter-connect by edges of this edge type or share the same value for the specific attribute, respectively.

5.2.2 Similarity Function and Membership Calculation

We define the unified similarity function *sim* as follows:

$$sim(v_i, \mathcal{C}_j) = link\_sim(v_i, \mathcal{C}_j) \cdot attr\_sim(v_i, \mathcal{C}_j) \tag{6}$$

where $link\_sim(v_i, \mathcal{C}_j)$ and $attr\_sim(v_i, \mathcal{C}_j)$ are the structural and attribute similarities of vertex v_i and cluster $\mathcal{C}_j$, respectively.

The reason for selecting to multiply the two similarities is the following; a vertex has high similarity with a cluster if **both** their similar connectivity and attribute coherence are high. For example, if *link_sim* is high (i.e. 0.9) and *attr_sim* is low (i.e. 0.1) the total *sim* will be low (i.e. 0.09). Similarly, if *link_sim* is low and *attr_sim* is high the total *sim* will still be low.

We calculate the membership of a vertex to clusters by normalizing its similarities using the following equation:

$$\Theta_{i,j} = \frac{sim(v_i, \mathcal{C}_j)}{\sum_{l=1}^{K} sim(v_i, \mathcal{C}_l)} \tag{7}$$

Equation (7) assigns a vertex v_i to a cluster $\mathcal{C}_j$ with high probability $\Theta_{i,j}$ if their similarity is high. Thus, the higher the similarity of a vertex v_i and a cluster $\mathcal{C}_j$ is, the higher its membership will be. Hence, the objective function (Equation (5)) is maximized.

The following paragraphs focus on the problem of calculating similar connectivity and attribute coherence as well as on how to adjust the attribute and edge-type weights in order to calculate the final clustering.

5.2.3 Structural Properties

As stated earlier, vertices and clusters can be connected via edges of $|\mathcal{T}|$ types. The initial unweighted multi-graph is projected to a weighted "projected graph" following a process that aggregates various edge types. The weight of the edge (v_i, v_j) is calculated based on the importance of each edge type using the following function:

$$\omega(v_i, v_j) = \sum_{t=1}^{|\mathcal{T}|} w(t) \cdot \omega_t(v_i, v_j) \tag{8}$$

The weight of the edge from a cluster $\mathcal{C}_j$ to a vertex u is given by the following equation:

$$\omega(\mathcal{C}_j, u) = \frac{\sum\limits_{\forall v_i \in \mathcal{V}} (\Theta_{i,j} \cdot \omega(v_i, u))}{\sum\limits_{\forall v_i \in \mathcal{V}} \Theta_{i,j}} \tag{9}$$

Provided the projected graph we define a similarity function $link\_sim(v_i, \mathcal{C}_j)$ which represents the similar connectivity pattern of v_i and $\mathcal{C}_j$. The similarity connectivity pattern of vertex v_i and cluster $\mathcal{C}_j$ takes a value in the range $(0, 1]$, and is given by the following equation:

$$link\_sim(v_i, \mathcal{C}_j) = \frac{1}{1 + \sqrt{\sum\limits_{\forall u \in \mathcal{V}} (\omega(v_i, u) - \omega(\mathcal{C}_j, u))^2}} \tag{10}$$

If v_i connects to the same vertices with same link types as $\mathcal{C}_j$, then their similar connectivity is one because their distance is zero.[1] On the opposite side, if there is not a vertex u that both v_i and $\mathcal{C}_j$ connect to then their distance will be high and their similar connectivity will be very close to zero. The value one is added to the distance to ensure that the result is in the range $(0, 1]$ as the distance can be less than one. Equation (10) is preferred over any other traditional measure as it takes into account the outgoing edges of a vertex and not its incoming edges.

5.2.4 Attribute Properties

We calculate the attribute similarity of a vertex and a cluster, based on attribute importance, by transforming to similarity the distance of their attribute vectors,

[1]This is a sufficient but not necessary condition. The link distance of a cluster and a vertex can actually be zero if the sum of the importance of their outgoing edges is the same. Since the weight (importance) of an edge-type is updated at every iteration, the probability of this scenario is very low and is decreasing proportionally to the number of iterations.

using the following equation:

$$attr\_sim(v_i, \mathcal{C}_j) = \frac{1}{1 + \sqrt{\sum\limits_{\forall \alpha \in \mathcal{A}} w(\alpha) \cdot \delta\left(a_\alpha(v_i), a_\alpha(\mathcal{C}_j)\right)}} \tag{11}$$

where

$$\delta(x, y) = \begin{cases} 0 & \text{if} \quad x = y \\ 1 & \text{else} \end{cases}$$

The attribute value of a cluster $\mathcal{C}_k$ for an attribute α, denoted as $a_\alpha(\mathcal{C}_k)$, is the value that characterizes most of its members:

$$a_\alpha(\mathcal{C}_k) = \arg\max_{y \in \mathcal{D}_\alpha} \sum_{i=1}^{|\mathcal{V}|} 1 | a_\alpha(v_i) = y \wedge \Theta_{i,k} > 0 \tag{12}$$

We add one to the denominator of Equation (11) for two reasons. The first reason is to ensure that *attr_sim* is always in the range $(0, 1]$, and the second reason is to ensure that the result is close to one if the attribute values of vertex v_i are very similar to the attribute values of cluster $\mathcal{C}_j$.

5.2.5 Adjustment of Edge-Type and Attribute Weights

Given the clustering results at each iteration we must identify the importance of the various edge-types and attributes. Equally, we must adjust the weights, $w(\alpha)$ and $w(t)$, for each attribute α and edge-type t. In order to automatically adjust the weights we use a scoring mechanism, based on which before proceeding to the next iteration we adjust accordingly the edge-type and attribute weights. We prefer the following mechanism over the available approaches, such as ReliefF [69], since we update the importance of edge-types and attributes at each iteration to incorporate the knowledge from the clustering results. At each iteration a score is assigned to each edge-type and each attribute based on the current clustering and their importance. Specifically, an edge-type or an attribute is considered more important if vertices in the same cluster are inter-connected by edges of this type or share the same value for the specific attribute, respectively. Based on its score, a temporary weight is calculated and used to compute the weights for the next iteration.

Let $w^r(p)$ be the weight of property (edge-type or attribute) p at the beginning of iteration r. Also, let $w^r(p)'$ be the temporary weight at iteration r, which represents the importance of edge-type or attribute p by considering only the scores and the clusters at iteration r. The updated weight is then given by:

$$w^{r+1}(p) = \frac{1}{2} \cdot (w^r(p) + w^r(p)') \tag{13}$$

where

$$w^r(p)' = \frac{edge\_type\_score(\mathcal{E}_p, \Theta)}{\sum\limits_{\forall t \in \mathcal{T}} edge\_type\_score(\mathcal{E}_t, \Theta)} \text{ if } p \in \mathcal{T} \tag{14}$$

$$w^r(p)' = \frac{attr\_score\,(p, \Theta)}{\sum\limits_{\alpha \in \mathcal{A}} attr\_score\,(\alpha, \Theta)} \text{ if } p \in \mathcal{A} \tag{15}$$

We define the function *edge_type_score*, which returns the score of edge-type t based on the current clustering, as follows:

$$\begin{aligned} edge\_type\_score\,(\mathcal{E}_t, \Theta) \quad &= \sum_{(u,v) \in \mathcal{E}_t} \sum_{j=1}^{K} (\Theta_{u,j} + \Theta_{v,j}) \\ \text{if} \qquad &\Theta_{u,j} > 0, \Theta_{v,j} > 0 \end{aligned} \tag{16}$$

In other words, to calculate the score of an edge-type t, we add the probabilities of each pair of vertices u and v that belong to a cluster $\mathcal{C}$, if there is an edge of type t between u and v. It is noted that *edge_type_score* returns a value that is not a probability but just a measure of the importance of edges of type t in the current clustering. The higher the score of edge-type t is, the more important the edge-type t is considered.

Similarly to edge-types, the higher the score of an attribute the more important it is. For an attribute its score is given by the sum of the number of vertices in the same cluster that are sharing the same value for that attribute:

$$\begin{aligned} attr\_score\,(\alpha, \Theta) &= \sum_{(u,v)\in\mathcal{V}\times\mathcal{V}} \sum_{j=1}^{K} (1) \\ &\quad \text{if } a_\alpha(u) = a_\alpha(v) \text{ and } \Theta_{u,j} > 0, \Theta_{v,j} > 0 \end{aligned} \tag{17}$$

Lastly, it is noted that because $\sum w^r(p) = 1$ and $\sum w^r(p)' = 1$ the constraint $\sum w^{r+1}(p) = 1$ is not violated.

Based on the above weighting mechanism, if a property p is important it will get higher weight at the next iteration because $w^r(p)'$ will be higher than $w^r(p)$. On the other hand, if a property is not important then $w^r(p)'$ will be less than $w^r(p)$ and thus its importance will be lowered for the next iteration.

5.3 HASCOP Algorithm

The following paragraphs describe the initialization and the clustering process of HASCOP as well as the way clusters are handled and updated at each iteration.

5.3.1 Initialization

Initially, edge-types (and attributes) are considered to share the same importance among them. Thus, initial weights are given by the following equations:

$$w(t) = \frac{1}{|\mathcal{T}|} \quad \text{and} \quad w(\alpha) = \frac{1}{|\mathcal{A}|} \tag{18}$$

For clusters initialization we set each vertex to be a singleton cluster. A singleton cluster is a cluster which has only one member.

5.3.2 Clustering Process

HASCOP takes as parameters only the information network which is modelled as an attributed multi-graph. At each iteration, the attributed multi-graph is projected to an attributed graph (Equation (8)) and the clustering memberships are computed (Equation (7)). We prune the membership values below a threshold ϵ (we set $\epsilon = 0.001$). This pruning, firstly, guarantees that a vertex does not belong to all the clusters, and secondly it enables the use sparse matrices for efficient implementation. After pruning, identical clusters (clusters that represent exactly the same vertices) may exist. Thus, we eliminate these clusters as follows. Each group of identical clusters ($\mathcal{C} = \{\mathcal{C}_x, \ldots, \mathcal{C}_y\}$) is replaced by a new cluster ($\mathcal{C}_{new}$). The probability of a vertex u belonging to cluster $\mathcal{C}_{new}$ is given by $\max(\Theta_{u,x}, \ldots, \Theta_{u,y})$. As a result, each vertex will belong to the new cluster with the highest calculated membership probability it belongs to the identical clusters. We select the maximum membership probability for each vertex u since we assume that this one captures better the similarity of vertex u with the other vertices that belong to the identical clusters (in

Algorithm 1 HASCOP

```
Input: Attributed Multi-graph
Output: Fuzzy clustering of the network denoted by a $|\mathcal{V}| \times K$ matrix $\Theta$
 1: Initialize weights by Equation (18)
 2: while true do
 3:    Project multi-graph to graph - Equation (8)
 4:    for all vertices $v_i \in \mathcal{V}$ do in parallel
 5:       for all cluster nodes $\mathcal{C}_j$ do in parallel
 6:          Calculate $\Theta_{i,j}$ using Equations (7), (10), (11)
 7:       end for
 8:    end for
 9:    if NOT ProcessDuplicateClusters($\Theta$) then
10:       return $\Theta$;
11:    end if
12:    Parallel update cluster outgoing edges
13:    Parallel update cluster attribute centroids
14:    Parallel update all weights - Equations (14), (15)
15: end while
```

fact, maximum of probabilities represents the union of sets). At each iteration, the number of identical clusters is not known apriori since it depends on the dataset and the current clustering results. If identical clusters do not exist at an early iteration, the algorithm terminates resulting in a clustering that consists of many clusters. At the end of each iteration, the clusters and the weights are updated according to the new clustering Θ.

The pseudo-code of HASCOP clustering process is presented by Algorithm (1). HASCOP terminates when at the end of an iteration the number of clusters does not change. It returns the number of clusters (K) and the fuzzy clustering of the network $\Theta_{|\mathcal{V}| \times K}$. A vertex belongs to the clusters for which is membership is not zero, i.e. $v_i \in \mathcal{C}_j$ if $\Theta_{i,j} \neq 0$. Lastly, we note that memberships for each vertex as well as the cluster centers and the weights can be computed independently. Hence, *HASCOP can be implemented and executed in parallel.*

The time complexity of HASCOP can be expressed as the sum of the costs for calculating Θ, processing duplicate clusters, updating centroids and updating vertex property weights.

The time complexity for calculating $\Theta_{i,j}$ is the cost for calculating the structural and attribute distances between vertex v_i and cluster $\mathcal{C}_j$. The cost for calculating the weighted Euclidean distance of two vectors $v_{d\times 1}$ is $O(d)$. Therefore, the total cost for calculating a similarity between a vertex and a cluster is $O(|\mathcal{V}|+|\mathcal{A}|)$. Since the columns of Θ are K_i, where K_i is the number of clusters at the end of the first step at iteration i, the cost for calculating Θ is $O(|\mathcal{V}| \cdot K_i \cdot (|\mathcal{V}| + |\mathcal{A}|))$. The time complexity for deleting the same clusters is $O(K_i \cdot |\mathcal{V}|)$ and the cost for updating clusters based on Θ is $O(K_i \cdot |\mathcal{V}| \cdot |\mathcal{A}|)$. The time complexities for updating edge type and attribute weights are $O(K_i \cdot \sum_{\forall t} |\mathcal{E}_t|)$ and $O(|\mathcal{A}| \cdot |\mathcal{V}|^2 \cdot K_i)$ respectively. Thus, the total cost for updating all weights is $\approx O(K_i \cdot (|\mathcal{E}| + |\mathcal{V}|^2 \cdot |\mathcal{A}|))$. Putting them all together, the worst case time complexity of Algorithm (1) for R iterations is $\approx O(R \cdot K \cdot (|\mathcal{E}| + |\mathcal{V}|^2 \cdot |\mathcal{A}|))$. During our experiments we observed that HASCOP converges in less than 15 iterations.

Chapter 6

Clustering Attributed Multi-graphs with Information Ranking - CAMIR

CAMIR (Clustering Attributed Multi-graphs with Information Ranking) [65] is distance-based clustering algorithm for attributed multi-graphs that adopts spectral clustering. Although HASCOP achieves high quality results, it is quite slow because at each iteration it computes the weights (Section 5.2.5) and projects the attributed multi-graph to an attributed graph (Section 5.2.3). Also, as an agglomerative-like algorithm during the first few iterations the number of clusters is much higher than the final number of clusters (it is in the order of the number of vertices), which increases significantly the iteration run time. CAMIR computes the weights and projects the attributed multi-graph only once, and thus it is much faster than HASCOP while it achieves clustering results of comparable quality.

Initially, CAMIR uses *a mechanism to rank and consequently weigh the vertex properties in attributed multi-graphs*, by iteratively co-regularizing the clustering hypotheses across the vertex properties. Co-regularization is a well-known technique [51], which we use to compute the 'agreement' among the vertex attributes and the edge-types in attributed multi-graphs. Two vertex properties 'agree' if they assign vertices the same cluster labels when they are used individually. The vertex property with the highest agreement best separates the vertices into clusters, while the property with the lowest agreement introduces noise and reduces the clustering accuracy. CAMIR ranks all vertex properties accordingly; that is, its assigns the highest and the lowest ranks to the properties with the highest and the lowest agreements respectively. According to the ranking, it assigns a weight parameter to each vertex property to compute a unified similarity measure for attributed multi-graphs. Based on the computed measure, spectral clustering technique is adopted to partition the attributed multi-graph and to generate the final clusters.

Formally, the clustering problem presented in Section 3.2 is refined as follows. Given an attributed multi-graph G and a number of clusters K ($K \geq 2$), CAMIR goals are:

- Compute a weight $w(p)$ for each vertex property $p \in \mathcal{P}$, in order to construct a unified similarity matrix $S \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$
- Generate the K clusters based on S, by maximizing the similarity of vertices within a cluster and minimizing the similarity between vertices in different clusters.

6.1 CAMIR Clustering Model

The proposed CAMIR method consists of the following steps:

- The attributed multi-graph is processed to rank the vertex properties and to calculate their weights accordingly; then a unified similarity measure is computed by considering all vertex properties and their importance based on the calculated weights.
- A spectral clustering approach is adopted to embed the vertices into the respective eigenspace of the unified similarity measure.

The reason for selecting spectral clustering is that it identifies clusters of arbitrary shapes and sizes (see Section 3.3.2). The key idea in spectral clustering is to achieve graph partitioning by finding the best cut. CAMIR follows the Normalized Cut (NCut) [75] method.[1] NCut tries to minimize the total cost of the edges crossing the cluster boundaries, normalized by the total degree of each cluster, making thus the clusters having similar degrees. In the following Sections we present each step of the CAMIR method in detail.

6.1.1 Information Ranking

Given an attributed multi-graph G, we compute the affinity matrices $S^p \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$, where $S^p_{ij} \geq 0$ denotes the relationship - similarity between v_i and v_j for property $p \in \mathcal{P} \equiv \{\mathcal{A} \cup \mathcal{T}\}$. For each edge-type $t \in \mathcal{T}$, the respective similarity

[1]Alternatively, several parallel spectral clustering methods could be used in the proposed approach, such as the works of [22], [45], to reduce the computational time of spectral clustering.

matrix is calculated as follows:

$$S^t_{ij} = \omega_t(v_i, v_j) \tag{19}$$

For each vertex attribute $a \in \mathcal{A}$ we calculate the respective similarity matrix based on the Gaussian kernel:[2]

$$S^{\alpha}_{ij} = \exp\left(-\frac{||a_\alpha(v_i) - a_\alpha(v_j)||^2}{2 \cdot \sigma_i \cdot \sigma_j}\right) \tag{20}$$

where σ_i is a scaling parameter to control how rapidly the similarity S^{α}_{ij} reduces according to the distance between v_i and v_j. For each vertex v_i, the scaling parameter σ_i allows the self-tuning of the vertex-to-vertex distances according to the local statistics of the neighbourhoods surrounding v_i. We followed the self-tuning strategy of [22]. Provided that v_i has ϵ neighbours, σ_i is calculated as the average of the ϵ distances.

According to Equations (19) and (20), for each vertex property $p \in \mathcal{P}$ we construct an affinity matrix, denoted as $S^p \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$. We compute the normalized Laplacian $L^p \in \mathbb{R}^{|\mathcal{V}|\times|\mathcal{V}|}$ of S^p as follows:

$$L^p = I - D^{p^{-1/2}} \cdot S^p \cdot D^{p^{-1/2}} \tag{21}$$

where I is the identity matrix and D^p is a diagonal matrix calculated as follows:

$$D^p_{ii} = \sum_{j=1}^{|\mathcal{V}|} S^p_{ij} \tag{22}$$

where $D^{p^{-1/2}}$ indicates the inverse square root of D^p. For any S with $S_{ij} \geq 0$, the Laplacian matrix is symmetric positive semi-definite [56]. After computing the

[2]Also, other types of kernel functions could be used, such as linear and polynomial, thoroughly examined in [39] for machine learning methods.

$|\mathcal{P}|$ different normalized Laplacians, we perform eigendecomposition on each of the normalized Laplacians to retrieve their top K eigenvectors, denoted by $U^p \in \mathbb{R}^{|\mathcal{V}| \times K}$ for the p-th Laplacian L^p.

According to [51] the vertex property p that best separates the vertices is selected using the following equation:

$$f(\mathcal{P}) = \underset{U^p \in \mathbb{R}^{|\mathcal{V}| \times K}}{\arg\max}\, tr \left[U^{p^T} \cdot \left(L^p + \lambda \cdot \sum_{\substack{i=1 \\ p_i \neq p}}^{|\mathcal{P}|} \left(U^{i^T} \cdot U^i \right) \right) \cdot U^p \right] \tag{23}$$

where $\mathcal{P}$ is the set of all vertex properties, tr is the trace of the matrix, U^{i^T} is the transpose matrix of U^i, and λ is a co-regularization parameter that controls the penalization of a property according to its 'disagreement' with the other properties,[3] denoted by the sum in Equation (23). According to [51] two different vertex properties 'agree' if they produced the same clustering result when used individually. Equation (23) returns the property p that has the highest 'agreement' with the rest properties, assuming that if we use only the selected property p for clustering we expect to find accurate clusters, independently of the rest of the properties. The mathematical proof can be found in [51].

However, considering just a single property to perform clustering contradicts with works elaborating on the use of all vertex properties to improve clustering accuracy [24], [68]. We propose to iteratively apply Equation (23) $|\mathcal{P}|$ times. At each iteration we exclude the properties that have been already ranked. In particular, given the set of unranked properties, denoted as $\mathcal{P}_u$, we compute the ranking $r(p)$ of a property

[3]Following [51] we use a common λ for all properties. In practice though, $\lambda = 0.001$ is an appropriate value to control the impact of the other properties, as we observed in our experiments.

p by the following equation:

$$r(p) = \begin{cases} |\mathcal{P}_u| & : \ |\mathcal{P}_u| > 1 \wedge f(\mathcal{P}_u) = U^p \\ 1 & : \ |\mathcal{P}_u| = 1 \end{cases} \tag{24}$$

In doing so, the highest rank is given to the property with the highest 'agreement' with the other properties; while the lowest rank is given to the property that is selected last, does not 'agree' with the rest of properties, and consequently introduces noise to the clustering. Equation (24) maps the vertex properties to the ranks $\{|\mathcal{P}|, |\mathcal{P}|-1, \ldots, 1\}$ according to the order the properties are selected. To calculate the weight of each property, we perform a normalization of the properties' ranking as follows:

$$w(p_i) = \frac{r(p_i)}{\sum\limits_{j=1}^{|\mathcal{P}|} r(p_j)} \tag{25}$$

Equation (25) assigns higher weights to more important vertex properties and down-weighs the properties that have lower ranks and introduce noise over the clustering. Finally, according to property weights we compute the unified similarity matrix $S \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ as the weighted sum of the $|\mathcal{P}|$ similarity matrices S^p:

$$S = \sum_{\forall p \in \mathcal{P}} w(p) \cdot S^p \tag{26}$$

6.2 CAMIR Algorithm

In our method, spectral clustering is formulated as follows: given the $|\mathcal{V}|$ vertices of the attributed multi-graph G and the unified similarity matrix $S \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$, calculated based on Equation (26), the goal is to find K disjoint vertex subsets,

Algorithm 2 CAMIR Algorithm

```
Input: Attributed multi-graph $G$, number of clusters $K$
Output: Partitioning of the network denoted by a $K \times 1$ vector
   $\mathcal{P}_u = \mathcal{P}$
   while $|P_u| \neq \emptyset$ do
      Rank a property $p$ using Equation (24)
      $\mathcal{P}_u = \mathcal{P}_u \setminus \{p\}$
   end while
   Compute weights $w(p_i)$ based on Equation (25)
   Compute the unified similarity matrix $S$ using Equation (26)
   Compute the normalized Laplacian $L$ of $S$ according to Equation
   (21)
   Perform eigendecomposition on $L$ to obtain $U$ - the top $K$
   eigenvectors
   Run $K$-means on $U$ to generate the final cluster labels
```

namely clusters, whose union is the whole data set, by solving the following standard eigendecomposition problem:

$$\arg\max_{U \in \mathbb{R}^{|\mathcal{V}| \times K}} tr\left(U^T \cdot L \cdot U\right), \text{ s.t. } U^T \cdot U = I \tag{27}$$

where L is the normalized Laplacian of the unified similarity matrix S given by Equation (26). The columns of $U \in \mathbb{R}^{|\mathcal{V}| \times K}$ are the top K eigenvectors of the Laplacian matrix L, while its rows are the embeddings of the $|\mathcal{V}|$ vertices in the K-th dimensional eigenspace. The final K clusters are generated by applying K-means algorithm to the $|\mathcal{V}|$ embeddings of the vertices.

Algorithm (2) presents the pseudocode of the CAMIR algorithm. We note that parallel eigendecomposition solvers have been widely studied in the literature [71, 86, 96], successfully solving relatively quick the problem on billion-sized matrices. Hence, CAMIR can take advantage of such solvers to scale to large attributed multi-graphs. The first step of the proposed approach is depicted by lines 2-6. Lines 2-5 do the properties ranking, and line 6 computes the property weights. The complexity for the first step is $\approx O(|\mathcal{P}| \cdot |\mathcal{V}|^2 \cdot K)$. Lines 8 and 9 correspond to the second step of CAMIR algorithm. That is to compute the normalized Laplacian and the embedding of the attributed multi-graph to the eigenvectors space. This step has complexity $\approx O(|\mathcal{V}|^2 \cdot K)$. Lastly, line 10 applies K-means algorithm to find the cluster labels, which complexity is $\approx O(|\mathcal{V}|^{K^2+1} \cdot \log |\mathcal{V}|)$ [10]. Putting all together, the worst case time complexity is $\approx O(|\mathcal{P}| \cdot K \cdot |\mathcal{V}|^2 + |\mathcal{V}|^{K^2+1} \cdot \log |\mathcal{V}|)$.

Chapter 7

Weighted CLustering of Attributed Multi-graPhs - CLAMP

CLAMP (CLustering Attributed Multi-graPhs) [64] is a distance-based clustering method which maximizes attribute and structural similarities according to automatically identified importance of each attribute and edge type. CLAMP balances globally the sets of attributes and edge-types, applies to directed weighted attributed multi-graphs and handles the heterogeneous vertex attributes (both numerical and categorical), thus overcoming limitations of the methods presented in Chapters 5 and 6.[1] *CLAMP is the first method to perform fuzzy clustering on weighted directed attributed multi-graphs with heterogeneous attributes.*

During the execution of CLAMP, the intermediate results and the importance of the various edge-types and attributes enhance each other until convergence. The importance of edge-type and attribute sets are reflected by the global weights. Similarly, edge-type and attribute weights reflect the importance of each edge-type and

[1]HASCOP handles directed unweighted attributed multi-graphs with categorical attributes. CAMIR handles undirected weighted attributed multi-graphs with numerical attributes.

attribute. At each iteration, all the weights are updated based on the current clustering. Taking into account the calculated weights and a given number of clusters, vertices belonging to the same cluster should: (a) have the maximum similar connectivity, in terms of structure; and (b) have the minimum diversity in their attribute values.

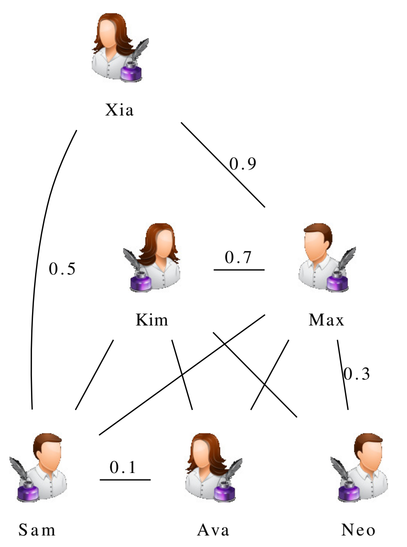
Formally, based on the clustering problem presented in Section 3.2, given an attributed multi-graph and a number of clusters K ($K \geq 2$), CLAMP goals are:

- Identify the importance of each edge-type and attribute. That is to compute the weights $w(t)$ and $w(\alpha)$, subject to $\sum_{\forall t \in \mathcal{T}} w(t) = 1$ and $\sum_{\forall \alpha \in \mathcal{A}} w(\alpha) = 1$.
- Calculate the importance of structural and attribute properties of the vertices, denoted as W_{links} and W_{attr} respectively.
- Find a fuzzy clustering $\Theta_{|\mathcal{V}| \times K}$, where $\Theta_{i,j}$ is the probability of v_i belonging to cluster C_j subject to $\sum_{j=1}^{K} \Theta_{i,j} = 1$ for all vertices.

7.1 CLAMP Clustering Model

7.1.1 Overview

In this section we present CLAMP (CLustering Attributed Multi-graPh). CLAMP combines in a unified distance measure for attributed multi-graphs the distance measures we present in Section 7.1.2 below. Leveraging the proposed unified distance function, in Section 7.1.3 we define the clustering objective function. In Section 7.1.4 we discuss the optimization process which seeks to assign each vertex: (a) to



(a) Example graph. The weight of unlabeled edges is 1.

	Kim	*Max*	*Sam*	*Xia*	*Neo*	*Ava*
Kim	1.0	0.7	1.0	0	1.0	1.0
Max		1.0	1.0	0.9	0.3	1.0
Sam			1.0	0.5	0	0.1
Xia				1.0	0	0
Neo					1.0	0
Ava						1.0

(b) Adjacency matrix

	Kim	*Max*	*Sam*	*Xia*	*Neo*	*Ava*
Kim	0	1.48	2.15	4.29	2.16	1.90
Max		0	1.15	1.85	3.88	1.80
Sam			0	1.52	2.75	1.87
Xia				0	3.61	3.17
Neo					0	2.50
Ava						0

(c) Similar connectivity matrix

Figure 6: Similar Connectivity on a Toy Graph

its closest cluster with the highest probability; and (b) to its furthest cluster with the lowest probability, with respect to the weights. Lastly, we present the CLAMP algorithm in Section 7.2.

7.1.2 Distance Measures

7.1.2.1 Similar Connectivity

It is a measure that represents how dissimilar two vertices are based on their outgoing edges. We formally define Type-Similar Connectivity as follows.

Definition 1. ***(Type-Similar Connectivity)*** *The type-similar connectivity of two vertices u, v for edge type t, denoted as $SC_t(u, v)$, is given by:*

$$SC_t(u,v) = \sum_{i=1}^{|\mathcal{V}|} (w_t(u, v_i) - w_t(v, v_i))^2 \qquad (28)$$

where

$$w_t(u,v) = \begin{cases} w_t(u,v) & \textit{if} \quad (u,v) \in \mathcal{E}_t \\ 1 & \textit{if} \quad u = v \\ 0 & \textit{else} \end{cases} \tag{29}$$

Figure (6) presents a toy co-authorship graph.[2] Specifically, Figure (6b) depicts the upper part of the adjacency matrix of the toy graph of Figure (6a), and Figure (6c) presents the similar connectivity computed by Equation (28).[3] For instance, similar connectivity of authors Kim and Max is computed as follows:

$$SC(Kim, Max) = [w(Kim, Kim) - w(Max, Kim)]^2$$
$$+ [w(Kim, Max) - w(Max, Max)]^2 + [w(Kim, Sam) - w(Max, Sam)]^2$$
$$+ [w(Kim, Xia) - w(Max, Xia)]^2 + [w(Kim, Neo) - w(Max, Neo)]^2$$
$$+ [w(Kim, Ava) - w(Max, Ava)]^2 = (1 - 0.7)^2 + (0.7 - 1)^2 + (1 - 1)^2$$
$$+(0 - 0.9)^2 + (1 - 0.3)^2 + (1 - 1)^2 = 1.48$$

We see that $SC(Kim, Max) = 1.48$ while $SC(Xia, Neo) = 3.61$. Hence, as it is also seen from Figure (6a), authors $\{Kim, Max\}$ are more similar than authors $\{Xia, Neo\}$. The lowest similar connectivity between Sam and Max is justified by the fact that they are connected to each other and they have three out of four common neighbors (only Neo is not a common neighbor). Thus, low *Similar Connectivity* among two vertices denotes that they share common neighbors and should be grouped together.

Total Similar Connectivity measures the distance of two vertices based on all their outgoing edges. We calculate the Total-Similar Connectivity of two vertices

[2]Edge weights have been scaled to $[0, 1]$.
[3]Type-similar Connectivity can be calculated on directed graphs as well.

as follows:

$$SC(u,v) = \frac{1}{|\mathcal{V}|} \cdot \sum_{\forall t} w(t) \cdot SC_t(u,v), \sum_{\forall t} w(t) = 1 \tag{30}$$

where $w(t)$ is the importance of the graph view corresponding to edge type t.

7.1.2.2 Attribute Distance

We compute attribute distance of two vertices using a distance measure suitable for both numerical and categorical attributes [41]. Specifically, the attribute distance of vertices u and v, denoted as $AD(u,v)$, is given by:

$$AD(u,v) = \sum_{\forall \alpha} w(\alpha) \cdot \delta_\alpha(u,v), \sum_{\forall \alpha} W_\alpha = 1 \tag{31}$$

where $w(\alpha)$ is the importance of the graph view corresponding to attribute α, and $\delta_\alpha(u,v)$ is the attribute distance of vertices u and v for attribute α. Function δ_α depends only on the type of attribute α.

For numerical attributes scaled to $[0,1]$ we use: $\delta_\alpha(u,v) = (a_\alpha(u) - a_\alpha(v))^2$. For categorical attributes we use the Kronecker's delta function:

$$\delta_\alpha(u,v) = \begin{cases} 0 & \text{if} \quad a_\alpha(u) = a_\alpha(v) \\ 1 & \text{else} \end{cases} \tag{32}$$

Other distance functions such as Minkowski, Hamming or Semantic could be adopted as well. We leave this analysis for future work.

7.1.2.3 Unified Distance Function

In order to calculate the total distance between two vertices, we must combine and balance their similar connectivity and attribute distance. The problem is quite challenging because structural and attribute distances are two seemingly independent objectives [24, 101]. A possible solution which we adopt is to use a distance function that balances these two objectives using appropriate weighting factors:

$$d(u, v) = W_{attr} \cdot AD(u, v) + W_{links} \cdot SC(u, v) \tag{33}$$

where W_{attr} and W_{links} represent the importance of the attribute and structural properties of the vertices respectively. These weighting factors are fine-tuned by CLAMP based on the network properties and the clustering results during the clustering process. We defer more detailed discussion of the weighting mechanism to Section 7.1.4.

7.1.3 Clustering Model

Recall that our goal is to find the optimal solution that assigns each vertex to: (a) its closest cluster with the highest probability; and (b) its furthest cluster with the lowest probability, which ideally should be zero (usually it is a very small value). Intuitively, the optimal solution minimizes the weighted distance between all possible pairs of vertices and clusters. Thus, a naive approach would be to adopt traditional fuzzy clustering. That is, to find the optimal clustering that minimizes the function:

$$\sum_{i=1}^{|\mathcal{V}|} \sum_{k=1}^{K} (\Theta_{i,k})^f \cdot d(v_i, \mathcal{C}_k) \tag{34}$$

where $d(v_i, \mathcal{C}_k)$ is the distance of vertex v_i to cluster $\mathcal{C}_k$; and f is the so-called fuzzifier, a free parameter that takes real values in the range $(1, +\infty)$ and determines by how much clusters overlap [15, 48]. The fuzzifier is necessary for getting fuzzy clusters, but its optimal value depends on the dataset and can be defined only through experimentation [15].

However, the above objective is not suitable for the problem we study because using the unified distance measure of Equation (33) it expands to weighted sum of individual terms corresponding to attribute and structural distances. Hence, to minimize it we must assign weight 1 to the lowest term, i.e. $W_{attr} = 1$, and consequently zero weight to the highest term. The same holds for the attribute and edge-type weights, i.e. $w(\alpha_1) = 1$ and $w(\alpha_i) = 0 \, \forall i \neq 1$, if attribute α_1 yields the lowest sum of distances. Withal, the above objective is minimized if: (a) we ignore either structural or attribute properties of the vertices; and (b) we consider only the edge type or the attribute that yields the minimum distance among all possible pairs of vertices and clusters. To address these issues we perform negative entropy regularization [58] on all the weights, to 'force' them to be close to each other. Specifically, we perform regularization on weighting factors W_{links} and W_{attr} to limit weights diversity and make it impossible for attributes to dominate edge types and vice versa. Similarly, we regularize attribute and edge-type weights to prohibit the dominance of a specific attribute and edge type.

In this manner, we formulate the problem of fuzzy clustering an attributed multi-graph as the identification of the optimal clustering (membership probabilities) and weights that minimize the following objective function:

$$\sum_{i=1}^{\mathcal{V}} \sum_{k=1}^{K} (\Theta_{i,k})^f \cdot d(v_i, \mathcal{C}_k) + \lambda \cdot [W_{links} \log (W_{links}) \tag{35}$$

$$+W_{attr} \log (W_{attr}) + \sum_{\forall t} w(t) \log (w(t)) + \sum_{\forall \alpha} w(\alpha) \log (w(\alpha))]$$

subject to

$$\begin{cases} \sum\limits_{k=1}^{K} \Theta_{i,k} = 1, & \forall v_i \in \mathcal{V} \\ W_{attr} + W_{links} = 1, & W_{attr} > 0, W_{links} > 0 \\ \sum\limits_{\forall t} w_t = 1, & w_t > 0 \\ \sum\limits_{\forall \alpha} w_\alpha = 1, & w_\alpha > 0 \end{cases} \tag{36}$$

where:

- $\mathcal{C}_k$ is the representation of cluster k. Each cluster is characterized by $\mathcal{A}$ attribute values and connects to vertices by weighted edges of $|\mathcal{T}|$ types.[4] The attribute values of a cluster and the weights of its outgoing edges are computed based on its members, as we present in Section 7.1.4 below.

- $d(v_i, \mathcal{C}_k)$ is the unified distance of vertex v_i and cluster $\mathcal{C}_k$ (Equation (33)).

- $\lambda > 0$ is the regularization parameter that controls how much a solution is penalized according to weights entropy. High entropy equals to high weight diversity. Thus, the more the weights deviate the higher the regularization term (entropy) becomes, and the solution is penalized accordingly depending on the value of λ. The λ parameter needs to be tuned empirically or using cross validation techniques [98].

[4]Hence, $\mathcal{C}_k$ is a valid parameter to Equations (27) - (33).

7.1.4 Objective Function Optimization

Finding the global optimum of Equation (35) is computationally difficult (NP-hard). To tackle the high complexity of the problem, we adopt the technique of gradient descent [16, 35]. Gradient descent requires minimum computations at each iteration, converges quickly to a local optimum, does not impose new parameters into the model, and is being widely used by many clustering algorithms, i.e. K-means. According to gradient descent we iteratively optimize either the membership probabilities, the weights or the cluster representations while considering the other parameters fixed. Therefore, we make the following propositions that suggest how to update the parameters at each iteration to reach a minimum of the CLAMP objective function.

7.1.5 Cluster Representations

Proposition 1 (Cluster Outgoing Edges). *If partitioning and view weights are fixed then the objective function is minimized when the weight of the edge from cluster $\mathcal{C}_j$ to a vertex u of edge type t is given by:*

$$\omega_t(\mathcal{C}_j, u) = \frac{\sum_{i=1}^{|\mathcal{V}|} (\Theta_{i,j})^f \cdot \omega_t(v_i, u)}{\sum_{i=1}^{|\mathcal{V}|} (\Theta_{i,j})^f} \tag{37}$$

Proof of Proposition 1. Based on similar connectivity definition and excluding the attributes part and regularization terms of the objective function (because they yield to partial derivative equal to zero as they are constant) we have to minimize the following function:

$$\sum_{i=1}^{|\mathcal{V}|} \sum_{k=1}^{K} (\Theta_{i,k})^f \cdot W_{links} \cdot \sum_{\forall t} w(t) \cdot SC_t\left(\mathcal{C}_k, v_i\right) \tag{38}$$

We differentiate the above equation with respect to $w_t(\mathcal{C}_k, u)$:

$$\begin{aligned} \frac{\partial O}{\partial w_t(\mathcal{C}_k,u)} &= \sum_{i=1}^{|\mathcal{V}|} (\Theta_{i,k})^f \cdot W_{links} \cdot w(t) \cdot 2 \cdot w_t(\mathcal{C}_k, u) \\ &- \sum_{i=1}^{|\mathcal{V}|} (\Theta_{i,k})^f \cdot W_{links} \cdot w(t) \cdot 2 \cdot w_t(v_i, u) \end{aligned} \tag{39}$$

Setting the derivative equal to zero and solving to $w_t(\mathcal{C}_k, u)$ the result is Equation (37). □

In order to calculate the attribute values of a cluster we have two cases: (a) categorical attributes; and (b) numerical attributes.

Proposition 2 (Cluster Categorical Attributes). *If membership probabilities and view weights are fixed then for categorical attributes the objective function is minimized when:*

$$a_\alpha(\mathcal{C}_j) = \arg\max_{y \in \mathcal{D}_\alpha} \sum_{i=1}^{|\mathcal{V}|} \Theta_{i,j} | a_\alpha(v_i) = y \tag{40}$$

Proof Sketch for Proposition 2. The objective function considering only one categorical attribute can be rewritten as:

$$\arg\min_{\mathcal{C}_k} W_{attr} \cdot w(\alpha) \cdot \sum_{i=1}^{|\mathcal{V}|} \sum_{k=1}^{K} (\Theta_{i,k})^f \cdot \delta_\alpha(\mathcal{C}_k, v_i) \tag{41}$$

Expanding the summation, the no-zero terms are the ones for which $\Theta_{i,k} > 0$ and $\delta_\alpha(\mathcal{C}_k, v_i) = 1$. Since memberships are considered fixed, the objective function minimizes when Equation (40) holds. □

Equation (40) represents the weighted mode of the cluster [41]. That is, a categorical attribute of a cluster takes the attribute value that characterizes most of its members, according to the membership probabilities.

Proposition 3 (Cluster Numerical Attributes)**.** *If membership probabilities and view weights are fixed then for numerical attributes the objective function is minimized when:*

$$a_\alpha(\mathcal{C}_j) = \frac{\sum\limits_{i=1}^{|\mathcal{V}|} \left((\Theta_{i,j})^f \cdot a_\alpha(v_i)\right)}{\sum\limits_{i=1}^{|\mathcal{V}|} (\Theta_{i,j})^f} \tag{42}$$

Proof of Proposition 3. For numerical attributes we calculate the partial derivative of the objective function as follows:

$$\begin{aligned} \frac{\partial O}{\partial a_\alpha(\mathcal{C}_k)} &= W_{attr} \cdot \sum_{i=1}^{|\mathcal{V}|} \sum_{k=1}^{K} (\Theta_{i,k})^f \cdot w(\alpha) \cdot AD'(\mathcal{C}_k, v_i) \\ \frac{\partial O}{\partial a_\alpha(\mathcal{C}_k)} &= 0 \Rightarrow a_\alpha(\mathcal{C}_k) = \frac{\sum\left((\Theta_{i,k})^f \cdot a_\alpha(v_i)\right)}{\sum(\Theta_{i,k})^f} \end{aligned} \tag{43}$$

□

Intuitively, Propositions (1)-(3) update the outgoing edges and the attribute values of a cluster according to its members with respect to the membership probabilities.

7.1.6 Partitioning - Membership Probabilities

Proposition 4 (Optimise Partitioning)**.** *If clusters and view weights are fixed then the objective function is minimized when:*

$$\Theta_{i,j} = \left[\sum_{k=1}^{K} \left(\frac{d(v_i, \mathcal{C}_j)}{d(v_i, \mathcal{C}_k)}\right)^{\frac{1}{f-1}}\right]^{-1} \tag{44}$$

where $d(v_i, \mathcal{C}_k)$ is given by Equation (33).

Proof of Proposition 4. For calculating $\Theta_{i,k}$ we consider the partial derivative of O with respect to Θ. In order to take into account the constraints of the objective function we use the Lagrange multiplier L_i:

$$\sum_{i=1}^{|\mathcal{V}|} L_i \cdot \left(\sum_{k=1}^{K} \Theta_{i,k} - 1 \right) \tag{45}$$

Note that we add only the constraint $\sum\limits_{k=1}^{K} \Theta_{i,k} = 1$ and its Lagrange multiplier is denoted as L_i. Normally all the constraints must be included, but this does not affect the partial derivatives with respect to $\Theta_{i,k}$ as they are treated as constant. Taking the partial derivative of the objective function with respect to $\Theta_{i,k}$ yields:

$$\begin{aligned} \tfrac{\partial O}{\partial \Theta_{i,k}} = \;& W_{attr} \cdot f \cdot (\Theta_{i,k})^{f-1} \cdot AD(\mathcal{C}_k, v_i) \\ & + W_{links} \cdot f \cdot (\Theta_{i,k})^{f-1} \cdot SC(\mathcal{C}_k, v_i) \\ & + L_i \end{aligned} \tag{46}$$

Setting the derivative equal to zero we get:

$$-L_i = \; f \cdot \Theta_{i,k}^{f-1} \cdot d(\mathcal{C}_k, v_i) \tag{47}$$

$$\Theta_{i,k} = \left[\frac{-L_i}{f \cdot d(\mathcal{C}_k, v_i)} \right]^{\frac{1}{f-1}} \tag{48}$$

where $d(\mathcal{C}_k, v_i)$ is Equation (33). It also holds that $\sum\limits_{j=1}^{K} \Theta_{i,j} = 1$. Hence:

$$\sum_{j=1}^{K} \left[\tfrac{-L_i}{f \cdot d(c_j, v_i)} \right]^{\frac{1}{f-1}} = 1 \tag{49}$$

By replacing L_i, as given by Equation (47), and solving to $\Theta_{i,k}$ we get Equation (44). $\square$

Equation (44) confirms that $\Theta_{i,j}$ must be high for low $d(\mathcal{C}_j, v_i)$ and vice versa.

7.1.7 Optimizing Attribute, Edge-type and Global Weights

Proposition 5 (Optimize Attribute and Edge-Type Weights)**.** *If membership probabilities and clusters are fixed then the objective function is minimized when:*

$$w(t) = \frac{\exp\left[\frac{S_t \cdot \ln(2)}{-\lambda}\right]}{\sum\limits_{\forall i \in \mathcal{T}} \exp\left[\frac{S_i \cdot \ln(2)}{-\lambda}\right]} \tag{50}$$

$$w(\alpha) = \frac{\exp\left[\frac{AD_\alpha \cdot \ln(2)}{-\lambda}\right]}{\sum\limits_{\forall \beta \in \mathcal{A}} \exp\left[\frac{AD_\beta \cdot \ln(2)}{-\lambda}\right]} \tag{51}$$

where

$$S_t = W_{links} \cdot \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{K} (\Theta_{i,j})^f \cdot SC_t(v_i, \mathcal{C}_j) \tag{52}$$

$$AD_\alpha = W_{attr} \cdot \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{K} (\Theta_{i,j})^f \cdot \delta_\alpha (v_i, \mathcal{C}_j) \tag{53}$$

Proof of Proposition 5. We prove the proposition for attribute weights as follows; solutions for edge type weights and weighting factors W_{links} and W_{attr} are calculated following the same steps. Initially, we derivate the objective function with respect to attribute weights. By substituting Equation (31) and adding the term $L_2 \cdot (\sum_{\forall\alpha} w(\alpha) - 1)$ to the objective function, where L_2 is the Lagrange multiplier, we get the first derivative:

$$\frac{\partial O}{\partial w(\alpha)} = AD_\alpha + \lambda \cdot (\log(w(\alpha)) + 1) + L_2 \tag{54}$$

where

$$AD_\alpha = \sum_{i=1}^{\mathcal{V}} \sum_{k=1}^{K} (\Theta_{i,k})^f \cdot W_{attr} \cdot \delta_\alpha (\mathcal{C}_k, v_i) \tag{55}$$

By setting the derivative to zero, we get:

$$L_2 = -AD_\alpha - \lambda \cdot (\log(w(\alpha)) + 1) \tag{56}$$

$$w(\alpha) = \exp\left[\left(\frac{-AD_\alpha}{\lambda} + 1\right) \cdot \ln(10)\right] \cdot \exp\left[\frac{-L_2 \cdot \ln(10)}{\lambda}\right] \tag{57}$$

Taking into account the constraint $\sum_{\forall\beta\in\mathcal{A}} w(\beta) = 1$ we get:

$$\exp\left[\frac{-L_2 \cdot \ln(10)}{\lambda}\right] = \frac{1}{\sum_{\forall\beta\in\mathcal{A}} \exp\left[\left(\frac{-AD_\beta}{\lambda} + 1\right) \cdot \ln(10)\right]} \tag{58}$$

Substituting the above equation to Equation (57) the result is Equation (51). □

Intuitively, Proposition (5) suggests that we assign a 'score' to each edge-type and attribute according to Equations (52) and (53) respectively, based on the individual contribution of vertices to each edge-type and attribute, i.e. $\sum_{j=1}^{K} (\Theta_{i,j})^f \cdot SC_t(v_i, \mathcal{C}_j)$ is the contribution of vertex v_i to edge-type t. Then, edge-type and attribute weights are computed accordingly using Equations (50) and (51).

Proposition 6 (Optimize Global Weights)**.** *If membership probabilities, clusters and attribute and edge-type weights are fixed then the objective function is minimized when:*

$$W'_{links} = \frac{\exp\left[\frac{S_{links}\cdot\ln(2)}{-\lambda}\right]}{\mathcal{W}} \tag{59}$$

$$W'_{attr} = \frac{\exp\left[\frac{S_{attr}\cdot\ln(2)}{-\lambda}\right]}{\mathcal{W}} \tag{60}$$

where

$$S_{links} = \frac{1}{W_{links}} \cdot \sum_{\forall t} w(t) \cdot SC_t \tag{61}$$

$$S_{attr} = \frac{1}{W_{attr}} \cdot \sum_{\forall\alpha} w(\alpha) \cdot AD_\alpha \tag{62}$$

$$\mathcal{W} = \exp\left[\frac{S_{attr} \cdot \ln(2)}{-\lambda}\right] + \exp\left[\frac{S_{links} \cdot \ln(2)}{-\lambda}\right] \tag{63}$$

Similarly to Proposition (5), Equations (61) and (62) assign a ‘score’ to the set of edge-types and to the set of attributes according to the contribution of each vertex. The updated structural and attribute property weights are given by Equations (59) and (60) respectively.

We note that according to Propositions (5), (6) and Equation (33) vertex property weights and global weighting factors are interrelated. That is, an edge-type importance is the product of the global edges weight and the weight of the edge-type. The same holds for attributes. However, if we combine global and individual weights to reduce model parameters regularization of both global and individual weights will not be feasible and the set of attributes or edges may dominate. Moreover, weights combination should be done by modifying the proposed unified distance measure and consequently the objective function. Although such model sounds much simpler, it will consider of the same type both vertex attributes and connections. For instance, the attribute ‘gender’ and the edge-type ‘friends’ would be incorrectly considered of the same type. That is because property weights will be computed using the same formulas derived from a simplified unified distance measure that does not consider simultaneously the individual importance of the vertex properties and the sets of attributes and edges. Overall, a simplified model would neither capture the different type of information encoded in an attributed multigraph nor balance properly the vertex structural and attribute properties.

Algorithm 3 CLAMP - Attributed Multi-graphs Clustering Algorithm

```
Input: Attributed multi-graph $G$, number of clusters $K$
Output: Fuzzy clustering of the network denoted by a $|\mathcal{V}| \times K$ matrix $\Theta$
 1: Initialize iteration number:  $r \leftarrow 0$
 2: Initialize $K$ cluster prototypes
 3: Initialize weights using Equation (64)
 4: while true do
 5:    Update memberships $\Theta^{r+1}$ by Equation (44)
 6:    if  $\|\Theta^{r+1}$ - $\Theta^{r}\| < \delta$  then
 7:       return $\Theta^{r+1}$
 8:    end if
 9:    Compute clusters by Equations (37)-(42)
10:    Update weights by Equations (50)-(63)
11: end while
```

7.2 CLAMP Algorithm

Putting everything together, Algorithm (3) depicts the pseudo code of the proposed algorithm for clustering attributed multi-graphs. CLAMP following the gradient descent technique iteratively (r is the iteration number) updates the vertex memberships, the clusters and the weights according to the propositions in Section 7.1.4. It terminates when two successive partitionings differ less than a small threshold (i.e. convergence delta $\delta \approx 10^{-4}$).

To initialize the clusters we follow the approach of random selection. That is, K vertices are randomly selected as clusters. Alternatively, several centroid initialization methods could be extended and used in the proposed approach, such as the works of ([12, 74]), to preprocess the network aiming to reduce the number of iterations and/or improve clustering accuracy. We adopt random selection because it is the fastest approach and does not require specialized knowledge or preprocessing-analysis of the network [77], although it yields slightly different results for the same input.

The initial weights are computed by equations:

$$W_t = \frac{1}{|\mathcal{T}|}\ ,\ W_\alpha = \frac{1}{|\mathcal{A}|}\ ,\ W_{attr} = W_{links} = \frac{1}{2} \tag{64}$$

We experimentally observed that weights initialization does not affect results quality, but it may affect (increase or decrease) the number of iterations until convergence. As we observed in our experimental evaluation CLAMP converges in less than 10 iterations.

The time complexity of CLAMP (Algorithm (3)) is the sum of the costs for calculating memberships, updating clusters and updating vertex property weights. The worst-case time complexity for calculating Θ is $O(|V| \cdot K \cdot (|\mathcal{A}| + |\mathcal{T}|.|\mathcal{V}|)$; for updating clusters is $O(K.(|\mathcal{T}|.|\mathcal{V}| + |\mathcal{A}|.|\mathcal{V}|))$; and for updating the weights is $\approx O(|\mathcal{V}| \cdot K \cdot (|\mathcal{P}|))$. Putting all together, the total complexity of Algorithm (3) for R iterations is $\approx O(R \cdot K \cdot |\mathcal{V}| \cdot (|\mathcal{T}| \cdot |\mathcal{V}| + |\mathcal{P}|))$.

7.2.1 CLAMP Algorithm in MapReduce Model

To tackle the high complexity of the clustering attributed multi-graphs problem, we can easily parallelize CLAMP in a variety of parallel computational models. In the following paragraphs we describe a parallel implementation of our algorithm in the MapReduce model [28], that can significantly lower the runtime and improve the scalability of the algorithm.

Algorithm 4 CLAMP - Attributed Multi-graphs Clustering Algorithm on MapReduce

```
Input: Attributed multi-graph $G$, number of clusters $K$
Output: Clustering $\Theta$, Weights
 1: Initialize iteration number:  $R \leftarrow 0$
 2: Select randomly $K$ vertices as initial clusters
 3: Initialize weights using Equation (64)
 4: while true do
 5:    Run first Map Reduce job
 6:    if all clusters converged then
 7:       return
 8:    end if
 9:    Run second Map Reduce job
10:    Update weights using Equations (51), (60)
11: end while
```

The implementation in MapReduce confirms that CLAMP can be executed on modern cloud many- and multi-core architectures to scale to large datasets and keep runtime low. CLAMP iteratively updates the vertex memberships, the clusters and the weights according to the propositions in Section 7.1.4.

Algorithm (4) depicts the pseudo code of the proposed CLAMP algorithm (R denotes the iteration number). CLAMP process consists of two MapReduce jobs. The first job (Algorithm (5)) calculates the membership probabilities and updates the clusters; and the second job updates the weights. The clustering process terminates when at the end of the first job all clusters change insignificantly, i.e. less than δ.

MAP 1. Mappers of the first job compute the clustering (membership probabilities). Since each vertex can compute independently its memberships by knowing only the weights and the clusters, a mapper responsible for a vertex v_i emits

Algorithm 5 CLAMP - First MapReduce Job

```
Input: Attributed multi-graph $G$, number of clusters $K$, weights
Output: Clustering $\Theta$
    MAP 1
 1: Input:  Clusters $\mathcal{C}$, Weights, Set of vertices $S \subset \mathcal{V}$
 2: Output:  <$k$, [$i$, $\Theta_{i,k}$]> $\forall k$
 3: for all $v_i \in S$ do
 4:     for all $\mathcal{C}_k \in \mathcal{C}$ do
 5:         Compute membership of vertex $v_i$ to cluster $\mathcal{C}_k$ using Equation (44)
 6:         Output:  <$k$, [$i$, $\Theta_{i,k}$]>
 7:     end for
 8: end for
    REDUCE 1
 9: Input:  <$k$, list of [$i$, $\Theta_{i,k}$]>
10: Output:  <$k$, [$\mathcal{C}_k$]>
11: Compute cluster properties by Equations (37)-(42)
12: if  $\|\mathcal{C}_k^r$ - $\mathcal{C}_k^{r-1}\| \leq \delta$  then
13:     Increase counter of converged clusters
14: end if
```

<k, [i, $\Theta_{i,k}$]>, where k is the id of cluster k. In practice, a mapper may be responsible for a set of vertices, i.e. $\frac{|\mathcal{V}|}{N}$ where N is the number of MapReduce nodes.[5] The time complexity for calculating Θ is $O(\frac{|\mathcal{V}|}{N} \cdot K \cdot (|T|.|\mathcal{V}| + |\mathcal{A}|))$.

Computed memberships are grouped by cluster id and thus each reducer receives the membership probabilities of vertices to a specific cluster, i.e. $\mathcal{C}_k$.

REDUCE 1. A reducer is responsible to update a cluster and output its description, i.e. <k, [C_k]> where [C_k] consists of two vectors representing the attribute values and the outgoing edges of cluster k. It also checks whether the specific cluster has converged. The time complexity for updating clusters is $O(\frac{K}{N} \cdot |\mathcal{V} \cdot |\mathcal{P}|)$, assuming the maximum size of a cluster is $|\mathcal{V}|$.

[5]The number of vertices must be higher than the number of available MapReduce nodes, i.e. $|\mathcal{V}| \gg N$.

Algorithm 6 CLAMP - Second MapReduce Job

```
    MAP 2
 1: Input: Clusters $\mathcal{C}$, Weights, Set of vertices $S \subset V$, Vector $\Theta_i$
    $\forall v_i \in S$
 2: Output: <{$t|\alpha|T|A$}, $v_i$ contribution> $\forall v_i \in S$
 3: for all $v_i \in S$ do
 4:     for all $t \in T$ do
 5:         Compute contribution of $v_i$ to $t$
 6:         Output <$t$, $v_i$ contribution>
 7:     end for
 8:     for all $\alpha \in A$ do
 9:         Compute contribution of $v_i$ to $\alpha$
10:         Output <$\alpha$, $v_i$ contribution>
11:     end for
12:     Output <$A$, $v_i$ contribution to set of attributes>
13:     Output <$T$, $v_i$ contribution to set of edges>
14: end for
    REDUCE 2
15: Input: <{$t|\alpha|T|A$}, list of contributions (one value for each
    vertex)>
16: Output: $S_{links}$, $S_{attr}$, $S_t$, $AD_\alpha$
17: Compute $S_{links}$, $S_{attr}$, $S_t$ or $AD_\alpha$ by Equations (52), (53) or (62)
```

The second MapReduce job (Algorithm (6)) starts the weight update process.

MAP 2. The mapper responsible for a vertex v_i outputs its contribution to every attribute and edge-type. Thus, each mapper outputs many key-value pairs, with key being the id of an attribute or an edge-type.

The individual contributions for specific attribute and edge-type are grouped into a list and forwarded to the reducers.

REDUCE 2. Each reducer sums up the contributions and outputs S_t and AD_a depending on the input key.

Weights for the next iteration are then computed sequentially. The time complexity of weights update process is $\approx O(\frac{|\mathcal{V}|}{N} \cdot K \cdot (|\mathcal{P}|))$.

Summarizing, the total worst-case complexity of the proposed CLAMP algorithm (Algorithm (4)) is $\approx O(R \cdot \frac{|\mathcal{V}|}{N} \cdot K \cdot (|\mathcal{T}| \cdot |\mathcal{V}| + |\mathcal{P}|))$, where R is the number of iterations and N is the number of MapReduce nodes. Complexity analysis confirms that CLAMP is highly parallelizable suggesting that time complexity is reduced proportionally to the amount of available MapReduce nodes.

We note that if the set of clusters can not be held in memory or be distributed among all the mappers, i.e. because there are a lot of clusters, the calculation of memberships can still be computed in parallel as follows. Each mapper computes the distances between a vertex and the small set of clusters it holds, while a reducer computes the final memberships. Such an implementation does not increase the time complexity, though will have the overhead of an additional MapReduce job.

Chapter 8

Experimental Study

This chapter presents the experimental evaluation of the proposed methods. We firstly outline the datasets used for the experiments, the evaluation measures and the experimental setup. Secondly, we present results on synthetic datasets and three real-world datasets. Lastly, we conclude the section with a discussion on proposed methods.

8.1 Datasets

8.1.1 Synthetic Datasets

To generate synthetic datasets, we modified the state-of-the-art generator presented in [68] to capture the multiple edge types and the similar connectivity aspects. The generator in [68] is based on planted partitions model [26]. It generates dense clusters where vertices are connected by only one edge type. For each cluster two parameters control the density and attribute homogeneity of the cluster. Following the same approach we split the $|\mathcal{V}|$ vertices into K clusters, i.e. K blocks, and

Table 5: Synthetic and Real-world Datasets Used for Evaluation. N and C are the number of numerical and categorical attributes respectively

Dataset	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{T}\rvert$	Weighted
Synthetics	{100, 500, 1000, 5000}	≈ 1000 - 1230000	N={2, 4, 8, 16}	{1, 2, 4, 8, 16}	No
GoogleSP-23-Name	1297	60204	5 (N=4, C=1)	1	No
GoogleSP-23-Path	1297	208752	5 (N=4, C=1)	1	No
GoogleSP-23	1297	268956	5 (N=4, C=1)	2	No
DBLP-10K	10000	65734	2 (N=1, C=1)	1	Yes
EU-Projects	1965	178623	7 (N=6, C=1)	2	Yes

use two parameters for each cluster to specify its similar connectivity and attribute homogeneity. Specifically, if a vertex in cluster $\mathcal{C}_j$ connects to a vertex u then the *Similar Connectivity parameter* specifies the least fraction of vertices in the group that also connect to vertex u. Vertex outgoing degrees follow a uniform distribution. The *Attribute Homogeneity parameter* specifies the least fraction of vertices in cluster $\mathcal{C}_j$ that share the same attribute value for each attribute. Numerical attributes are drawn from a uniform distribution, and categorical attributes are drawn from a Bernoulli distribution. In our experiments both parameters were set to 0.8. Hence, generated graphs exhibit both high attribute homogeneity and low similar connectivity. We generated synthetic attributed graphs and multi-graphs for various cluster properties as shown in Table 5. Particularly, we vary the number of vertices in {100, 500, 1000, 5000} with 4 attributes; the number of attributes in {2, 4, 8, 16} with 1000 vertices; and the number of edge types in {2, 4, 8, 16} for 1000 vertices and 4 attributes. For each variation, we generate five graphs.

Table 6: GoogleSP-23 Dataset - File Attributes and Their Description

Attribute	**Description**
FILE_SIZE	The size of the file in bytes
A_TIME	Last access time in seconds since Epoch
M_TIME	Last content modify time in seconds since Epoch
C_TIME	Time of most recent metadata change on Unix, or the time of creation on Windows in seconds since Epoch
FILE_TYPE	The type of the file as given by the command *file.* e.g. text/x-java;charset=us-ascii, application/x-executable; charset=binary

8.1.2 Real-world Datasets

We used software packages (GoogleSP-23), bibliography (DBLP-10K), and research/innovation projects (EU-Projects) datasets which we summarize in Table 5 and describe below.

Software Packages - *GoogleSP-23* is a dataset constructed by crawling the file system of a virtual machine in which 23 software packages downloaded from the Google code repository[1] were installed.

In GoogleSP-23 dataset, a vertex represents a file described by the five attributes shown in Table 6. There are two edge types in this dataset based on the file name and file path similarities which we refer as t_{name} and t_{path} respectively. Given two files f_a and f_b we calculate their file-system paths distance $d_p(f_a, f_b)$ using a string edit distance algorithm. The weight of the edge (f_a, f_b, t_{path}) is then given by:

$$w_{path}(f_a, f_b) = \begin{cases} 1 & \text{if} \quad d_p\left(f_a, f_b\right) < \text{avg}\left(d_p\right) \\ 0 & \text{else} \end{cases} \tag{65}$$

[1]Available online at `http://code.google.com`

Edges of type t_{name} are added similarly. Thus, GoogleSP-23 is an attributed multi-graph. Additionally, we use as attributed graphs the *GoogleSP-23-Name* and the *GoogleSP-23-Path* datasets in which vertices are connected only by edges of type t_{name} or t_{path} respectively.

Clustering these datasets can be used to identify software packages installed in Cloud providers, making Cloud services easily accessible and attractive to a wide range of users (Cloud application users, developers, administrators) [30]. For instance, Amazon has recently launched AWS Marketplace[2] , which is an online store that helps customers find, buy, and immediately start using the software and services they need on the Cloud. The advantage of these datasets is that clusters are known (software packages). Thus, we do not attempt to analyse the properties of software packages/files, but to compare the clustering results to the ground truth.

DBLP Bibliography - *DBLP-10K* dataset[3] consists of 10000 vertices representing the top authors from the complete DBLP dataset. Each author is described by two attributes: the number of publications and the primary area of interest. We consider authors with four research areas, namely databases (DB), data mining (DM), information retrieval (IR) and artificial intelligence (AI). A weighted edge between two authors represents the number of publications they have co-authored.

Clustering this dataset into clusters with similar connectivity and attribute homogeneity is expected to identify groups of authors from the same area that have

[2]AWS Marketplace, by Amazon, is available at `https://aws.amazon.com/marketplace/`, April 2017

[3]The full DBLP dataset is available online at `http://kdl.cs.umass.edu/data/dblp/dblp-info.html`.

worked with common researchers probably from different areas. Such clusters can help us identify outliers or recommend new collaborations.

EU Projects - *EU-Projects* dataset consists of organizations that participated in projects funded by the European Union under the FP7 and the H2020 framework programmes for research, innovation and technological development.[4] FP7 programme ran from 2007 to 2013, and H2020 programme started in 2014 and is expected to finish by 2020. In this dataset, we consider only the projects for which information was available by early 2015. That is 25808 and 2400 FP7 and H2020 projects respectively. Vertices represent organizations which are described by the following attributes: the number of H2020 and FP7 projects they are involved, coordinated, and participated in, and their country of origin. The 1965 organizations present in this dataset participated in at least 10 projects and they are connected by two types of weighted edges representing the number of H2020 and FP7 collaborations. In order to execute the algorithms that do not apply to attributed multi-graphs we consider only one edge between two organizations that represents the total number of collaborations.

Partitioning this dataset is expected to identify groups of organizations that share mutual H2020 and/or FP7 partners, have participated in approximately the same number of projects, and have the same country of origin. Such clusters can help us identify outliers and recommend new collaborations.

[4]This dataset is available online at European Union Open Data Portal - `http://open-data.europa.eu`.

8.2 Evaluation Measures and Comparison Methods

In the following experiments, we use the **entropy**, **similar connectivity** (Equation (30)) and **normalized mutual information (NMI)** to evaluate the results. Low entropy is equivalent to high attribute homogeneity between the vertices in the same cluster. Low similar connectivity represents that vertices in the same cluster have similar outgoing edges. Average entropy and similar connectivity are weighted by cluster sizes.

For a given clustering $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$, the entropy of attribute α in cluster $\mathcal{C}_k$ is given by:

$$entropy(\alpha, \mathcal{C}_k) = -\sum_{j=1}^{|\mathcal{D}_\alpha|} p_{kj} \log(p_{kj}) \tag{66}$$

where p_{kj} is the number of vertices in $\mathcal{C}_k$ which have the j^{th} value in the domain of attribute α divided by the size of the cluster $\mathcal{C}_k$. Entropy takes a value in the range $[0, \infty)$ despite the negative sign because p_{kj} is in the range $[0, 1]$. Entropy for a specific attribute for a clustering is given by Equation (67) and is the average entropy weighted by the size of the clusters. The goal of each clustering method is to achieve low overall entropy, which is computed as the average of the individual attribute entropies:

$$entropy(\alpha) = \frac{1}{|\mathcal{V}|} \sum_{j=1}^{K} (|\mathcal{C}_j| \cdot entropy(\alpha, \mathcal{C}_j)) \tag{67}$$

NMI represents the similarity between the obtained clustering and the ground truth. NMI takes values in the range of $[0, 1]$, with 1 corresponding to clustering that perfectly matches ground truth. Given a clustering $\mathcal{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ and the ground-truth $\mathcal{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_{K2}\}$, NMI is calculated as follows:

$$NMI(\mathcal{B},\mathcal{C}) = \frac{H(\mathcal{C}) - H(\mathcal{B}|\mathcal{C})}{\min(H(\mathcal{B}), H(\mathcal{C}))} \tag{68}$$

where:

$$H(\mathcal{C}) = -\sum_{k=1}^{K} \frac{|\mathcal{C}_k|}{|\mathcal{V}|} \cdot \log\left(\frac{|\mathcal{C}_k|}{|\mathcal{V}|}\right)$$

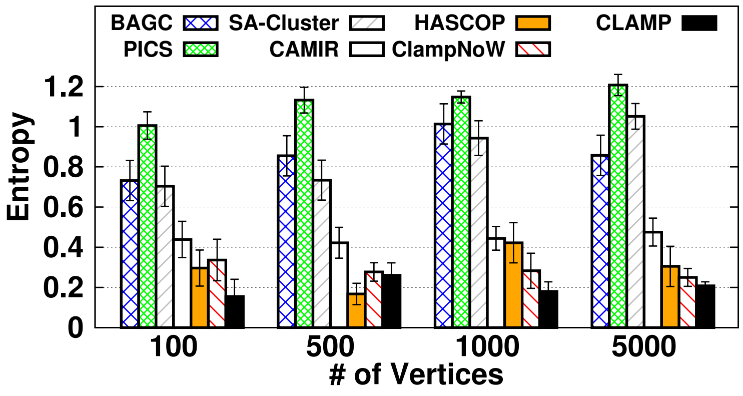
$$H(\mathcal{B}|\mathcal{C}) = -\sum_{i=1}^{K}\sum_{j=1}^{K_2} \frac{m_{ij}}{|\mathcal{V}|} \cdot \log\left(\frac{m_{ij}/|\mathcal{V}|}{|\mathcal{C}_j|/|\mathcal{V}|}\right)$$

where m_{ij} is the number of common vertices between clusters $\mathcal{B}_i$ and $\mathcal{C}_j$. In case $NMI(\mathcal{B},\mathcal{C}) = 1$, then the two clusterings are identical. At this point we must mention that NMI can be calculated, only if the ground-truth is available. Thus, we report NMI only for the experiments on synthetic datasets where clusters are known and for the experiment on GoogleSP-23 where ground truth is available.

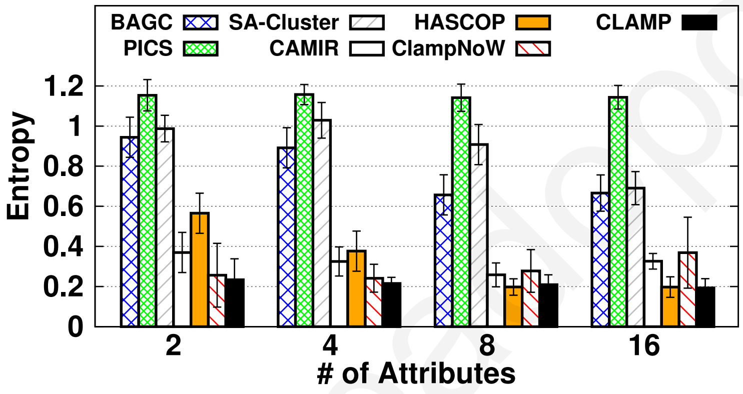
We evaluated the three proposed approaches, namely HASCOP, CAMIR and CLAMP presented in Chapters 5, 6, 7 respectively, against BAGC [90], SA-Cluster [101], PICS [8], and ClampNoW. The latter is a fictitious algorithm that uses CLAMP unified distance function but treats all the weights as constants as given by Equation (64).

Following experiments show average measurements out of five runs. Final clusterings were defuzzified by assigning each vertex to the cluster it belongs with the highest probability and thus use the same evaluation measures for all algorithms. The results of CLAMP and HASCOP are from multi-threaded implementations in Java 1.6. For vectors and matrices computations we use the JAMA library.[5] JAMA

[5]JAMA: A Java Matrix Package - `http://math.nist.gov/javanumerics/jama/`



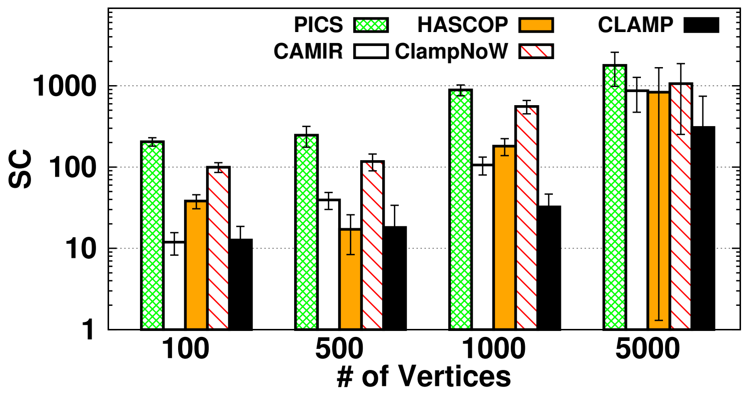
(a)



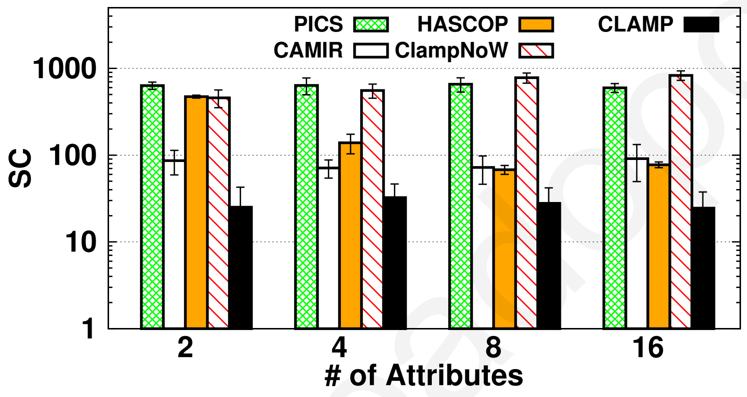
(b)

Figure 7: Entropy on synthetic attributed graphs

library provides sparse vector and matrix implementations and thus requires reasonable amount of memory. Each vertex is associated with a sparse vector of length $|\mathcal{A}|$ for its attributes and $|\mathcal{T}|$ sparse vectors of length $|\mathcal{V}|$ for its connections. For computing entropy and NMI we use WEKA data mining tool [85]. CAMIR, SA-Cluster, BAGC and PICS results are from sequential implementations in Matlab (source code provided by the authors). All experiments conducted on a Dell Server equipped with two 12 core Intel Xeon 3.47GHz processors and 80GB RAM.



(a)



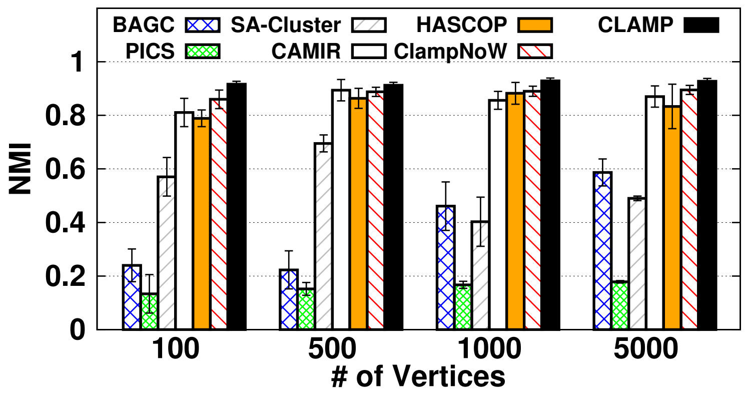
(b)

Figure 8: Similar Connectivity on synthetic attributed graphs. BAGC and SACluster are omitted from Figures because they achieve similar connectivity of at least one order of magnitude higher than the other approaches

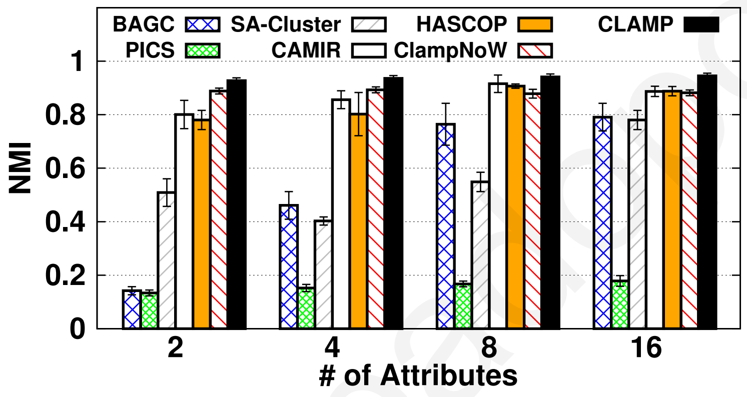
8.3 Evaluation on Synthetic Graphs

To study the clustering performance we generated synthetic attributed graphs and multi-graphs for various cluster properties. Particularly, we vary the number of vertices in {100, 500, 1000, 5000} with 4 attributes; the number of attributes in {2, 4, 8, 16} with 1000 vertices; and the number of edge types in {2, 4, 8, 16} for 1000 vertices and 4 attributes. For each variation, we generate 5 graphs.

According to Figures (7, 8, 9), CLAMP outperforms all its competitors in terms of entropy, similar connectivity and NMI. The high clustering accuracy of CLAMP



(a)



(b)

Figure 9: NMI on synthetic attributed graphs

is based on the presented weighting mechanism that correctly identifies the importance of the different vertex properties. PICS results in high entropy, high similar connectivity and low NMI, because it converges too early and returns few clusters, by using a self-tuning strategy to determine the number of clusters (Chapter 4). BAGC and SACluster achieve low entropy on all datasets, but they do not identify clusters characterized by low similar connectivity because they search for densely connected components. Specifically, they achieve similar connectivity of at least one order of magnitude higher than the other approaches, and thus we omit them from Figure (8). HASCOP and CAMIR achieve comparable clustering accuracy

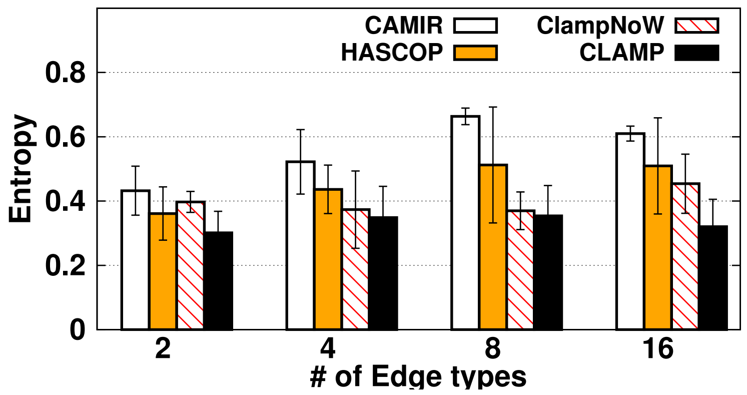


Figure 10: Entropy on synthetic attributed multi-graphs

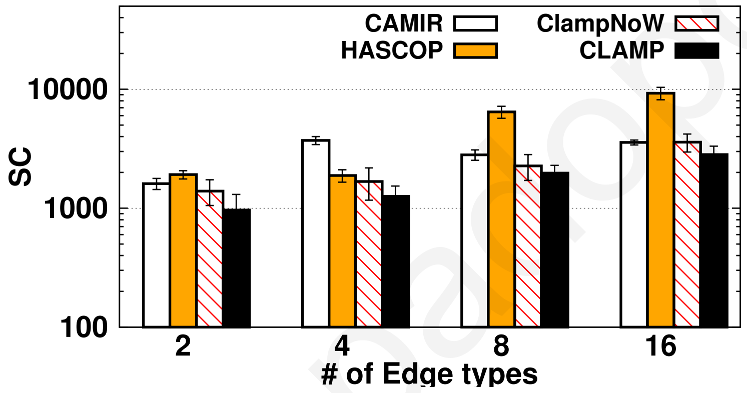


Figure 11: Similar Connectivity on synthetic attributed multi-graphs

on attributed multi-graphs (Figures (10, 11, 12)), since they also weigh the vertex properties efficiently. We further observe a rather steady performance of CLAMP. That is because it adapts the weights according to the properties of the datasets during the clustering process.

8.4 Evaluation on Real-world Graphs

8.4.1 GoogleSP-23 datasets

In this experiment algorithms have been executed for $K = \{20, 40, 60\}$. The goal is to identify the software packages (ground truth).

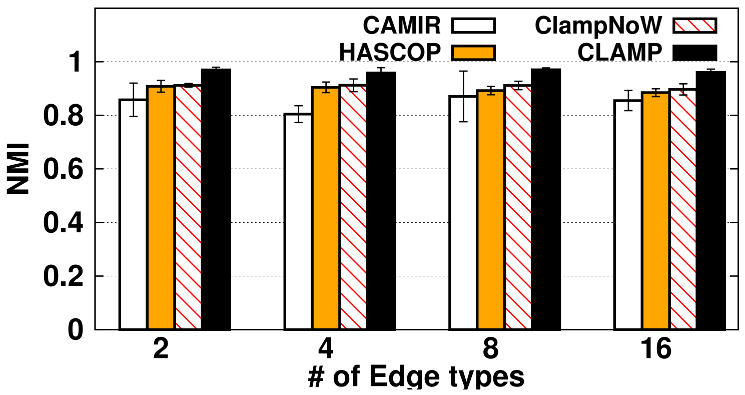


Figure 12: NMI on synthetic attributed multi-graphs

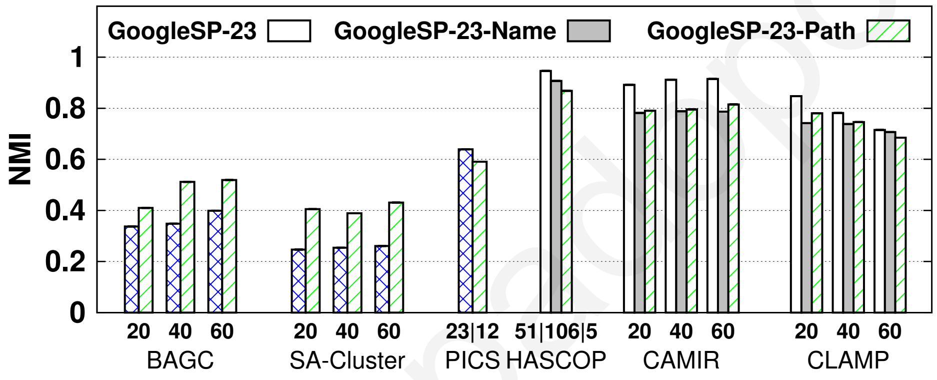


Figure 13: Clustering performance on GoogleSP-23 datasets. X-axis shows the number of clusters.

Figure (13) presents the normalized mutual information (NMI) of the results. It is evident that HASCOP, CAMIR and CLAMP exhibit superior performance over SACluster, PICS, and BAGC even for the datasets with only one edge type. SACluster and BAGC fail to achieve high NMI because they identify densely connected components while software packages are not densely connected [63]. PICS performs slightly better than SACluster and BAGC. Nonetheless, PICS results are not as good as the proposed approaches because it ignores the different importance of the vertex properties. We further illustrate that HASCOP, CAMIR and CLAMP

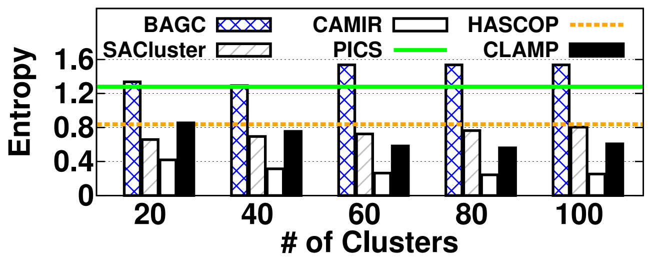


Figure 14: Entropy on DBLP-10K bibliography dataset. Since HASCOP and PICS use a self-tuning strategy to determine the number of clusters, both methods are denoted by straight lines

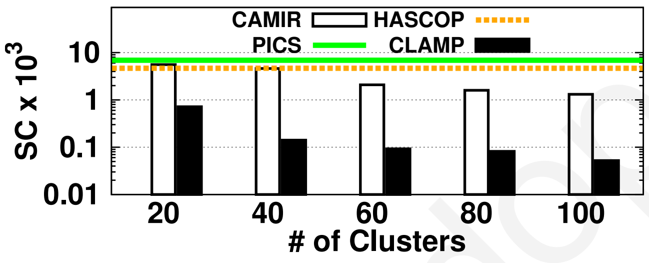


Figure 15: Similar Connectivity on DBLP-10K bibliography dataset. Since HASCOP and PICS use a self-tuning strategy to determine the number of clusters, both methods are denoted by straight lines. BAGC and SACluster achieve at least one order of magnitude higher similar connectivity

successfully identify a high percentage of software packages since they achieves high mutual information. HASCOP outperforms CLAMP because it has been tuned for software package identification. Despite its high mutual information, it returns 106 and 5 clusters for GoogleSP-23-Name and GoogleSP-23-Path datasets respectively, while there are 23 software packages in the dataset. CAMIR also achieves good results because based on spectral clustering technique it identifies clusters of arbitrary shapes and sizes. We further note that NMI for GoogleSP-23 dataset is increased compared to GoogleSP-23-Name and GoogleSP-23-Path for all algorithms confirming that *proposed methods exploit properly the existence of more than one edge type.*

8.4.2 DBLP-10K dataset

For DBLP-10K dataset we executed the algorithms for $K = \{20, 40, 60, 80, 100\}$. PICS and HASCOP returned 18 and 763 clusters respectively.

It is noted that since authors follow different careers and usually co-work with researchers from different locations and organizations, it is hard to find a group of authors that has many common co-authorships. This is also evident from the small average outgoing degree of a vertex (approximately $6, 5$). The weighting mechanisms of the proposed approaches 'captures' this property of the dataset and automatically assigns lower importance to structural properties of the vertices. Moreover, they identify the importance of each attribute and consider 'Are of Interest' more important than attribute 'Publications'. This is expected since 'Publications' entropy is higher and their objective function is optimized by assigning lower importance to it.

Figures (14, 15) present the average entropy and similar connectivity respectively, for multiple number of clusters. Proposed approaches achieve lower entropy than all their competitors (Figure (14)). Generally the more the clusters are the less the entropy is. This is due to the fact that the more the clusters are the easier it is to group together only vertices with the same attribute values. CAMIR resulted in lower entropy than CLAMP because it assigned much higher weight to attributes than edges. However, because of this weighting CAMIR results in much more similar connectivity than CLAMP (Figure (15)). Withal, Figures (14, 15) demonstrate an

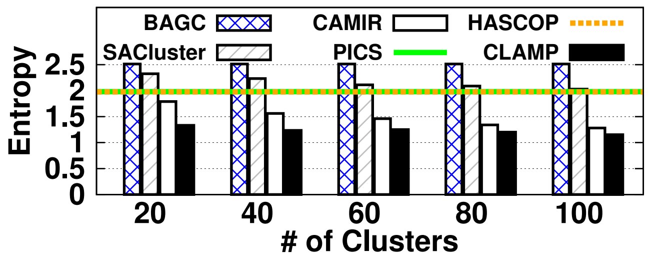


Figure 16: Entropy on EU-Projects dataset. Since HASCOP and PICS use a self-tuning strategy to determine the number of clusters, both methods are denoted by straight lines

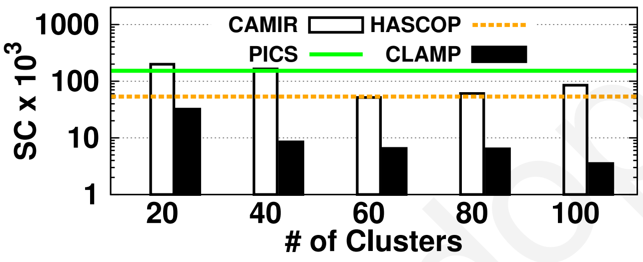


Figure 17: Similar Connectivity on EU-Projects dataset. Since HASCOP and PICS use a self-tuning strategy to determine the number of clusters, both methods are denoted by straight lines. BAGC and SACluster achieve at least one order of magnitude higher similar connectivity

overall good performance of the proposed approaches, while it confirms the efficiency of their weighting mechanisms.

8.4.3 EU-Projects dataset

For EU-Projects dataset we executed the algorithms for $K = \{20, 40, 60, 80, 100\}$. HASCOP and PICS returned 64 and 8 clusters respectively. Figures (16, 17) present the average entropy and similar connectivity respectively for multiple number of clusters. We observe that proposed approaches by adapting the vertex property weights achieve low entropy and similar connectivity, and over perform their competitors in most of the cases.

Table 7: Runtime (seconds) on Real-world Datasets. HASCOP and CLAMP runtimes are from multi-threaded implementations (Section 8.2).

	DBLP-10K	**GoogleSP-23-Path**	**GoogleSP-23-Name**	**GoogleSP-23**	**EU-Projects**
CLAMP	22110±215.918	69.843±12.026	48.135±8.421	179.128±35.190	32.514±14.938
HASCOP	32957±675.943	231.381±23.169	67.045±11.186	4595±237.189	3830 ±159.376
CAMIR	520.134±104.01	3.783±1.412	4.634±2.191	5.985±2.074	18.4± 2.77
BAGC	0.412±0.064	0.871±0.273	0.696±0.206	-	0.619±0.150
SACluster	404.581±68.065	38.583±3.682	30.583±3.682	-	51.931±19.730
PICS	494.993±0.000	19.932±2.915	15.012±1.256	-	38.030±1.873

8.5 Efficiency Study

We further examine the efficiency of our methods by measuring the execution time for the above datasets. Table 7 depicts the average runtimes for all number of clusters. The standard deviation is relatively high because the number of clusters affects the run time of the algorithms.

HASCOP and CLAMP are the slowest because they update the weights during the clustering process, and they perform fuzzy clustering. Around 50% of CLAMP and HASCOP run time is spent on the weights update process as we observed in our experiments. CLAMP is significantly faster than HASCOP for almost all used datasets. This improvement on run time is mainly because: (a) during the first few iterations of HASCOP the number of clusters is high (in the order of the number of vertices), which increases significantly the iteration time; and (b) HASCOP projects the original multi-graph at each iteration which is time-consuming. Despite the time overhead of CLAMP and HASCOP, as previous experiments demonstrate they achieve results of high quality.

SACluster and PICS generally require approximately the same time but *they are slower than CAMIR in most of the cases.* Also, PICS and BAGC do not consider

Table 8: Time complexity of proposed attributed multi-Graph clustering algorithms. It is noted that the number of iterations (R) differs for each method. Also, the number of clusters (K) for HASCOP is not constant.

Algorithm	**Time Complexity**
HASCOP	$\approx O(R \cdot K \cdot (\lvert\mathcal{E}\rvert + \lvert\mathcal{V}\rvert^2 \cdot \lvert\mathcal{A}\rvert))$
CAMIR	$\approx O(K \cdot \lvert\mathcal{V}\rvert^2 \cdot \lvert\mathcal{P}\rvert + \lvert\mathcal{V}\rvert^{K^2+1} \cdot \log \lvert\mathcal{V}\rvert)$
CLAMP	$\approx O(R \cdot K \cdot \lvert\mathcal{V}\rvert \cdot (\lvert\mathcal{T}\rvert \cdot \lvert\mathcal{V}\rvert + \lvert\mathcal{P}\rvert))$

the importance of the vertex properties which leads to lower complexity. BAGC is the fastest method, but achieves limited clustering quality as shown in previous σSections.

Lastly, the results are consistent with the time complexities presented in Table 4 (Section 4.3) and the time complexity analysis of proposed methods summarized in Table 8. Tables 7 and 8 show that CAMIR is faster than CLAMP, and CLAMP is faster than HASCOP.

8.6 Discussion and Connection to Previous Work

Clustering is a very important task in machine learning. We propose three methods for clustering attributed multi-graphs, in which objects are connected by multiple types of edges, and each object is characterized by a set of attributes. Table 9 depicts an overview of the proposed methods.

Despite the success of related works presented in Chapter 4, they differ from all proposed methods for at least one of the following aspects: (a) they assume equal importance of structural and attribute properties of the vertices; (b) they ignore the existence of multiple edge types; or (c) they aim to identify either community

Table 9: Proposed methods for clustering attributed multi-graphs

		HASCOP	**CAMIR**	**CLAMP**
Network Properties	Directed	✓		✓
	Weighted		✓	✓
	Numerical Attributes		✓	✓
	Categorical Attributes	✓		✓
Algorithm Properties	Parameter-free	✓		
	Fuzzy Clustering	✓		✓
	Automatic Properties Weighting	✓	✓	✓
	Optimization Technique	Heuristic	Spectral Clustering	Gradient Descent

outliers [52] or strongly connected components [7]. Proposed methods identify the importance of the various vertex properties in an automatic manner, handle the existence of multiple edge-types, identify clusters characterized by high similar connectivity and attribute homogeneity, and are highly parallelizable. Hence, they can exploit properly the computational power of modern many- and multi-core architectures to scale to large datasets. Nonetheless, each of them has its strong points and weaknesses.

HASCOP is parameter-free (does not require the number of clusters to be specified a priori), performs fuzzy clustering, and effectively identifies software packages installed in the file system of a computer or a virtual machine (Section 8.4.1). However, it does not handle either weighted attributed multi-graphs or numerical attributes, and requires more time than CAMIR and CLAMP (Section 8.5). HASCOP is the slowest because as an agglomerative-like algorithm during the first few iterations the number of clusters is much higher than the final number of clusters (it is in the order of the number of vertices), which increases significantly the iteration runtime. Also, it may converges quickly and consequently returns a high number of clusters.

CAMIR is the fastest of the proposed approaches, mainly because (a) it performs hard clustering; and (b) its weight identification process is decoupled from the clustering task and, thus it can take place only once. Upon completion of the weight identification process, the vertex property weights can then be used to rerun the algorithm with different parameters, i.e. number of clusters. In addition, because CAMIR follows a spectral clustering approach to partition the attributed multi-graph and to generate the final clustering, it identifies clusters of arbitrary shapes and sizes. However, it cannot apply to directed graphs since spectral clustering technique requires the input similarity matrix to be symmetric.

CLAMP is the first to perform fuzzy clustering on weighted directed attributed multi-graphs with heterogeneous attributes. CLAMP considers simultaneously the individual importance of the attributes and edge-types as well as the balance between the sets of attributes and edges, by assigning them different weights that are identified during the clustering process. Nevertheless, it is sensible to fuzzifier and regularization parameters that must be fine tuned for each dataset. Also, because it adopts gradient descent technique clusters initialization may significantly affect the results.

Overall, our extensive experimental evaluation on synthetic datasets and a diverse collection of real world information networks demonstrates the efficiency and effectiveness of proposed approaches on various scenarios.

Chapter 9

Use Case: Clustering-based Recommendation System

As Chapter 2 discusses, providing reliable, evidence-based recommendations is a challenging task useful in many real-world applications. In this chapter, we present our solution towards recommending collaborations to organizations that have participated in R&D projects funded by the European Union.

European research activities network[1] is a context rich dataset. It consists of more than 5000 organizations each of which is characterized by its activity type, i.e. public research institute or private organization, the number of projects it has participated in or coordinated, and its location at city or country level. Two organizations are connected by multiple types of edges depicted in Table 10. Edges represent their collaborations and role in projects, i.e. organization A coordinated a project in which organization B has participated; this relation is reflected by a weighted edge from A to B of type coordinator. Organizations participated to more

[1]The dataset is freely available for download at European Union Open Data Portal - `http://open-data.europa.eu`

Table 10: Description of Organization Connections

Connection Type	**Description**
participations	An undirected edge from A to B reflects the number of projects that both organization participated
coordinations	A directed edge from A to B reflects the number of projects that A coordinated and B participated
beneficiaries	A directed edge from A to B reflects the number of projects that A was beneficiary or host and B participated

than 15000 projects so far and they established more than 250K collaborations. Therefore, it would seem beneficial for organizations and researchers to find potential successful collaborators through an easily accessible recommender system.

Recommender systems are used in many application scenarios like social networks (e.g., Facebook, LinkedIn) and e-commerce sites (e.g., Amazon, Ebay). Most recommender systems rely on collaborative filtering techniques [17, 43] and aim to predict a user's interest in an item based on his past ratings of similar items or on the past ratings of similar users. Although collaborative filtering has become the standard method for the recommendation problem using only ratings of similar users/items to make recommendations without taking into account any other information results in low quality recommendations. To handle the issues of traditional recommendation systems, recommenders that consider contextual information, such as the user location when he rated a product, have attracted a lot of attention [2, 72]. Contextual recommenders have been recognized to improve results quality [72].

Nevertheless, existing recommender systems do not apply to European research activities network because of one or more of the following reasons. Firstly, they cannot efficiently combine different types of contextual information (e.g., the contexts with discrete values versus the ones with continuous values [2]). Secondly, they do not consider the multiple types of connections among the organizations. Thirdly, they suffer from high computational complexity (e.g., matrix factorization model [50] is impractical for extremely large dataset and multiple matrix factorizations are needed [29]).

In this chapter, we propose a clustering-based recommendation algorithm for the European research activities network. Our goal is to provide reliable, evidence-based partnership recommendations to organizations participated to R&D projects funded by the European Union. To do so we exploit the information from both the heterogeneous organizations' attributes and their multiple collaborations by applying CLAMP clustering algorithm to the European research activities network. CLAMP is suitable for this problem since it considers the multiple types of collaborations, handles the heterogeneous organizations' attributes, is quite fast and performs fuzzy clustering. Also, it achieves better results on EU-projects dataset (Chapter 8). We further propose a recommendation ranking method that derives and ranks the top N recommendations for each organization based on cluster membership probabilities.

Furthermore, we develop a prototype system that is able to automatically fetch and import updates from European open-data portal, thus updating its datastore and consequently constructing a new attributed multi-graph and performing clustering on it. Clustering results are then leveraged not only to provide recommendations but also to extract cluster descriptions, i.e. main topic such as cloud computing.

Results are stored in the system's datastore and then presented to the users upon request via a web-based user interface. To the best of our knowledge, this the first system to provide such services to the community. We find our contribution as a great tool in the hands of researchers and organizations towards establishing new collaborations and thus encourages research advances.

The remaining of this chapter is organized as follows. Section 9.1 overviews the related work. Sections 9.2 and 9.3 present the proposed solution and the developed system architecture. Section 9.4 outlines the evaluation section, and Section 9.5 concludes the chapter.

9.1 Background

Traditionally, recommender systems help users to find items suiting their wishes, needs or preferences. The different approaches to generate the recommendations can be divided in two broad categories: content-based and collaborative filtering approaches. Their goal is to predict the ratings of a user to items, and consequently recommend the items corresponding to the top N highest predicted ratings. Our goal is to recommend organizations (i.e. items) to organizations (i.e. users).

In content-based recommender systems, each item is characterized by an attribute vector describing its properties, e.g., for documents this vector is the Term Frequency-Inverse Document Frequency (TF-IDF) of the most informative keywords. A preference profile vector for each user is computed according to the items he preferred. Subsequently, the N most similar items to the user profile vector are recommended. Recommender systems based purely on content generally suffer from the problems of over-specialization. That is the system recommends only items that are similar to the ones liked by this user. Hence, the system may fail to recommend items that are different but still interesting to the user [2, 53].

Unlike content-based approaches, collaborative filtering approaches rely on the ratings of a user as well as those of other users in the system. The key idea is that the rating of a user for an item is likely to be similar to that of other users who rated similarly other items. In contrast to content-based systems, collaborative filtering ones can recommend items with very different content. Collaborative filtering methods can be grouped in the two general classes of model-based and neighborhood methods [29].

Model-based approaches use existing ratings to learn a predictive model. The general idea is to model the user-item interactions with factors representing latent characteristics of the users and the items. The models are trained using the available data and later used to predict the ratings of users for new items. Model-based approaches for the task of recommending items are mainly based on matrix factorization model [50] and include among others Support Vector Machines [29, 81], and Singular Value Decomposition (SVD) [67, 100].

In neighborhood-based collaborative filtering, the user-item ratings are directly used to predict ratings for new items. This can be done in two ways known as user-based or item-based recommendations. User-based systems evaluate the interest of a user for an item using the ratings of his neighborhood for this item. The neighbors of a user are typically the users who rated similarly the same items. Item-based approaches, on the other hand, predict the rating of a user for an item based on the item's neighborhood, i.e. the items which several users rated similarly.

Clustering has been used in various recommender systems to reduce the computation cost for finding the nearest user/item neighbors [9, 37, 95]. The typical use is to cluster users and/or items [36, 37, 92] and consider the users/items in the same cluster as the nearest neighbors. To predict a user's rating for an item they use the ratings of users/items in the user/item cluster, e.g. a user will rate an item with the average rating of users in his cluster. By doing so, the system has to consider only one cluster to provide recommendations instead of all users/items, thus limiting the processing time.

Despite the success of above recommender systems, they do not apply to the European research activities network because they ignore the existence of multiple collaboration types, i.e. consider a user having only one connection/rating to an item, and they do not handle the organizations' heterogeneous attributes.

9.2 Proposed Solution

The problem we study in this chapter is given the European research activities network to identify and rank for each organization a list of N organizations with which it can collaborate.

To exploit the organizations' heterogeneous attributes and their multiple-typed connections, we model the problem as an attributed multi-graph clustering task.

Specifically, we perform fuzzy clustering on the network to identify the most similar organizations, that is organizations in the same clusters. Then, cluster memberships are used to derive and rank new collaborations.

To cluster the European research activities network we employ CLAMP algorithm (Chapter 7) for the following reasons: (a) fuzzy clustering memberships can be used for recommendations ranking (Chapter 2), thus we excluded CAMIR; (b) HASCOP does not handle the weighted edges between the organizations; and (c) CLAMP is quite fast and performs better on EU-projects dataset (Chapter 8). Hence, given the network we fine-tuned CLAMP clustering process using cross validation technique to determine the model parameters. By doing so the clustering process does not require any user interaction.

Clustering results are used to perform the recommendation task. Given the clustering result of CLAMP, i.e. $\Theta \in \mathbb{R}^{|\mathcal{V}| \times K}$, we have to derive and rank the recommendation lists. Recall that an organization belongs to multiple clusters with different probabilities. A possible way to derive recommendations for an organization is to consider the members of its winner cluster (the cluster it belongs with the highest probability). On the other hand, we may ignore the winner cluster and recommend the top organizations from all the clusters it belongs. Either of the two results in information lose. Thus, we propose to combine the multiple memberships by computing a matrix $R \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ using the following equation:

$$R_{i,j} = \max_{\forall k}(\Theta_{i,k}.\Theta_{j,k}) \tag{69}$$

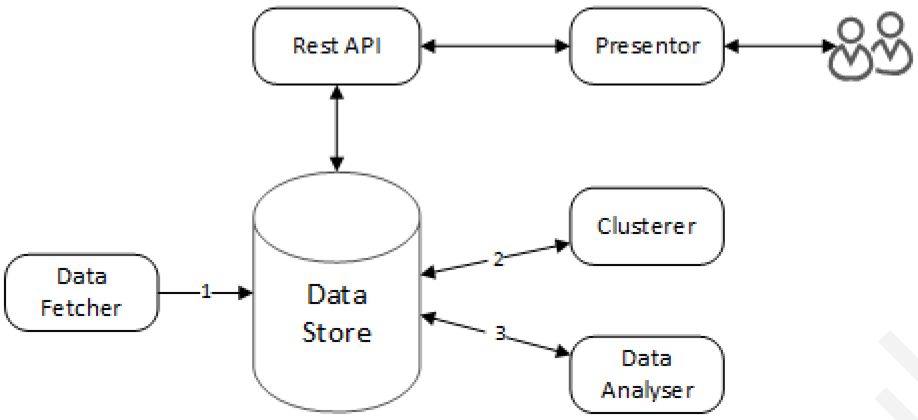


Figure 18: Architecture of the Clustering-based Recommender System

$R_{i,j}$ denotes the computed likelihood of organization i to collaborate with organization j according to the clustering Θ. The recommendation list for organization i is ranked according to the values in the i^{th} row of R, i.e. the organization with the highest value is ranked first followed by the organization with the second highest value and so on.

9.3 System Overview and Architecture

In this section, we describe the developed system prototype which integrates the proposed clustering-based recommender. The system is modular, extensible and performs the following operations: (a) imports new data, analyses and converts them to attributed multi-graph; (b) performs clustering; (c) analyzes clustering results to extract recommendations and group statistics; and (d) visually presents the results.

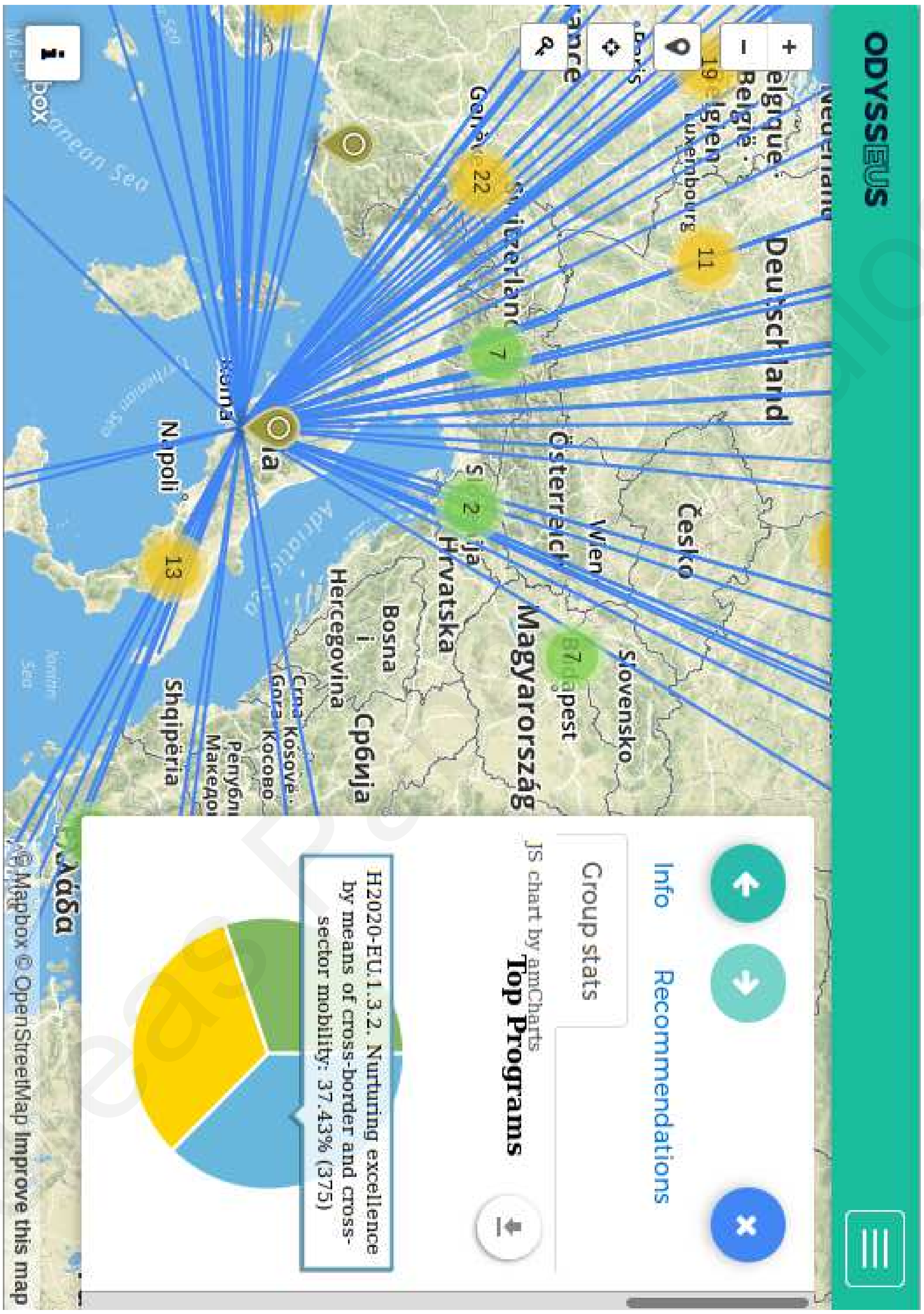


Figure 19: User Interface of the Clustering-based Recommender System

Figure (18) presents the main components of the system. *Data Fetcher* component is responsible to check periodically if there are any updates on the research activities network though the European open data portal. If so, it automatically fetches, parses, validates and stores the new data. *Clusterer* is then triggered to perform the clustering and store the results to the system's data store. *Data Analyser* fetches and analyses the clustering results to derive group statistics and a ranked recommendation list for each organization. Lastly, the *Presentor*[2] leverages the system's *Rest API* to expose a nice user-friendly interface accessible through standard web browsers. It presents organizations in an interactive map and allows users to view collaboration recommendations, filter the results and navigate through the various group statistics (Figure (19)).

Data Analyser extracts various group statistics based on the clustering results. Particularly, for each group it computes the winner topic, i.e. cloud computing, according to the organizations memberships to the group. To do so, it sums per topic each organization's contribution, which is the number of projects under the specific topic it has participated. By computing the winner topic per group we allow users to easily filter the results and get insights on various group properties. Other group characteristics such as main research activity type are computed similarly.

Additionally, the system allows users to select topics of interest and time period for projects start/end, and to optionally type some keywords. Based on user input the system automatically filters the organizations and constructs a new dataset. It

[2]The Presentor component has been implemented by Andreas Andreou, email: andreou.andreas@cs.ucy.ac.cy

then applies CLAMP clustering algorithm to the new dataset, derives recommendation lists and group statistics, and presents the updated results to the user. This is particularly useful when someone is interested only in organizations participated in projects having description matching specific keywords, i.e. recommender system.

9.4 Evaluation

The system is implemented using PHP, mysql and Javascript and thus it is easily accessible via standard web browsers. The clustering task is performed by a multi-threaded CLAMP implementation in Java.

9.4.1 European Research Activities Network

As of April 2017 the European Research Activities Network consists of 5198 organizations that participated to 16621 projects funded under the H2020 programme. Projects lifespan is between 1/1/2014 (start date) and 31/12/2023 (end date). We represent each organization by a vertex characterized by the attributes presented in Table 11. Organizations are connected by three types of edges depicted in Table 10. There are about 527K weighted edges/collaborations in the dataset. We note that network edges are based on organization participation to *funded projects* and may not capture all organization relationships. For instance, information regarding R&D project proposals that were not approved for funding is not publicly available.

Table 11: Description of Organization Attributes

Organization Attributes
The number of projects the organization participated
The number of projects the organization coordinated
The number of projects the organization was beneficiary or host
Location at city level
Activity type, i.e. research organization, higher or secondary education establishment, private for-profit entity, public body, other.

To statistically evaluate the effectiveness of our clustering-based recommender we performed a standard hold-out experiment. Specifically, we partitioned the European research activities network into two sets: a training set consisting of the collaborations established in the first 70% of the projects (regarding projects start date); and a test set consisting of the collaborations established in the last 30% of the projects. By doing so we guaranteed that the training set has the same properties as the original dataset. Training set consists of 3577 organizations participated in 7748 projects (30% of the total number of projects). The 3577 organizations are connected by approximately 245K connections. In the test set 48495 collaborations were established (among 1402 organizations) from which 15407 between unconnected organizations in the training set. Average new collaborations per organization is 37.3. However, the per-organization maximum and minimum number of new collaborations in the test set is 402 and 0 respectively. In other words, in the test set there are organization that established a lot of collaborations and organizations that did not establish new collaborations. The clustering algorithm is applied to the training set and the results are analyzed to see how well the system predicts the 15407 "unknown" collaborations in the test set.

9.4.2 Evaluation Measures and Comparison Methods

To evaluate the results we use the widely accepted Precision and Recall evaluation measures [59]. To compute precision and recall, given an organization's recommendation list, we firstly calculate the following values:

1. True Positives (TP): the number of recommended collaborations that are in the test set
2. False Positives (FP): the number of collaborations not in the test set that have been recommended
3. False Negatives (FN): the number of collaborations in the test set that have not been recommended

Precision and recall are defined as follows [59]:

$$Precission = \frac{TP}{TP + FP} \tag{70}$$

$$Recall = \frac{TP}{TP + FN} \tag{71}$$

Both precision and recall take values in $[0, 1]$. Precision measures the proportion of collaborations in a recommendation list that are correct (appear in the test set), and indicates how well the system separates the collaborations in the test set from random collaborations. Recall measures the proportion of collaborations in the test set that have been recommended ($TP+FN$ is the size of the test set), and indicates how well the system predicts the recommendations in the test set. An ideal system achieves both precision and recall of value 1. However, there exists an important trade-off between precision and recall: allowing for a longer list of recommendations improves recall but is likely to reduce precision. Improving precision often worsens recall [11]. Total precision and recall are taken as the average for all organizations. Precision@N and Recall@N refer to precision and recall values when we consider only top N items in the recommendation lists.

We note that a recommended collaboration that is not in the test set does not imply that these two organizations are not related or will never collaborate, e.g. some organizations may have already collaborated on an EU-project proposal that was not funded. This fact may result in low precision and recall values.

We compared our system for different values of N to SVD and KNN [14] recommenders provided by mahout machine learning library. [3] SVD and KNN parameters are set to the default values in mahout library. The input of SVD and KNN is the collaboration matrix, i.e. the preference/rating of organization A for organization B is the number of their collaborations. We also present the results of a fictitious algorithm that applies CLAMP and then recommends to an organization the top organizations in its winner cluster.

9.4.3 Evaluation Results

Figures (20, 21) show the average precision and recall values for multiple number of recommendations (N). We observe that the proposed recommender on average

[3]`https://mahout.apache.org/users/algorithms/recommender-overview.html`

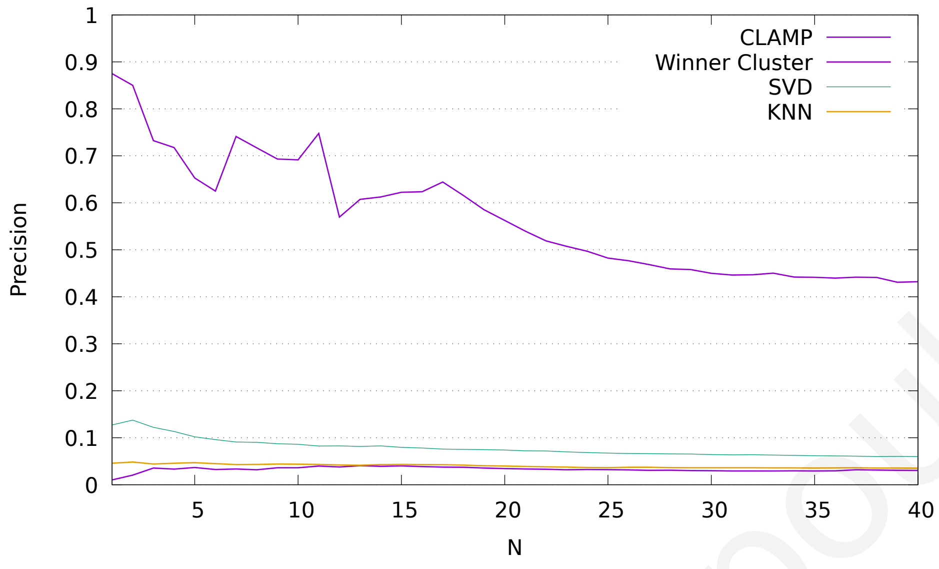


Figure 20: Precision for Recommending New Collaborations

predicts correctly more than half of excluded collaborations. Also, it achieves Precision@5 of about 0.7 (Figure (20)). Reaching high precision at low N is very important for a recommender ranking system indicating that recommendations are correctly ranked on top of the recommendation lists. Proposed method also achieves high (≈ 0.85) Recall@40 (Figure (21)). This suggests that our system recommended back almost all the recommendations in the test set. The success of our recommeder system is based on the proposed ranking method and the CLAMP ability to automatically identify the importance of each organization's property. Specifically, CLAMP assigned almost equal importance/weight to attributes number of participations, collaborations and hosted/beneficiary (0.3), and lower importance to attributes activity type (0.06) and city (0.04). Also, it assigned slightly higher importance/weight to the edges than the attributes (0.6 and 0.4 respectively). This is reasonable since organizations with different activity types and from different cities

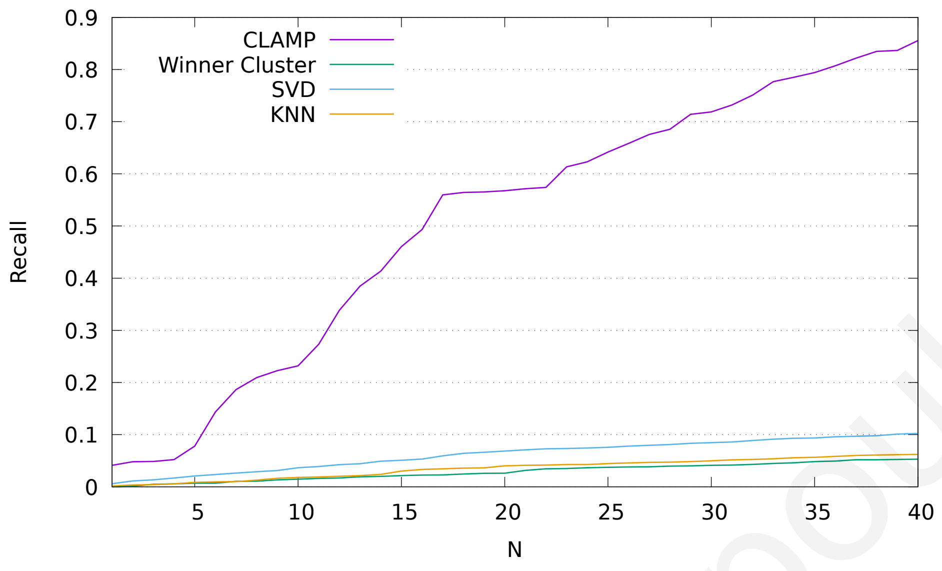


Figure 21: Recall for Recommending New Collaborations

collaborate together. Also, connections are more important than attributes since it is common for organizations with divert number of projects to collaborate.

Overall, we observe a sharp improvement over the SVD, KNN and WinnerCluster baselines which fail to handle the European Research activities network. SVD and KNN mainly fail because they ignore the multiple collaborations types and the organization heterogeneous attributes. Additionally, the collaboration matrix is relatively sparse and has low rank (number of linearly independent rows). That is, because many organizations participate in the same project and the average number of projects per organization is low ($\approx$ 2). Thus, the network does not contain sufficient information for SVD and KNN recommender (also known as the cold start problem) [9]. Proposed method and WinnerCluster use the same clustering results to provide recommendations. The poor performance of WinnerCluster confirms the superiority of proposed ranking mechanism. Also, proposed method recommends towards the top of the recommendation lists the connections in the test set. We conclude that our experimentation demonstrates the effectiveness and efficacy of the proposed clustering-based recommender.

9.5 Conclusions

In this chapter, we propose a clustering-based recommender for research collaborations at European level based on CLAMP clustering algorithm. Our experimentation shows a clear advantage of the proposed solution over well-established recommendation algorithms such as SVD and KNN. Proposed system can be used from researchers and organizations to find new partners and thus encourages research advances. Moreover, the presented recommender system demonstrates the applicability of the work presented in this thesis to a real-world scenario.[4]

For future work, since statistical offline evaluation can not replace a test with real user [49] and because our system is easily accessible to end users though a web browser, we plan a test with real user to further evaluate our system's performance.

[4]We note that the methods proposed in this thesis are generic and can be applied to other real world applications as well.

Chapter 10

Conclusions and Future work

This thesis focuses on the problem of clustering information networks modeled as attributed multi-graphs and introduces three novel methods that efficiently overcome various limitations of existing state-of-the-art methods. Specifically, the proposed methods for identifying clusters in an attributed multi-graph, named HASCOP (Homogeneous Attributes and Similar COnnectivity Patterns), CAMIR (Clustering Attributed Multi-graphs with Information Ranking) and CLAMP (CLustering Attributed Multi-graPhs): (a) identify automatically the importance of structural and attribute properties of the vertices; (b) consider the existence of multiple edge types; and (c) detect clusters that exhibit similar connectivity, in terms of structural, i.e. relate/connect to the same vertices, and attribute coherence, in terms of attributes, i.e. vertices are characterized by close attribute values. Also, CLAMP is the first method to perform fuzzy clustering on weighted directed attributed multi-graphs with heterogeneous attributes. Proposed methods can exploit properly the computational power of modern many- and multi-core architectures to scale to large datasets. They identify the importance of each vertex property using proposed weighting mechanisms. They so balance and combine the vertex properties efficiently. Our extensive experimental evaluation on synthetic datasets and a diverse collection of real world information networks: (a) confirms that proposed weighting mechanisms improve clustering quality; and (b) demonstrates the efficiency and effectiveness of proposed approaches on various scenarios.

Moreover, we are among the first to optimize similar connectivity on attributed multi-graphs. We demonstrate the importance of similar connectivity in identifying software packages and recommending new collaborations. Similar connectivity can be of great help in other real-world applications as well, i.e. friendship recommendations in online social networks.

We further leverage proposed methods to solve a practical issue. That is, how to offer reliable, evidence-based recommendations to European organizations. We propose a clustering-based recommendation method for the European research activities network. Organization and researchers can use our system to establish new collaborations. To the best of our knowledge, this is the first system to offer such services to the community.

The European research activities use case demonstrates the applicability of the work presented in this thesis to real-world scenarios. Since proposed methods are generic and can be applied to other real world networks as well, we plan to investigate the results of our methods on social network and protein interaction attributed multi-graphs. A Protein Interaction Network can be modeled as an attributed graph where a vertex is a gene, and there is an edge between two genes if there is an interaction between the proteins corresponding to these two genes. The attributes associated to a gene are simply the biological situations in which the gene was over-expressed. Examining the clusters of such network can raise interesting questions, i.e. “are there some interactions between an isolated cluster and the others, while no such interaction is known?”, or “is there any order in the activation of the genes in the same group?”. These questions can lead to interesting deeper investigations through biology experiments [62].

The work this thesis presents can be used to extract meaningful knowledge from the networks under study. Yet, it concerns only with full space clustering of static homogeneous attributed multi-graphs. Real-world information networks change continuously and consist of heterogeneous objects, while a set of objects may be ‘related’ based on a subset of their properties. To this end, we pose as future research directions the proposal and development of *scalable methods for subspace clustering of heterogeneous evolving networks.*

References

[1] Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 670–688. IEEE, 2015.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 191–226. Springer, 2015.

[3] Charu C. Aggarwal and Haixun Wang. A Survey of Clustering Algorithms for Graph Data. In Charu C. Aggarwal, Haixun Wang, and Ahmed K. Elmagarmid, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 275–301. Springer US, 2010.

[4] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. Mixed membership stochastic blockmodels. *J. Mach. Learn. Res.*, 9:1981–2014, June 2008.

[5] C.G. Akcora, B. Carminati, and E. Ferrari. Network and profile based measures for user similarities on social networks. In *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, pages 292 –298, aug. 2011.

[6] Leman Akoglu. Mining and modeling real-world networks: patterns, anomalies, and tools. Technical report, DTIC Document, 2012.

[7] Leman Akoglu, Hanghang Tong, and Danai Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 29(3):626–688, 2015.

[8] Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In *Proceedings of the 12th SIAM International Conference on Data Mining, SDM 2012*, pages 439–450. SIAM / Omnipress, April 2012.

[9] Xavier Amatriain and Josep M. Pujol. *Data Mining Methods for Recommender Systems*, pages 227–262. Springer US, Boston, MA, 2015.

[10] David Arthur and Sergei Vassilvitskii. How slow is the k-means method? In *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*, SCG '06, pages 144–153, New York, NY, USA, 2006. ACM.

[11] Iman Avazpour, Teerat Pitakrat, Lars Grunske, and John Grundy. *Dimensions and Metrics for Evaluating Recommendation Systems*, pages 245–273. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[12] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proc. VLDB Endow.*, 5(7):622–633, March 2012.

[13] Alessandro Baroni, Alessio Conte, Maurizio Patrignani, and Salvatore Ruggieri. Efficiently clustering very large attributed graphs. *arXiv preprint arXiv:1703.08590*, 2017.

[14] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17(4):305–338, 2016.

[15] James C. Bezdek, Robert Ehrlich, and William Full. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2–3):191 – 203, 1984.

[16] J.C. Bezdek, R.J. Hathaway, R.E. Howard, C.A. Wilson, and M.P. Windham. Local convergence analysis of a grouped variable version of coordinate descent. *Journal of Optimization Theory and Applications*, 54(3):471–477, 1987.

[17] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013.

[18] Brigitte Boden. *Combined clustering of graph and attribute data.* Apprimus-Verlag, 2014.

[19] Cecile Bothorel, Juan David Cruz, Matteo Magnani, and Barbora Micenkova. Clustering attributed graphs: Models, measures and methods. *Network Science*, 3:408–444, 9 2015.

[20] V. K. Bulitko. On graphs with given vertex neighborhoods, 1972.

[21] Florentina Bunea, Yang Ning, and Marten Wegkamp. Overlapping variable clustering with statistical guarantees. *arXiv preprint arXiv:1704.06977*, 2017.

[22] Wen-Yen Chen, Yangqiu Song, Hongjie Bai, Chih-Jen Lin, and E.Y. Chang. Parallel spectral clustering in distributed systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(3):568–586, 2011.

[23] Hong Cheng, Yang Zhou, Xin Huang, and Jeffrey Xu Yu. Clustering large attributed information networks: an efficient incremental computing approach. *Data Mining and Knowledge Discovery*, 25(3):450–477, 2012.

[24] Hong Cheng, Yang Zhou, and Jeffrey Xu Yu. Clustering large attributed graphs: A balance between structural and attribute similarities. *ACM Trans. Knowl. Discov. Data*, 5(2):12:1–12:33, February 2011.

[25] Fan R. K. Chung. *Spectral Graph Theory.* American Mathematical Society, Providence, RI, 1997.

[26] Anne Condon and Richard M. Karp. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2):116–140, March 2001.

[27] Dragoš M. Cvetković, Michael Doob, and Horst Sachs. *Spectra of Graphs: Theory and Application.* Academic Press, New York, 1980.

[28] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, January 2008.

[29] Christian Desrosiers and George Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[30] Marios D. Dikaiakos, Asterios Katsifodimos, and George Pallis. Minersoft: Software retrieval in grid and cloud computing infrastructures. *ACM Trans. Internet Technol.*, 12(1), July 2012.

[31] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering large graphs via the singular value decomposition. *Machine Learning*, 56(1):9–33, 2004.

[32] Maurizio Filippone, Francesco Camastra, Francesco Masulli, and Stefano Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recogn.*, 41(1):176–190, 2008.

[33] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1 – 44, 2016. Community detection in networks: A user guide.

[34] Esther Galbrun, Aristides Gionis, and Nikolaj Tatti. Overlapping community detection in labeled graphs. *Data Mining and Knowledge Discovery*, 28(5-6):1586–1610, 2014.

[35] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.

[36] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 625–628, Washington, DC, USA, 2005. IEEE Computer Society.

[37] Modou Gueye, Talel Abdessalem, and Hubert Naacke. *Dynamic Recommender System: Using Cluster-Based Biases to Improve the Accuracy of the Predictions*, pages 79–104. Springer International Publishing, Cham, 2016.

[38] S. Gunnemann, I. Farber, S. Raubach, and T. Seidl. Spectral subspace clustering for graphs with feature vectors. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 231–240, Dec 2013.

[39] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *Ann. Statist.*, 36(3):1171–1220, 2008.

[40] Hsin-Chien Huang, Yung-Yu Chuang, and Chu-Song Chen. Multiple kernel fuzzy clustering. *Fuzzy Systems, IEEE Transactions on*, 20(1):120–134, Feb 2012.

[41] Zhexue Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Min. Knowl. Discov.*, 2(3):283–304, September 1998.

[42] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, September 1999.

[43] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender systems: an introduction.* Cambridge University Press, 2010.

[44] S. Jenkins and S.R. Kirk. Software architecture graphs as complex networks: A novel partitioning scheme to measure stability and evolution. *Information Sciences*, 177(12), 2007.

[45] U. Kang, Brendan Meeder, Evangelos E. Papalexakis, and Christos Faloutsos. Heigen: Spectral analysis for billion-scale graphs. *IEEE Trans. Knowl. Data Eng.*, 26(2):350–362, 2014.

[46] Brian Karrer and M. E. J. Newman. Stochastic blockmodels and community structure in networks. *Phys. Rev. E*, 83:016107, Jan 2011.

[47] L. Kaufman and P. Rousseeuw. *Clustering by Means of Medoids.* Reports of the Faculty of Mathematics and Informatics. Delft University of Technology. Fac., Univ., 1987.

[48] Frank Klawonn and Frank Höppner. What Is Fuzzy about Fuzzy Clustering? Understanding and Improving the Concept of the Fuzzifier. In *Advances in Intelligent Data Analysis V*, volume 2810 of *Lecture Notes in Computer Science*, pages 254–264. Springer Berlin Heidelberg, 2003.

[49] Ron Kohavi and Roger Longbotham. *Online Controlled Experiments and A/B Testing*, pages 1–8. Springer US, Boston, MA, 2016.

[50] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009.

[51] Abhishek Kumar, Piyush Rai, and Hal Daume. Co-regularized multi-view spectral clustering. In J. Shawe-Taylor, R.S. Zemel, P.L. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1413–1421. Curran Associates, Inc., 2011.

[52] Nan Li, Huan Sun, Kyle C. Chipman, Jemin George, and Xifeng Yan. A probabilistic approach to uncovering attributed graph anomalies. In Mohammed Javeed Zaki, Zoran Obradovic, Pang-Ning Tan, Arindam Banerjee, Chandrika Kamath, and Srinivasan Parthasarathy, editors, *Proceedings of the 2014 SIAM International Conference on Data Mining, Philadelphia, Pennsylvania, USA, April 24-26, 2014*, pages 82–90. SIAM, 2014.

[53] Xin Liu and Karl Aberer. Soco: A social network aided context-aware recommender system. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 781–802, New York, NY, USA, 2013. ACM.

[54] Bo Long, Zhongfei (Mark) Zhang, Xiaoyun Wú, and Philip S. Yu. Spectral clustering for multi-type relational data. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 585–592, New York, NY, USA, 2006. ACM.

[55] Panagiotis Louridas, Diomidis Spinellis, and Vasileios Vlachos. Power laws in software. *ACM Trans. Softw. Eng. Methodol.*, 18(1):2:1–2:26, October 2008.

[56] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[57] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[58] Gideon S. Mann and Andrew McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Human Language Technologies 2007: The Conference of the North American Chapter of the*

Association for Computational Linguistics; Companion Volume, Short Papers, pages 109–112. Association for Computational Linguistics, 2007.

[59] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

[60] Marina Meila and William Pentney. Clustering by weighted cuts in directed graphs. In *In Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.

[61] Flavia Moser, Recep Colak, Arash Rafiey, and Martin Ester. Mining cohesive patterns from graphs with feature vectors. In *SDM*, pages 593–604. SIAM, 2009.

[62] Pierre-Nicolas Mougel, Christophe Rigotti, and Olivier Gandrillon. Finding collections of k-clique percolated components in attributed graphs. In Pang-Ning Tan, Sanjay Chawla, ChinKuan Ho, and James Bailey, editors, *Advances in Knowledge Discovery and Data Mining*, volume 7302 of *Lecture Notes in Computer Science*, pages 181–192. Springer Berlin Heidelberg, 2012.

[63] Andreas Papadopoulos, George Pallis, and Marios D. Dikaiakos. Identifying clusters with attribute homogeneity and similar connectivity in information networks. *IEEE/WIC/ACM International Conference on Web Intelligence*, 2013.

[64] Andreas Papadopoulos, George Pallis, and Marios D. Dikaiakos. Weighted clustering of attributed multi-graphs. *Computing*, pages 1–28, 2016.

[65] Andreas Papadopoulos, Dimitrios Rafailidis, George Pallis, and Marios Dikaiakos. Clustering attributed multi-graphs with information ranking. In *Database and Expert Systems Applications*, Lecture Notes in Computer Science. Springer International Publishing, 2015.

[66] E.E. Papalexakis, L. Akoglu, and D. Ience. Do more views of a graph help? community detection and clustering in multi-graphs. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 899–905, July 2013.

[67] Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

[68] Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused Clustering and Outlier Detection in Large Attributed Graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14. ACM, 2014.

[69] Marko Robnik-Šikonja and Igor Kononenko. Theoretical and Empirical Analysis of ReliefF and RReliefF. *Mach. Learn.*, 53(1-2):23–69, October 2003.

[70] Yiye Ruan, David Fuhry, and Srinivasan Parthasarathy. Efficient community detection in large networks using content and links. In *Proceedings of the 22nd International Conference on World Wide Web*, WWW '13, pages 1089–1098, Republic and Canton of Geneva, Switzerland, 2013. International World Wide Web Conferences Steering Committee.

[71] E. M. Rutledge, B. A. Miller, and M. S. Beard. Benchmarking parallel eigen decomposition for residuals analysis of very large graphs. In *2012 IEEE Conference on High Performance Extreme Computing*, pages 1–5, Sept 2012.

[72] Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 129–136. ACM, 2014.

[73] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27 – 64, 2007.

[74] Shimo Shen and Zuqiang Meng. Optimization of initial centroids for k-means algorithm based on small world network. In Zhongzhi Shi, David Leake, and Sunil Vadera, editors, *Intelligent Information Processing VI*, volume 385 of *IFIP Advances in Information and Communication Technology*, pages 87–96. Springer Berlin Heidelberg, 2012.

[75] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[76] Laura M. Smith, Linhong Zhu, Kristina Lerman, and Allon G. Percus. Partitioning networks with node attributes by compressing information flow. *ACM Trans. Knowl. Discov. Data*, 11(2):15:1–15:26, November 2016.

[77] Michael. Steinbach and Vipin Kumar. Cluster analysis: Basic concepts and algorithms. In *Introduction to data mining*. Pearson Addison Wesley, 1st edition, 2005.

[78] Karsten Steinhaeuser and NiteshV. Chawla. Community detection in a large real-world social network. In Huan Liu, JohnJ. Salerno, and MichaelJ. Young, editors, *Social Computing, Behavioral Modeling, and Prediction*, pages 168–175. Springer US, 2008.

[79] Heli Sun, Jianbin Huang, Jiawei Han, Hongbo Deng, Peixiang Zhao, and Boqin Feng. gSkeletonClu: density-based network clustering via structure-connected tree division or agglomeration. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 481–490, Washington, DC, USA, 2010. IEEE Computer Society.

[80] Yizhou Sun, Charu C. Aggarwal, and Jiawei Han. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. *Proc. VLDB Endow.*, 5(5):394–405, January 2012.

[81] Panagiotis Symeonidis. Matrix and tensor decomposition in recommender systems. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 429–430. ACM, 2016.

[82] Mirwais Tanai, Jongwan Kim, and JoongHyuk Chang. Model-based clustering analysis of student data. In Geuk Lee, Daniel Howard, and Dominik Slezak, editors, *Convergence and Hybrid Information Technology*, volume 6935 of *Lecture Notes in Computer Science*, pages 669–676. Springer Berlin Heidelberg, 2011.

[83] Wei Tang, Zhengdong Lu, and Inderjit S. Dhillon. Clustering with multiple graphs. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, pages 1016–1021, Washington, DC, USA, 2009. IEEE Computer Society.

[84] Wikipedia. Mixture model. `http://en.wikipedia.org/wiki/Mixture_model`, February 2013.

[85] Ian H. Witten, Eibe Frank, and Mark A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[86] M. M. Wolf and B. A. Miller. Sparse matrix partitioning for parallel eigenanalysis of large static and dynamic graphs. In *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–6, Sept 2014.

[87] Zhonggang Wu, Zhao Lu, and Shan-Yuan Ho. Community detection with topological structure and attributes in information networks. *ACM Trans. Intell. Syst. Technol.*, 8(2):33:1–33:17, November 2016.

[88] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. SCAN: a structural clustering algorithm for networks. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 824–833, New York, NY, USA, 2007. ACM.

[89] Zhiqiang Xu, James Cheng, Xiaokui Xiao, Ryohei Fujimaki, and Yusuke Muraoka. Efficient nonparametric and asymptotic bayesian model selection methods for attributed graph clustering. *Knowledge and Information Systems*, pages 1–30, 2017.

[90] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 international conference on Management of Data*, SIGMOD '12, pages 505–516, New York, NY, USA, 2012. ACM.

[91] Zhiqiang Xu, Yiping Ke, Yi Wang, Hong Cheng, and James Cheng. GBAGC: A general bayesian framework for attributed graph clustering. *ACM Trans. Knowl. Discov. Data*, 9(1):5:1–5:43, August 2014.

[92] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 114–121. ACM, 2005.

[93] Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1151–1156, Dec 2013.

[94] H Yu and Blair R Hageman. A framework for attribute-based community detection with applications to integrated functional genomics. In *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, volume 21, page 69, 2016.

[95] Sobia Zahra, Mustansar Ali Ghazanfar, Asra Khalid, Muhammad Awais Azam, Usman Naeem, and Adam Prugel-Bennett. Novel centroid selection approaches for kmeans-clustering based recommender systems. *Information Sciences*, 320:156 – 189, 2015.

[96] Qian Zhang, Ye Tian, Ting Wang, Feng Yuan, and Qiang Xu. Approxeigen: An approximate computing technique for large-scale eigen-decomposition. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '15, pages 824–830, Piscataway, NJ, USA, 2015. IEEE Press.

[97] Xiaolong Zheng, Daniel Zeng, Huiqian Li, and Feiyue Wang. Analyzing open-source software systems as complex networks. *Physica A: Statistical Mechanics and its Applications*, 387(24):6190 – 6200, 2008.

[98] Erheng Zhong, Wei Fan, Qiang Yang, Olivier Verscheure, and Jiangtao Ren. Cross validation framework to choose amongst models and datasets for transfer learning. In *Proceedings of the 2010 European Conference on Machine Learning and Knowledge Discovery in Databases: Part III*, ECML PKDD'10, pages 547–562, Berlin, Heidelberg, 2010. Springer-Verlag.

[99] Dengyong Zhou and Christopher J. C. Burges. Spectral clustering and transductive learning with multiple views. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 1159–1166, New York, NY, USA, 2007. ACM.

[100] Xun Zhou, Jing He, Guangyan Huang, and Yanchun Zhang. Svd-based incremental approaches for recommender systems. *Journal of Computer and System Sciences*, 81(4):717–733, 2015.

[101] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph clustering based on structural/attribute similarities. *Proc. VLDB Endow.*, 2(1):718–729, August 2009.

[102] Yang Zhou, Hong Cheng, and J.X. Yu. Clustering large attributed graphs: An efficient incremental approach. In *Data Mining (ICDM), IEEE 10th International Conference on*, 2010.