Knowledge and Information Systems manuscript No.
(will be inserted by the editor)

Local bilateral clustering for identifying research topics and groups from bibliographical data

Sara Elena Garza Villarreal · Satu Elisa Schaeffer

Received: Jun 18, 2014 / Revised: May 13, 2015 / Accepted: Jul 27, 2015

Abstract The structure of scientific collaboration networks provides insight on the relationships between people and disciplines. In this paper, we study a bipartite graph connecting authors to publications and extract from it clusters of authors and articles, interpreting the author clusters as research groups and the article clusters as research topics. Visualisations are proposed to ease the interpretation of such clusters in terms of discovering leaders, the activity level, and other semantic aspects. We discuss the process of obtaining and preprocessing the information from scientific publications, the formulation and implementation of the clustering algorithm, and the creation of the visualisations. Experiments on a test data set are presented, using an initial prototype implementation of the proposed modules.

Keywords Clustering · Knowledge discovery · Collaboration networks · Network analysis

1 Introduction

The structure of scientific collaboration networks provides information on the relationships between people and disciplines. Extracting clusters (also known as *communities*) from such networks helps to discover which researchers work together and on what topics. This information serves for the identification of human or knowledge resources and the establishment of new collaborations — between research groups, with the industry, or graduate students seeking a thesis adviser, for example.

Sara Elena Garza Villarreal
FIME, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, NL, Mexico

Satu Elisa Schaeffer
FIME, Universidad Autónoma de Nuevo León, San Nicolás de los Garza, NL, Mexico
E-mail: elisa.schaeffer@uanl.edu.mx

Scientific collaboration networks are generally represented as *bipartite graphs* that connect researchers to publications. This representation allows to detect both research groups (author clusters) and topics (article clusters). Our proposed method is based on local graph clustering [45] and community search [46]; it includes an *asymmetrical mode* that strengthens or weakens connections according to whether vertices are found together on the same or different clusters. The goal is to improve cohesion by the the gradual refinement of the resulting clusters. The interpretation of the clusters (e.g. leader discovery, topic activity rating) is facilitated through our *visualisation framework* based on tag clouds and spring layout. Our contributions include a preprocessor to detect duplicate authors, a similarity metric for generating edge weights, a clustering algorithm (using an objective function and a local search procedure derived from our previous work, operating in a novel manner, taking turns on each side of a bipartite graph), and a categorisation for cluster topologies.

The rest of this paper is organised as follows: Section 2 provides the foundations and Section 3 discusses related work. Then, Section 4 describes the proposed solution and Section 5 presents experiments and results. Finally, Section 6 provides conclusions and future work.

2 Background

In this section we discuss the fundamental definitions of the present work, which include graph theory and collaboration networks. The standard mathematical model for any network formed by elements and their interconnections is a *graph*. For a textbook on graph theory, we recommend that of Diestel [10], which is also available on-line. A graph is a pair of sets, $G = (V, E)$, where the elements of $V = \{v_1, v_2, \ldots, v_n\}$ are called *vertices* and the elements of E are, typically, pairs of distinct vertices (v, w) and are referred to as *edges*. Should larger subsets be considered edges as well, the term *hypergraph* is used. If the edges are assigned weights, the graph is *weighted*. If the edges are not bidirectional, that is $(u, v) \in E$ does not necessary imply that also $(v, u) \in E$, the graph is said to be *directed*. If at most one edge may connect each pair of vertices, the graph is *simple* (it is otherwise a *multigraph*). The vertices that form an edge are said to be *adjacent* to each other and *incident* to the edge. Two vertices v and w are said to be *neighbours* if they are connected by an edge $(v, w) \in E$. The *neighbourhood* of a vertex v is the set of its neighbours and is denoted by $\Gamma(v)$. The *degree* of a vertex v is the number of edges incident to it, which in a simple undirected graph equals the number of neighbours, $\deg(v) = |\Gamma(v)|$. A vertex with no adjacent vertices is said to be *isolated* and has degree zero. A vertex with an unusually high degree is a *hub*.

The *order* of the graph is the cardinality of its vertex set, $n = |V|$, and its *size* is the number of edges in it, $m = |E|$. A graph $H = (W, F)$ is a *subgraph* of $G = (V, E)$ if and only if $W \subseteq V$ and

$$F \subseteq \{(u, v) \mid u, v \in W \wedge (u, v) \in E\}. \tag{1}$$

A subgraph *induced* by a set $S \subseteq V$ is a subgraph $H = (S, F_S)$ where

$$F_S = \{(u, v) \mid u, v \in S \wedge (u, v) \in E\}, \tag{2}$$

that is, the subgraph where all edges between vertices in S that were present in E are conserved. In a general subgraph (Eq. (1)), some or all of these edges could be excluded. The *density* of (a simple graph) G is the proportion of edges present in it, comparing with the theoretical maximum: $\delta = m/m_{\max}$. For a directed graph, this maximum is $m^{\text{dir}}_{\max} = n(n-1)$: each vertex connects to every other vertex, supposing that edges only connect distinct vertices. For an undirected graph, it is half of that, as each pair is included just once:

$$m^{\text{undir}}_{\max} = \frac{n(n-1)}{2} = \binom{n}{2}. \tag{3}$$

An induced subgraph with density that is *maximal* (that is, no further vertices can be included without losing the property of density being equal to one) is called a *clique*. A graph that has density one is called *complete*.

A *partition* of a graph is a subdivision of its vertices into two or more subsets $S_1, \ldots, S_k$ such that $\bigcup_{i=1}^{k} S_i = V$ and $S_i \cap S_j = \emptyset$ for $i \neq j$. A *cover* is also a collection of subsets that includes every element of V in at least one subset, but without the requirement of empty intersections between the subsets. A graph is k-partite if a partition to k subsets exists such that

$$(u, v) \in E \Rightarrow u \in S_i \wedge v \in S_j, i \neq j, \tag{4}$$

that is, all edges connect vertices from *distinct* subsets and there are no edges connecting vertices within the same subset. A 2-partite graph is said to be *bipartite*. A *cut* is a 2-partition to two non-empty subsets.

A *path* from v to w in G is a sequence of edges such that v is the source vertex of the first edge, w is the target vertex of the last edge, and the ith edge ends in the same vertex that the $(i+1)$th vertex begins; this also holds for undirected graphs, where either vertex may be considered the source or the target arbitrarily. A path from a vertex to itself is the empty set. If there exists at least one path from each vertex v to every other vertex in V, the graph is *connected*. The *distance* $d_{v,w}$ from v to w is the number of edges in the shortest path that connect them in G.

2.1 Collaboration networks

Collaboration networks are complex networks [12, 38] that represent common participations among social entities, e.g. films involving different actors or publications involving different researchers. A subtype of this kind of network is a *scientific collaboration network*, in which entities and connections are specifically related to scientific research. A collaboration network is represented by a bipartite graph where one subset of vertices stands for the collaborators or executors (researchers) and the other subset stands for the acts of collaboration

(publications); the edges of the graph, thus, connect the collaboration acts with their corresponding executors. From the bipartite graph, two additional graphs or *projections* can be derived. A projection has no longer two subsets of vertices, but only one, which is connected *through* the other (see Fig. 1). Formally, denote one of the vertex subsets by S and the other by $T = V \setminus S$. We now define a *similarity measure* over the vertices in S based on the vertices in T; the construction also works interchanging the roles of the sets S and T. Let $v, w \in S$. We employ the Jaccard similarity for the neighbourhoods[1] of v and w in T

$$\xi(v, w) = |\Gamma(v) \cap \Gamma(w)| \,/\, |\Gamma(v) \cup \Gamma(w)|, \tag{5}$$

that is, the fraction of neighbours that v and w share in T. Other natural alternatives include using $|\Gamma(v)| + |\Gamma(w)|$ in the denominator. Should there be edge weights or multiplicities, the definition of similarity can be adjusted if the range of these values is finite. Using the above similarity measure, we derive from G two graphs: a projection of G onto S, $G_S = (S, E_S)$, and a projection of G onto T, $G_T = (T, E_T)$, where

$$E_K = \{(u, v) \in E \mid u \in K,\, v \in K,\, \xi(u, v) > \tau\}, \tag{6}$$

for $K \in \{S, T\}$ and a threshold parameter $\tau \geq 0$ (note that $\tau = 0$ conserves all edges, whereas high values for this parameter create a sparser graph by only keeping "strong" edges). One may conserve the values $\xi(u, v)$ or some transformation of them as edge weights in G_K. An example using edge weights is given in Fig. 2. If the collaborations can occur between groups of more than two actors (an actor being used here as a synonym for executor), a hypergraph is a natural representation for the projected actor network. However, typically the hyperedges are transformed into cliques connecting the participants of the collaboration each with the others; information is lost when several collaborations overlap (cf. Fig. 1 on the right).

For scientific collaboration networks, the two derived projections correspond to *authors* and *publications*; while the former projection — commonly called "co-authorship network" — connects researchers through paper co-authorships, the latter connects publications through common authors. Several variants of edge weights have been used in literature to portray aspects such as communication or co-authorship frequency [3, 11, 23, 26].

Complex networks in general are characterised in terms of structural measures, that is, functions that assign a numerical value which permits to identify the structural properties of the system [14]. In scientific collaboration networks, for example, distances between scientists are short, paths in the author projection usually contain the same individuals [32, 36], there is transitivity in acquaintances [35], and scientists with a high number of collaborations have a higher probability of participating in new collaborations [7, 8, 20, 30, 34, 52]; these structural properties are respectively known as the *small-world effect*, *high clustering*, and *preferential attachment* [38].

[1] As the graph is bipartite, necessarily $\Gamma(v) \subseteq T$ as well as $\Gamma(w) \subseteq T$.

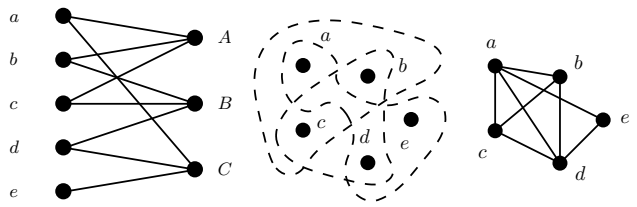


Fig. 1: On the left, a bipartite graph representing three acts of collaboration realised among five actors (a total of eight vertices), $V = \{a, b, c, d, e, A, B, C\}$. The collaborators (a, b, c, d, e) are drawn on the left and collaborations (A, B, C) on the right. The same set of collaborations is portrayed in the hypergraph in the middle that represents three collaborations, $E = \{(a, b, c), (b, c, d), (a, d, e)\}$, carried out among five collaborators, $V = \{a, b, c, d, e\}$; each edge is encompassed in a dashed line. On the right, these collaborations are represented as a clique in a simple, unweighted graph; edge weights could be used to preserve frequencies of collaboration (b and c collaborated twice).

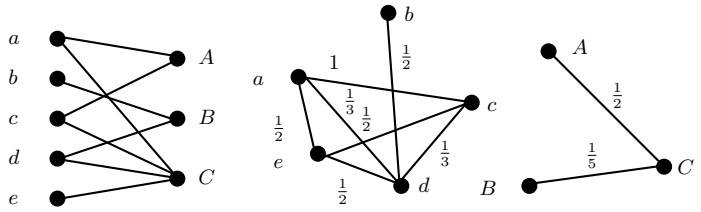


Fig. 2: The bipartite graph on the left gives rise to two weighted graphs on the centre and on the right, as defined by Eq. (5) and Eq. (6) with $\tau = 0$.

Another essential structural property in networks is the presence of communities or *clusters*, i.e. groups of vertices whose members have many (or denser) edges among themselves and only a few (or sparser) edges with respect to other elements in the network [15, 42]. Algorithms for community detection or *graph clustering* (the terms being largely interchangeable) extract such groups in a variety of ways [16, 44], for instance using *modularity* [37]. In scientific collaboration networks, graph clustering allows to detect *research groups* and *topics*. Section 3 discusses work related to the detection of these groups based on bibliographical data.

3 Related work

In this section we discuss briefly similar methods and applications presented in the literature for identifying clusters in collaboration networks and/or document collections. Newman [33] was one of the first to report findings on the structure of collaboration networks on the new wave of network science

that began with the new millennium. He discusses the extraction of such networks [31] as well as some of the relevant structural properties [32]. He mostly studies the network created by considering co-authorships in bibliographical databases, focusing solely on the author side of our bipartite formulation. This was to our knowledge the first published study on the structure of co-authorship networks based on automated processing of bibliographical data. The essential of the structural measures evaluated, namely publications per author and number of coauthors, behave as a power-law with an exponential cutoff; the analysis also confirms the six degrees of separation of Milgram [28] as well as a high clustering coefficient in some research areas.

ArnetMiner[2] is an on-line software for mining scientific collaboration networks that provides information, recommendations, and visualisations for research groups, papers, conferences, and trends [48]. More recently, Bian et al [4] present CollaborationViz, an interactive tool for visualising collaborations among researchers. This tool, based on a force-directed graph layout, has the capability of displaying different centrality measures, as well as a network timeline; furthermore, it gives the user the ability to track a researcher's career, make drill-downs, and predict links on the network.

Du et al [13] propose a community-detection algorithm for social network that functions efficiently for large sparse graphs, as those reported by Newman [33], with the purpose of using community discovery as a tool for social network analysis. The method allows for overlap in the communities, which is an essential feature for social networks in particular: a person often associates to two or more groups that are not necessarily subgroups of a larger structure but rather separate communities (consider for example work versus hobby in a general social setting and the diversity of methods and applications that a scientist may work on during a lifetime for the scientific collaboration scenario of the present work). However, Du et al [13] also focus only on the co-authorship upon analysing scientific collaboration networks and do not include information on article similarity.

An approach that combines information on the edge structure and the "content" of the vertices, which in our case maps to the similarity of article titles, is presented by Yang et al [50], although not towards the explicit computation of communities or topics, but rather in the modelling of link-existence based on content similarity, permitting characterisations of whether a set of vertices with specific content and edges among them should be considered to form a community.

Ramasco et al [43] also work with bipartite collaboration networks, proposing a model that mimics the structure of such networks. Moody [29] studies the scientific collaboration network (the author-to-author side of out network) — extracted from co-authorships — of sociologists over more than three decades, similarly to our present work, but no attempt is made at community discovery; the author concentrates on structural measures. Huang et al [19] report a similar study but on the computer science community, studying and mod-

[2] Available at: `http://arnetminer.org`.

elling not only static properties of a network snapshot but also discussing the way in which the collaboration evolves. Also, Perianes-Rodríguez et al [40] present a method for not only detecting research groups, but also describing and visualising these groups. Using the co-authorship network of a specific department as a case study, the detection is carried out by means of factor analysis. With regard to visualisation, a graph-based layout with node sizes and distinct colours is used to depict the groups found; in contrast with our method, this visualisation technique does not take time into account. Following a similar line, Ye et al [51] analyse and visualise the co-authorship network of a university. As part of their analysis is community detection over the network, and this is carried out by means of a k-clique algorithm. To perform the visualisation of a large network, Kruskal's minimum spanning tree algorithm is used.

More recently, Liu et al [24] propose a method for creating co-authorship networks, which aims at providing a more accurate metric for author importance; this metric, based on PageRank, considers citations over time (newer papers may not be as cited as old ones, but this does not imply that the former are less important). Gleiser and Danon [18] study yet another example of a collaboration network, where "co-authorship" means that two jazz musicians have recorded together, also reporting findings regarding the distribution of community sizes. The communities are computed with a global hierarchical algorithm based on edge betweenness. More recent works include the one by Larremore et al [22], which uses a bipartite stochastic block model for community detection in this type of network; this model works directly on the bipartite graph, not on one-mode projections.

Papadopoulos et al [39] provide a survey on community detection in another type of social network: social media. There instead of coauthoring publications, the users comment on the activity of other users, and having commented on a same activity can be considered as a collaboration of a kind. We believe that many of the methods examined by Papadopoulos et al [39] could be easily adapted for clustering scientific collaboration networks as well and hope to carry out a computational comparison within future work.

4 Bilateral clustering algorithm

In this section we describe the approach proposed for computing and visualising clusters of authors (research groups) and clusters of articles (research topics) in bipartite graphs representing bibliographical data. We will first define the necessary subprocedures.

4.1 Similarity scores

The *set similarity* of two sets A and B that have empty union and/or intersection is defined to be zero. For other sets, we use the following scoring

function

$$\tfrac{1}{3}\big((2i-u)\,/\,(u+(i-1)/i+1)\big), \tag{7}$$

that may take negative values, where i is the cardinality of the intersection $A \cap B$ and u is the cardinality of the union $A \cup B$. For the *text similarity* between two sets of string tokens A and B, we compute how many of the tokens are in the intersection of the two sets and then remove the intersection from both sets, and continue to analyse the similarity between $A \setminus (A \cap B)$ and $B \setminus (A \cap B)$. If the remaining sets are empty, the text similarity is one. Denoting by $\ell(S)$ the total length of the string tokens, $S \in \{A, B\}$, let

$$C = \arg \min_{S \in \{A,B\}} \ell(S) \tag{8}$$

and D be the other set. If the smaller of these sets is empty, the similarity is one (that is, there is no punishment for extra tokens, such as middle names of authors that sometimes are included and sometimes omitted). For example, *Abraham Benjamin Colt* yields a perfect match for *Abraham Colt*. We then analyse all pairs in $C \times D$ for similarity using the *edit distance* between the string tokens. For each $c \in C$, we find a $d \in D$ that minimises the edit distance. We take these minimum edit distances ε, each normalised individually by the length of the longer string token to obtain a value in $[0, 1]$. Upon computing the edit distance, we use addition cost equal to two, elimination cost equal to one, and replacement cost equal to three, using the standard dynamic-programming Levenshtein distance. We then calculate the similarity score for the two sets of string tokens A and B as

$$\frac{\ell(A \cap B)}{\min\{\ell(A), \ell(B)\}}\Big(1 - \frac{1}{\ell(C)} \sum_{c \in C} \varepsilon(c)\Big), \tag{9}$$

with C and D as defined above. This is to accommodate for spelling differences or errors in the names, such as *John* versus *Jon* or *Erick* versus *Eric*.

A research topic of its own right is how to distinguish between distinct authors who have the same or a similar name [49]. Heuristics and probabilistic models based on the similarity of the sets of coauthors and/or the keywords of the manuscript can be applied; if two authors of the same name work with the same people on the same topics, it is likely to be just one person, whereas if the topics and the coauthors differ, either the person has changed fields (which is entirely possible) or they are different people. Also self-citation frequency could be an indication — if we know that person A authored manuscript X, and then a person B of a similar name cites X, then B is likely to be A. We leave such filters to future work.

Existing online tools such as ResearchGate (`www.researchgate.net`) or the researcher profiles of Google Scholar (`scholar.google.com`) rely on user interaction: manuscripts that have author names resembling that of a registered user are offered listings of these manuscripts for manual revision so that the user can either confirm or deny authorship. It seems that at least Google Scholar takes advantage of coauthor information to automatically add

Table 1: Example of format for input data.

```
A djamal chaabane
A moncef abbas
T optimizing a linear function over an integer efficient set
Y 2006
J european journal of operational research
```

manuscript to the user profile when authorship is likely. Another (laborious) option is to compare the writing style if the full text of the manuscript is available, although automated attribution of authorship often requires long samples of the authors' work [9, 47].

4.2 String preprocessing

For cleaning input strings, we force them into lower case and replace by whitespace all occurrences of punctuation and HTML tags. We then eliminate any leading and trailing whitespace. Punctuation is removed and non-ASCII symbols are replaced with an ASCII version. Stemming is done with our implementation of the Porter stemming algorithm[3] [41], re-implemented simply to tweak it for the multiple languages contained in the input data.

We maintain a multilingual list of *stop words* consisting of *articles* (such as a, an, the, una, el, la), *prepositions* (such as from, auf, von, de), titles, prefixes and suffixes typical to names of persons (such as *Dr.*, *Jr.*), abbreviations of associations (such as *ACM*, *SIAM*, *IEEE*), and frequent words that contribute little to the semantics (such as *review*, *research*, *new*, *results*, *journal*, *letters*) — our experiments revealed that words like *problem* and *algorithm* should be omitted in future work. Words whose plural or singular form is a stop word are considered stop words.

4.3 Graph construction

We extract relevant information from a bibliographical repository; namely paper titles and authors. Journal names and years of publication are extracted for visualisation purposes. The format used is presented in Table 1. The author-name tokens are *not* stemmed, whereas journal names and article titles are. Any occurrences of stop words are eliminated. For each author, we store the name as an unordered set of strings. We associate to each author a set of *articles*. For each article, we store its set of authors, the title as a set of string tokens, the journal name as a set of string tokens, as well as the month and year of publication, where available. We store for each stemmed string also its original version for labelling purposes, whereas the stemmed one is employed in similarity computations.

Having parsed the input, we search for *duplicates*. First we compare all the author pairs to examine whether two objects are likely to refer to the same

[3] Available at `http://tartarus.org/martin/PorterStemmer/`.

person, which is time consuming (quadratic time complexity). We compute the text similarity between the set of name tokens for two authors. If it exceeds a merge-threshold parameter (we used 0.6 in our experiments), the sets of the name tokens are merged and one of the author objects is replaced by the other one in all the articles associated to the replaced one, and then the author object is added to a removal list. If the similarity is *not* above the merge threshold, but it is above a storage-threshold parameter (we use 0.4), we store the similarity in a hash table, using the pair of author objects as a key. The authors marked for removal are eliminated and the process is repeated until no further elimination occurs; the sets of name tokens of some authors have changed by the merges (for example, *A. Benjamin Colt* and *Abraham B. Colt* merge into *A. B. Abraham Benjamin Colt*), for which new pairs could now exceed one or both of the thresholds.

We then search for duplicate articles in the same fashion; it is not at all uncommon for the same publication to appear multiple times in bibliographical data with minor differences for example in journal name spelling (due to abbreviations) or in the title (due to typos or shortening). We use a stricter merge threshold (set to 0.8) and a looser storage threshold (set to 0.3) because we wish to associate articles that are not replicas but thematically related using the similarity of their titles in addition to co-authorship data. The remaining author and article objects form the vertex set of the bipartite graph; we will refer to these vertices simply as "articles" and "authors" although the preprocessing has altered their contents. An author is connected to an article if and only if that author is among the set of authors of that article.

4.4 Clustering

Clusters are computed starting with either an author vertex or an article vertex, traversing the author-to-author and article-to-article edge sets obtained of the two projections of the bipartite graph to determine which other vertices of the same kind should be included in that cluster. After each vertex has been assigned a cluster, the weights of the edges of each projection are adjusted according to the resulting clustering of the other projection.

The cluster creation is performed by local search. Initially, all clusters are set as undefined. To compute a cluster C, we begin with a *seed vertex* $s \in V$ and initially set $C = \{s\}$ and apply *simulated annealing* [21], a meta-heuristic that mimics the cooling of metal. We begin with an *initial temperature* $T = 10$ and cool on each iteration the current temperature t with a *cooling factor* $c = 0.95$; these values could be varied as parameters, but we leave that to future work. The ordering of the seed vertices during an iteration is a random permutation. If a *symmetric* clustering is requested, all vertices in a computed cluster are marked as assigned and will not be used again as seeds; for an *asymmetric* clustering where a cluster of a seed vertex may contain vertices that do not include that seed in their corresponding clusters, all vertices are used as seeds one by one.

The local search has two operations: *growth* (inclusion of an additional vertex in the cluster being computed) and *reduction* (the exclusion of a presently included vertex). To pick a step, we construct a *candidate set* consisting of all vertices presently included in the cluster, all neighbours of presently included vertices as well as vertices with a non-zero similarity to a presently included vertex. Then, one candidate is chosen uniformly at random. If it does not yet belong to the cluster and has not yet been fixed to some other cluster[4], it is included in the cluster (a growth step). If it does belong to the present cluster, it is excluded (a reduction step), with two exceptions: firstly, if the cluster contains a single vertex and the reduction mode is activated, the procedure exits with a singleton cluster, and secondly, when the "keep mode" is set, the seed vertex may never be removed from the cluster.

The fitness of the modified cluster is now evaluated. Namely, $\forall v \in C, \forall u \in \Gamma(v)$, we query the similarity $s(v,u)$ and average it with the edge weight[5], denoted by $w(v,u)$. If $u \in C$, we sum this to the *internal degree* d_i; otherwise we sum it to the *external degree* d_e. After this, we divide d_i by two; this is to overcome having summed all internal contributions twice. The *internal density* is then computed by normalising: $\delta_i = 2d_i/(|C|(|C|-1))$. The *relative density* is computed as $\delta_r = d_i/(d_i + d_e)$. The *fitness function* is then computed as $f = \delta_i \delta_r$. All singleton clusters are defined to have fitness zero.

If the fitness computed for a cluster candidate improves upon the present fitness, the modified cluster is *accepted*. If it improves upon the *best cluster* seen thus far for the given seed vertex, the cluster and its fitness are stored. If there is no improvement, the modification is accepted with probability $p = \exp(f_c - f_m)/t$ where f_c is the fitness of the cluster before the modification, f_m is the fitness of the modified cluster, and t is the temperature (as in simulated annealing). If the modification was *rejected*, the cluster is restored to its previous state. Upon rejection, a *stall counter* k is increased. When the counter reaches k_m (we use $k_m = 10$), the procedure is stopped; upon obtaining an improvement, the counter is reset to $k = 0$. After each iteration, we switch mode with probability $p_m = k/k_m$. The procedure always starts in growth mode.

A total of R *restarts* of the procedure is made (we set $R = 10$). At the end, the cluster that yielded the best fitness is returned as the result. All vertices in the produced cluster C are marked to have that cluster as their cluster for that iteration. When computing symmetrical clusters, only one iteration is done as nothing changes from the first iteration to the second. For each cluster, upon the postprocessing phase we compute a penalty: each missing internal edge provokes a unit penalty, as well as each external edge connecting the cluster to the rest of the graph. The total number of penalty units is then normalised by the total vertex count.

[4] In the symmetric mode, once a cluster is computed, the included vertices are no longer available for inclusion in future cluster computations.

[5] The *weight of an edge* $w(v,u)$ is computed as the multiplicity of that edge; for purposes of the clustering phase, the edges are treated as directed and the weight is normalised by the degree of vertex v, making the directed edge weight *asymmetric*.

For an *asymmetric* clustering, we first check on each iteration whether a cluster $C(b)$ for b has been defined $a \in C(b)$. If so, we amplify the current similarity score, multiplying it by an amplification factor (set to 1.2 in our experiments). On the contrary, if the cluster exists but a is not included in it, we punish the score, multiplying it with a dampening constant (set to 0.9 in our experiments). We then do the same for $C(a)$ to check whether $b \in C(a)$. This is repeated on each iteration to reward symmetrical assignment and thus filter the clusters; when using the symmetrical clustering mode, no iteration is needed as nothing will change in the graph structure during an iteration. The iteration stops when either the penalty gets below a threshold for the first time or the difference in the penalties of two subsequent iterations is lower than this threshold.

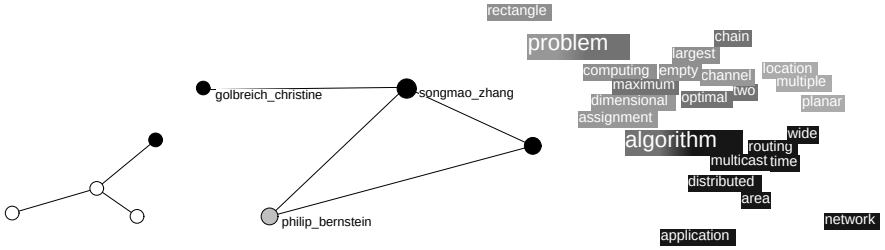
When *outputting* the cluster information, we set a threshold on the cluster order to withhold the printout of very small clusters. For each outputted clusters, a drawing of the subgraph induced by the cluster is produced. Upon outputting an article cluster, we output the words in the title, *without* stemming, excluding stopwords, and the year in which the article was published. The purpose of this is to include in the visualisation a tag cloud for each article cluster, reflecting the relative frequencies of the title terms as well as the distribution of these mentions over time.

4.5 Visualisation

Our visualisation framework aims to aid the understanding of the discovered clusters on two levels: *structural* and *semantic*. While the former is concerned with how the cluster is connected, the latter highlights aspects such as vertex importance and recency; at the semantic level, moreover, visualisations for author and article clusters differ from each other. The three visualisation types are based on a *spring-force layout algorithm* for drawing graphs, which relies on physics; repulsive forces are generated for non-adjacent or weakly tied vertices and attractive forces are generated for the opposite case [17]. Our implementation of the spring layout algorithm is based on the code published by Bader[6]. Our modifications include weighted graph processing, tag cloud generation, and colour-gradient generation. Fig. 3 shows visualisations at the structural and semantic levels for two related clusters. Because visualisations at the structural level are a straightforward result of applying the layout algorithm, we focus on describing the other two in more detail.

In *author-cluster visualisations*, importance, recency, and tightness in relationships are respectively represented with vertex (circle) size, vertex colour brightness, and edge length and width. Figure 3b illustrates some examples (actual clusters; it can be observed that words such as “algorithm” and “problem” are uninformative and should be pruned in the preprocessing phase).

[6] Available at `http://www.mathiasbader.de/studium/bioinformatics/`.



(a) (b) (c)

Fig. 3: Examples of cluster visualisations. In (a), the seed of the cluster is shown in black. For (b) and (c), colours depict recency (the darker the newer).

Vertex size is determined using PageRank [6]; the drawn circle is proportional to the obtained score. Colour brightness, on the other hand, is determined by categorising authors as *active* (fair brightness), *inactive* (dull colour), or *new* (bright colour) with a set of rules. These rules are based on the recency of the author's first and last articles; an article, in turn, is considered as recent when the difference between its publication year and a base year surpasses a given threshold. The base year may be the current year or the last year recorded in the bibliographic repository if treating with a static snapshot. Edge length and width are, finally, determined with the layout algorithm. Highly related vertices appear, in consequence, close to each other.

The *article-cluster visualisation* consists of a *semantic tag cloud* similar in spirit to Clark's Word Clouds[7]. We considered the use of tag clouds for their rapidly increasing popularity in Web communities [5] and their intrinsic nature for intuitively describing topics. To create a semantic tag cloud (or "topic cloud"), article titles are broken down into sets of keywords, and the n most frequent words are chosen (we consider $n = 30$ to be suitable value for this context); note that frequency is proportional to font size in tag clouds. The graph cluster is then converted into a word *co-occurrence graph* in which a pair of words shares an edge (co-occurs) if these are found on the same article title or on the titles of adjacent articles in the graph cluster. The layout algorithm thus produces a cloud in which highly co-occurring words are close to each other — a placement that is different from the alphabetic or aesthetic[8] ones conventionally used.

A colour fringe is created for each word by obtaining the publication years of the related articles, sorting the years in non-decreasing order, and assigning a part of the fringe to each year; a colour intensity is given to each part according to the recency of the year (the recency is calculated using a base year). A fringe with a solid bright colour indicates that the concept represented

[7] Available at `http://neoformix.com/2008/ClusteredWordClouds.html`.

[8] For example, the Wordle tool (`http://www.wordle.net`).

Table 2: Properties of the bipartite graph generated from the DBLP repository.

	Vertices (n)	Edges (m)	Density (δ)
Bipartite graph	1 million	1.3 million	0.3×10^{-5}
Article projection	$\approx$ 560,000	8.8 million	5.6×10^{-5}
Author projection	$\approx$ 450,000	1.2 million	1.2×10^{-5}

by a keyword has been recently used (new), whereas a fringe with a solid opaque or greyish colour indicates the opposite (old or not active); a fringe with a colour gradient (ranging from opaque to bright) indicates that a concept has been used throughout time (active).

5 Experimental evaluation

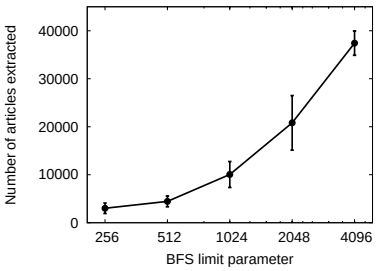
In order to evaluate the quality of the results and the computational feasibility of the proposed approach, we present visualisations and statistics to demonstrate functionality and performance. In particular, we discuss a case study on scientific collaboration graphs that was done with the DBLP bibliography repository[9], which is a well-known collection of valuable computer science publications, such as articles, thesis, proceedings, and books. Our extract of the DBLP bibliography repository focused on *article* publications, obtained from the available XML file (2010 version). For each article, author names, title, journal, and year of publication were collected. From the bipartite graph obtained, the corresponding author and article projections were calculated as weighted graphs using Eq. (7) to compute edge weights; Table 2 briefly characterises these graphs.

Subgraphs of different orders were then obtained via breadth-first search (BFS) graph traversal starting at a random seed vertex, $S = \{v\}$ and then replacing S by $S \cup \{\cup_{w\in S}\Gamma(w)\}$, adding the vertices one at a time, (that is, including direct neighbours of the previously included vertices), repeating this until the desired order is reached. The vertex set thusly obtained is then used to induce a subgraph; this is done to preserve local graph structure as much as possible within the sampled subgraph. Hence, if the BFS parameter $\beta = k$ and the random seed vertex is $v \in S$, then for all vertices $w \in S$

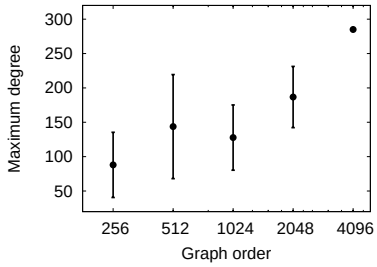
$$d_{v,w} \leq k. \tag{10}$$

The BFS was performed on one of the projections and then extended to the other side of the bipartite graph as follows: when using BFS on articles, we included all the authors of the articles in S and all edges induced by their inclusion, and vice versa. As shown in Fig. 4a, the number of articles extracted scales more or less linearly with the BFS parameter β (note that the horizontal axis has logarithmic scale). Fig. 4b shows that the maximum vertex degree in the preprocessed graphs behaves slightly sub-linearly to the BFS parameter.

[9] Available at `http://dblp.uni-trier.de/` in XML format.

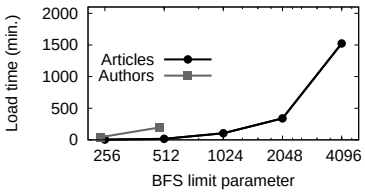


(a) The average and std.dev. of the number of articles in the subgraphs using the number of authors as a BFS limit β.

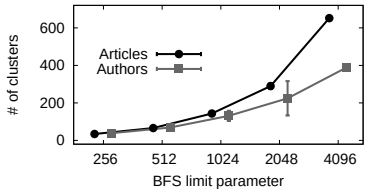


(b) The average and standard deviation of the maximum vertex degree of the subgraphs in terms of the BFS limit β.

Fig. 4: Degrees according to BFS limit parameters.



(a) Preprocessing time



(b) Clusters obtained

Fig. 5: Preprocessing time and the number of stored clusters.

The *clustering* begins with loading the preprocessed graph; the computational effort required is (sub-)linearly proportional to the order of the bipartite graph, as shown in Fig. 5a (horizontal axis is drawn with logarithmic scale). The figure shows the load time of the preprocessed data for each of the BFS-extracted data sets. The BFS runs parametrised by authors were only analysed for the lowest parameter value (i.e., 256) due to the order explosion introduced by including hub authors. We only stored clusters above an order threshold, set to four in our experiments; all the following results are computed on those clusters. Fig. 5b shows the number of article and author clusters, average and standard deviation, over the instances extracted by article-limited BFS. The computational effort in the clustering itself is also roughly linearly proportional to the number of vertices present in the database extract, as is shown in Fig. 6. Again the author-based projections turned out to be impractically slow due to the presence of author hubs.

We plot the time of computation against the number of vertices in the projection of the seed vertex, in order to explore a relationship with the graph order and the computation time; Fig. 7a illustrates this, averaged over each individual projection order. Article clusters show no clear dependence of the projection order in their computation times, whereas the computation time of the author clusters reflects a quadratic dependence (note that the horizon-

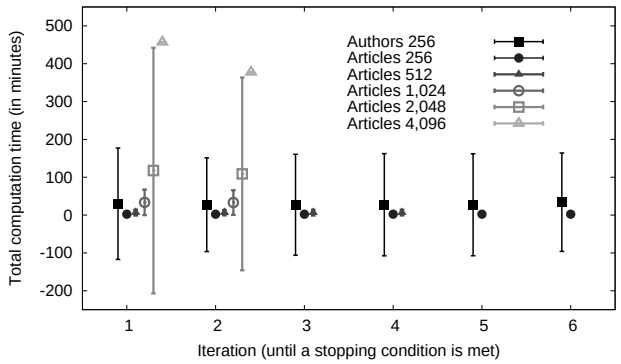
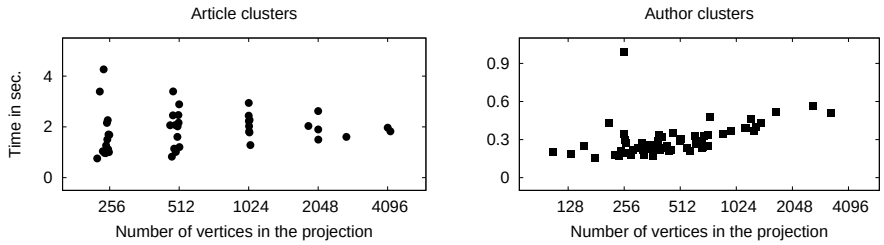
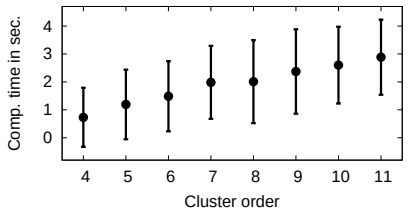


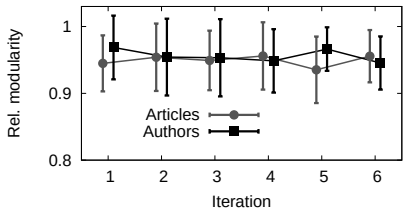
Fig. 6: The computation time for the clustering of the entire input graph (computing a cluster for each vertex in each iteration, asymmetrically) in minutes, average and standard deviation shown, for the clustered data sets.



(a) Time vs. graph order for the two projections



(b) Time vs. cluster order over all clusters



(c) Relative modularity

Fig. 7: Performance and quality measures for the obtained clustering.

tal axis has logarithmic scale). The computational effort to compute a single cluster depends directly on the order of the resulting cluster, as can be seen in Fig. 7b that plots the time in seconds that it took to compute a single cluster versus the order of the resulting cluster.

Modularity is a popular measure for the presence of a community structure [37], shown to emerge among other situations when a network attempts to synchronise [1]. It can be computed by calculating the fraction of inter-cluster

edges of all the edges in the graph and then subtracting the fraction of edges that would fall within the clusters if the same amount of edges were placed uniformly at random among the vertices of the graph. We examine the stability of our produced clusters as the algorithm advances by computing the modularity at each iteration and then normalising this by the maximum value achieved. The results are shown in Fig. 7c and there is no significant variation; typically a total of six iterations were executed and the maximum modularity was reached mostly at the fifth iteration, although some peaked on the first iteration.

We searched existing literature for a similar method in order to compare our results with other approaches and the closest match was LP&BRIM [2, 25], for which an implementation is provided at `https://github.com/tpoisot/biweb`. It works with bipartite input graphs, expressed as adjacency matrices (non-square, as the columns correspond to one side of the bipartite graph and the rows to the other), so we first converted our BFS-based database extracts into the specified input format. The file size grew over 21 times as large as the raw file on average (the median being five times the original, some requiring several gigabytes). Upon observing this, we switched to a more powerful computer with double the RAM and a faster processor, as the one used to run our own method would take too long with these files.

For $\beta \geq 1,024$ a single execution of LP&BRIM on an author-based BFS extract with $\beta = 1,024$ took nearly six hours and with $\beta = 512$ a half an hour to complete the global clustering for a single run of LP&BRIM using 30 iterations, which unfortunately produced rather different clusterings on every execution. Hence we only experiment on the input data for $\beta = 256$, only on the article-originated database extracts (in order to avoid the very large graphs resulting from hub authors). We allowed 500 iterations to LP&BRIM to produce a more stable clustering (as in the example run provided in the code repository). We could not work on all 32 of the inputs of this group, as we experienced an issue with the implementation from `https://github.com/tpoisot/biweb` that in some input graphs created clusters for "phantom vertices" that were not present in the bipartite input given to the algorithm and did not appear in the vertex list reported by the implementation, although no error was reported by the implementation at any time. We discarded all those as we cannot compare clusters when there are unknown additional vertices present; we attempted to circumvent the issue running the program multiple times and with different iteration counts with little luck.

The structural comparison is not entirely straightforward as our clusters are composed of either authors or articles, whereas theirs may freely combine the two types of vertices. Also, our clusterings are asymmetric whereas theirs are not. Hence we examined the *agreement* between the two methods in terms of whether or not two vertices are placed in the same cluster together. It is important to recall that LP&BRIM produces a global clustering of the entire bipartite graph, consisting of both author and article vertices, whereas our asymmetric method produces a cluster for each seed vertex and only among the vertices of the same kind (author or article).

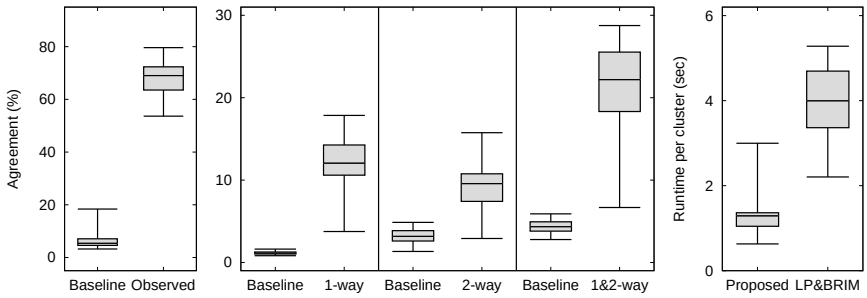


Fig. 8: Box-whiskers plots of the comparative study: agreement of LP&BRIM to our proposed method (left and centre) in terms versus the agreement of our proposed method to the existing method of LP&BRIM together with the runtime per cluster in seconds for the two methods (right).

First, for each pair of vertices that are placed in the same cluster in our clustering, we check whether or not they are placed in a cluster together also by LP&BRIM. Figure 8 shows a box-whiskers plot indicating the minimum and maximum values (whiskers) together with the quantiles of 25 and 75 % (box) and the median (the line inside the box).

Then, to carry out a comparison vice versa, we counted for each pair of vertices $\{v, w\}$ of the same type (author or article) that was clustered together by LP&BRIM whether also our method included that relation in the resulting clustering. Note that there are three possible outcomes: either v belongs to the cluster of w while also w includes v in its cluster (a two-way match), only one of them considers the other a member of their cluster (one-way match), or neither considers the other as a cluster member (no match).

Figure 8 reports the results of this comparison as well as boxplots of the runtimes per produced cluster of the two methods for the 15 input graphs that presented no phantom vertices under LP&BRIM (we normalise the time with the number of clusters as our method is asymmetric and uses each vertex as seed once, whereas theirs is global and hence computes only one set of clusters). The baselines in Figure 8 are computed by creating partitions at random with the cluster orders of the two methods for each input graph and computing the agreement among those, and averaged over 100 repetitions. The process as such produces a baseline for the comparison where we examine how many of the pairs that were grouped together in our clustering were also groups together in the clustering of LP&BRIM, whereas when comparing an asymmetric clustering to a global one, the latter is still a partition while the former are subsets with possibly repeated content.

The level of agreement is clearly superior to that of a random assignment and indicates the two methods produce clusterings that are related to one another. In particular, as most of the vertices that were clustered together by our method were also clustered together by LP&BRIM but our method

clustered together less frequently those that were put in the same cluster by LP&BRIM. This indicates that our clusters seem to be subsets of theirs, as can be expected, as our clusters are asymmetric whereas their clustering is symmetrical, and also as our clusters are formed by either authors or articles and their clusters combine the two.

On average the computed clusterings of LP&BRIM have modularity around 0.7, computed as a special case redefined for bipartite graphs as the edges connecting vertices on the same side of the graph cannot exist; our method does not require this as the produced clusters are not bipartite. As our clustering is not a global one, we cannot compute a modularity value as such that could be compared to that of LP&BRIM, even if the two graphs admitted the same formulation of modularity (their clusters reside in a bipartite graph, encompassing vertices from both sides, whereas ours consist in vertices on either one side or the other, with the projected edge set, and hence the two graphs are quite different even though they represent the same data set). We hope to find an implementation for a method that better corresponds to ours to be able to carry out a more detailed comparison as future work; a promising candidate is a recent work of Ma et al [27].

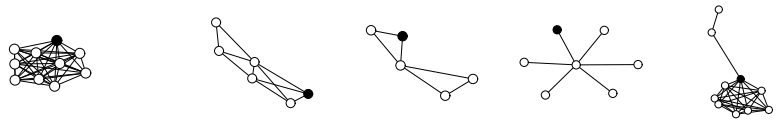
5.1 Qualitative visualisations

In this section we provide several cluster visualisations that depict the output of the proposed algorithm; we analyse these visualisations to show and discuss outstanding patterns in the discovered clusters. Results are presented for three types of analysis: topological, iterational, and textual.

5.1.1 Cluster topology analysis

A sample of $1,000$ clusters of different types, projections, and collection sizes was randomly chosen by means of systematic sampling[10]; visualisations at the structural level were generated for these clusters (see Section 4.5). The inspection of these visualisations led to the identification of different structures or *topologies* with varying degrees of robustness and transitivity; examples of these topologies are presented in Fig. 9. The first category of the topologies concerns *cliques* (Figs. 9a and 9e). These are well-known "closed" structures that typically represent communities; not surprisingly, the majority of the clusters in the sample fall into this category ($\approx 80\%$). Because cliques are maximal complete subgraphs by definition, all members of the cluster are adjacent to each other and all relationships are transitive; consequently, if one member is eliminated, the rest still relate to each other and the group continues to be strong. In a scientific collaboration context, an author clique represents a well-formed research group where individuals have all collaborated with one another. Article cliques, conversely, represent papers which all share authors

[10] In a systematic sample, each element is chosen after k steps, where k results from dividing the total number of elements by the desired sample size.



(a) Clique (b) Multicluster (c) Bottleneck (d) Star (e) Tail

Fig. 9: Cluster topologies.

with one another. Also, a high presence of *triangles* (the smallest cliques possible) was noticeable in the sample (35% for authors, 15% for articles); as the minimum accepted cluster order was four, these triangles had "tails" attached.

The second topological category consists of structures in which two or more cliques are bond together, e.g. two triangles (see Fig. 9b for another example); we refer to this structure as "multiclique" or "multicluster". Multiclusters could be seen as separate communities with several vertices in common or even as overlapping clusters. Regarding robustness, if the borderline vertices are eliminated, the cluster gets disconnected; relationships are transitive within each individual clique, but non-borderline vertices of different cliques clearly have no transitive relationships among them. For author clusters, these non-borderline vertices stand for authors who have not collaborated with each other; for article clusters, they stand for papers that have no authors in common. Semantically, author multiclusters could represent two or more groups that share collaborators, groups with fissions, or groups where some members work on several research areas at the same time. Multiclusters with three or more cliques potentially represent complex research groups and the topics derived from such groups.

Our third topological category is given by "bottlenecks", i.e. multiclusters that have only one borderline vertex (see Figs. 9c); if this vertex were removed, the cluster would be split into two components. Because bottlenecks are a special case of multiclusters, the properties and semantics of the latter can still be applied. Bottlenecks can be considered, nevertheless, as weaker groups.

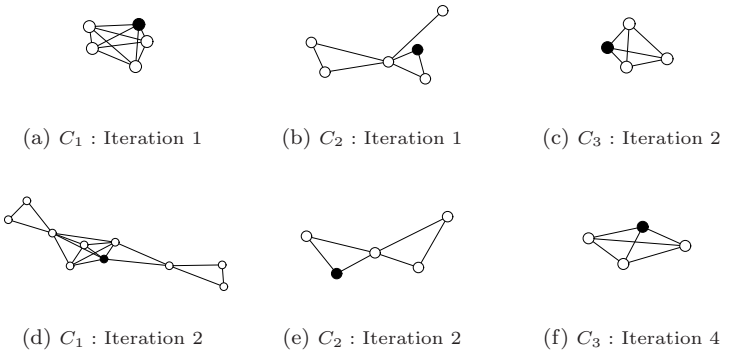
The last category is given by *stars* (Figure 9d). These are clusters held by a single vertex; if this were removed, the rest of the vertices would become isolated. Consequently, stars are the least robust structures, and have no transitivity. In a scientific collaboration context, the research group exists because of an individual interacting with the rest; e.g. consider a group where the thesis adviser is the centre of the star and the others are his students. For article clusters, the centre of the star is a paper that shares authors with the rest, but these others have no common authors; seen from another point of view, the centre of the star contains at least one author from each of the rest of the papers. It is interesting to note that stars are more present in author clusters (2% vs. 15% in the sample); this could be due to the social nature of author projections.

In general, cliques were by far the most frequent category in the sample. Similar results hold when organising the sample by projection or collection size.

Let us note that the subdivision into topologies is not only a finding by itself but also allows to generate a coarser level of granularity for other analyses. Approximately 30% of the clusters had *tails* or "satellites" that seem to hang from them, i.e. vertices connected to the cluster by a single edge. Such vertices could either be adjacent to the cluster or lie in a path that leads to it (tails of tails). In the case of stars, we only considered as tails the latter vertices. Tails usually originate when their presence increases the fitness value of the cluster or when seeds are kept. For author clusters, tails represent authors who — directly or indirectly — collaborate with one member of the research group. This case could be given by a new author who begins to integrate with the group or is not very gregarious, or an external collaborator of the group. For article clusters, the interpretation is symmetrical: tails are papers that have been coauthored by a member of the group and an external collaborator or by external collaborators (tails of tails). With respect to the sample, 20% of the article clusters contained tails, while this percentage doubled in author clusters; greatest tail length, in addition, was of three and was found within the author clusters as well. This behaviour could be due to the proper dynamics of a social group.

To analyse how groups change throughout the asymmetric clustering procedure, a systematic sample of 200 cluster iterations was gathered (≈ 800 clusters in total). Fig. 10 shows several examples of these cluster iterations. The first analysed aspect was topology shifting; this provides a perspective of how drastic the changes in the clusters are — both between iterations and between the first and last iterations only. Regarding the former, around 35% of the clusters changed from one topology category to another (e.g. star to clique) in some iteration; regarding the latter, approximately 30% of the clusters ended with a category that was different from the initial. While both article and author clusters showed similar percentages in these changes, author clusters slightly tended to shift to less robust categories in the final iterations (e.g. clique to star, clique to bottleneck, bottleneck to star, etc.). Article clusters presented the opposite behaviour. Interestingly, there were clusters that consistently preserved their size, order, and topology throughout iterations[11]; we refer to this phenomenon as the presence of a *dominant structure*. While the percentage of these structures is low in article cluster iterations (5%), author cluster iterations present more dominant structures (20%).

[11] Only iterations where the cluster order was above the threshold were considered.



(a) C_1 : Iteration 1 (b) C_2 : Iteration 1 (c) C_3 : Iteration 2

(d) C_1 : Iteration 2 (e) C_2 : Iteration 2 (f) C_3 : Iteration 4

Fig. 10: Iterations for different clusters, which are labelled as C_1, C_2, and C_3 for simplicity (seeds coloured in black). The left cluster depicts significant changes, while the middle one shows mild changes (note the addition and removal of tails), and the cluster on the right maintains structure.

5.1.2 Textual analysis

To assess the thematic cohesion of article clusters, a rough analysis was carried out over the titles of a small sample containing 60 clusters — 30 cliques and 30 stars. For equity, these samples included clusters with four or five elements only (the average title length was of six words in both cases). We considered a cluster as cohesive according to repeated words (articles talk about the same or similar topics). Stemming and lemmatisation were carried out on the text to group those words with the same root or base.

Results were considerably similar with both samples. From the clusters, 84% contained repeated words (80% for stars, 87% for cliques). The maximum word frequency in both cases was of four (all titles contain the word); three documents (two for cliques and one for stars) achieved this frequency. The average word frequency was of 1.08 for stars and 1.14 for cliques. While cliques, in general, seemed to be more cohesive than stars, the differences are still not highly significant. In that sense, it seems appropriate to recall that every article cluster represents a *research topic*. The cohesion of these clusters is, therefore, primarily given by the sharing of common authors and not a common thematic per sé; the latter — more than holding the clusters together — is a result of the research groups working on the same knowledge area. In that sense, a less robust cluster is not necessarily a less cohesive cluster in terms of topicality. Since other patterns related to text were not detected in the present work, a more comprehensive analysis on this aspect is left for future work.

6 Conclusions

We have proposed an algorithm for clustering scientific collaboration networks. The algorithm is based on local graph clustering that functions on both sides of a bipartite input graph in an asymmetric fashion, strengthening or weakening connections according to whether vertices are found together on the same or different clusters. This permits to maximise cohesion by the the gradual refinement of the resulting clusters. We also presented visualisations for the obtained clusters.

As future work, we consider improvements on the clustering algorithm, including the introduction of another objective function to assess if an output should actually be considered as a cluster (community) or not. This would be specially useful in scenarios where there are no natural clusters. The algorithm associated to this objective function could be supervised (i.e., perform a supervised classification). We plan to provide an actual online implementation with relaxed recalculation of clusters (upon the insertion of new related data) that relies on a database of the graph structure; presently the graph is loaded entirely in main memory that severely limits the scalability of the current implementation. We plan to study the convergence with such an implementation as the computation times can be reduced. We are also curious of analysing articles through a tripartite graph, where the publication forum (journal, congress proceeding series, etc.) is a third set of vertices. We also wish to introduce auto-adaptive parameter adjustment and stopping conditions for higher robustness.

Acknowledgements The first author was supported by SEP-PROMEP grant number 103.5/12/7884. We thank the anonymous reviewers for their useful suggestions that helped improve the manuscript.

References

1. Avalos-Gaytán V, Almendral JA, Papo D, Schaeffer SE, Boccaletti S (2012) Assortative and modular networks are shaped by adaptive synchronization processes. PRE 86(1):015,101(R)
2. Barber MJ (2007) Modularity and community detection in bipartite networks. Physical Review E 76(6):066,102
3. Batagelj V (2003) Efficient algorithms for citation network analysis. Tech. Rep. cs/0309023, arXiv.org
4. Bian J, Xie M, Hudson TJ, Eswaran H, Brochhausen M, Hanna J, Hogan WR (2014) Collaborationviz: Interactive visual exploration of biomedical research collaboration networks. PloS one 9(11):e111,928, 0
5. Bogárdi-Mészöly Á, Rövid A, Ishikawa H (2013) Topic recommendation from tag clouds. Bulletin of Netw, Comp, Sys, and Software 2(1):pp–25
6. Brin S, Page L (1998) The anatomy of a large-scale hypertextual Web search engine. Computer networks and ISDN systems 30(1-7):107–117

7. Catanzaro M, Caldarelli G, Pietronero L (2004) Assortative model for social networks. PRE 70(3)
8. Catanzaro M, Caldarelli G, Pietronero L (2004) Social network growth with assortative mixing. Phys A 338(1–2):119–124
9. Clement R, Sharp D (2003) Ngram and Bayesian classification of documents for topic and authorship. Literary and Linguistic Computing 18(4):423–447
10. Diestel R (2010) Graph Theory, GTM, vol 173, 4th edn. Springer
11. Ding Y, Yan E, Frazho A, Caverlee J (2009) PageRank for ranking authors in co-citation networks. JASIST 60(11):2229–2243
12. Dorogovtsev S, Mendes J (2002) Evolution of Networks: From Biological Nets to the Internet and WWW. Clarendon Press, Oxford, UK
13. Du N, Wu B, Pei X, Wang B, Xu L (2007) Community detection in large-scale social networks. In: Proc. of WebKDD & SNA-KDD, ACM, New York, NY, USA, pp 16–25
14. da F Costa L, Rodrigues F, Travieso G, Boas P (2007) Characterization of complex networks: A survey of measurements. Adv in Phys 56(1):167–242
15. Flake G, Lawrence S, Giles C (2000) Efficient identification of web communities. In: Proc. of KDD, ACM New York, NY, USA, pp 150–160
16. Fortunato S (2010) Community detection in graphs. Phys Rep 486:75–174
17. Fruchterman T, Reingold E (1991) Graph drawing by force-directed placement. Software: Practice and experience 21(11):1129–1164
18. Gleiser PM, Danon L (2003) Community structure in jazz. Adv in Complex Sys 6(4):563–573
19. Huang J, Zhuang Z, Li J, Giles CL (2008) Collaboration over time: Characterizing and modeling network evolution. In: Proc. of WSDM, ACM, New York, NY, USA, pp 107–116
20. Jeong H, Néda Z, Barabási A (2003) Measuring preferential attachment in evolving networks. Europhys Lett 61
21. Kirkpatrick S, Gelatt Jr CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
22. Larremore DB, Clauset A, Jacobs AZ (2014) Efficiently inferring community structure in bipartite networks. CoRR abs/1403.2933:012,805, URL `http://arxiv.org/abs/1403.2933`, 9
23. Li M, Fan Y, Chen J, Gao L, Di Z, Wu J (2005) Weighted networks of scientific communication: The measurement and topological role of weight. Phys A 350(2–4):643–656
24. Liu J, Li Y, Ruan Z, Fu G, Chen X, Sadiq R, Deng Y (2015) A new method to construct co-author networks. Physica A: Statistical Mechanics and its Applications 419:29–39
25. Liu X, Murata T (2009) Community detection in large-scale bipartite networks. In: Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on, IET, vol 1, pp 50–57
26. Liu X, Bollen J, Nelson M, Van de Sompel H (2005) Co-authorship networks in the digital library research community. Inf Proc & Mgmt

41(6):1462–1480
27. Ma T, Rong H, Ying C, Tian Y, Al-Dhelaan A, Al-Rodhaan M (2015) Detect structural-connected communities based on bschef in c-dblp. Concurrency and Computation: Practice and Experience DOI 10.1002/cpe.3437
28. Milgram S (1967) The small world problem. Psych Today 2:60–67
29. Moody J (2004) The structure of a social science collaboration network: Disciplinary cohesion from 1963 to 1999. Am Sociol Rev 69(2):213–238
30. Newman M (2001) Clustering and preferential attachment in growing networks. PRE 64(2)
31. Newman M (2001) Scientific collaboration networks. I. Network construction and fundamental results. PRE 64:016,131
32. Newman M (2001) Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. PRE 64
33. Newman M (2001) The structure of scientific collaboration networks. PNAS 98(2)
34. Newman M (2002) Assortative mixing in networks. PRL 89
35. Newman M (2004) Coauthorship networks and patterns of scientific collaboration. PNAS 101(Suppl. 1)
36. Newman M (2004) Who is the best connected scientist? A study of scientific coauthorship networks. Complex Networks 650:337–370
37. Newman M (2006) Modularity and community structure in networks. PNAS 103(23)
38. Newman M (2010) Networks: An introduction. Oxford University Press
39. Papadopoulos S, Kompatsiaris Y, Vakali A, Spyridonos P (2012) Community detection in social media. Data Mining and Knowledge Discovery 24(3):515–554
40. Perianes-Rodríguez A, Olmeda-Gmez C, Moya-Anegn F (2010) Detecting, identifying and visualizing research groups in co-authorship networks. Scientometrics 82(2):307–319, DOI 10.1007/s11192-009-0040-z, URL `http://dx.doi.org/10.1007/s11192-009-0040-z`
41. Porter M (1980) An algorithm for suffix stripping. Program 14(3):130–137
42. Radicchi F, Castellano C, Cecconi F, Loreto V, Parisi D (2004) Defining and identifying communities in networks. PNAS 101(9):2658–2663
43. Ramasco J, Dorogovtsev S, Pastor-Satorras R (2004) Self-organization of collaboration networks. PRE 70(3):036,106
44. Schaeffer S (2007) Graph clustering. CoSRev 1(1):27–64
45. Schaeffer SE (2005) Stochastic local clustering for massive graphs. In: Advances in knowledge discovery and data mining, Springer, pp 354–360
46. Sozio M, Gionis A (2010) The community-search problem and how to plan a successful cocktail party. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, pp 939–948
47. Stamatatos E (2009) A survey of modern authorship attribution methods. Journal of the American Society for Information Science and Technology 60(3):538–556

48. Tang J, Zhang J, Yao L, Li J, Zhang L, Su Z (2008) Arnetminer: extraction and mining of academic social networks. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 990–998
49. Tran DH, Takeda H, Kurakawa K, Tran MT (2012) Combining topic model and co-author network for KAKEN and DBLP linking. In: Intelligent Information and Database Systems, Lecture Notes in Computer Science, vol 7198, Springer, pp 396–404
50. Yang T, Jun R, Chi Y, Zhu S (2009) Combining link and content for community detection: A discriminative approach. In: Proc. of KDD, ACM, New York, NY, USA, pp 927–936
51. Ye Q, Wu B, Wang B (2008) Visual analysis of a co-authorship network and its underlying structure. In: Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on, vol 4, pp 689–693, DOI 10.1109/FSKD.2008.436
52. Zhou S, Cox I, Hansen LK (2009) Second-order assortative mixing in social networks. Tech. Rep. 0903.0687, arXiv.org

Technical Biographies

Sara Elena Garza Villarreal earned a Ph.D. in Information Technologies and Communications with a minor in Intelligent Systems at the Instituto Tecnológico y de Estudios Superiores de Monterrey, Mexico, in 2010. She works as a full-time professor and researcher at Universidad Autónoma de Nuevo León (Mexico). Her primary research interests include graph clustering, topic mining, complex network analysis, data mining, and artificial intelligence in general.

Satu Elisa Schaeffer studied her Master's and Doctorate in Computer Science and Engineering at Helsinki University of Technology (now Aalto University), in Finland, from 1996 to 2006. She is Associate Professor at Universidad Autónoma de Nuevo León in Mexico since 2006. Her field of study was Theoretical Computer Science and her primary research interests include modelling, analysis, and structural characterisation of complex systems.