Weighted Linking Decomposition: Mining Denser and More Compact Hierarchies for Bipartite Graphs

Edré Moreira, Guilherme Oliveira Campos, Wagner Meira Jr.

Universidade Federal de Minas Gerais, Brasil
{edre, gocampos, meira}@dcc.ufmg.br

Abstract. Dense subgraph detection is a well-known problem in graph theory. The hierarchical organization of graphs as dense subgraphs, however, goes beyond simple clustering, as it allows the analysis of the network at different scales. Although there are several hierarchical decomposition methods for unipartite graphs, only a few approaches for the bipartite case have been proposed. In this work, we explore the problem of hierarchical decomposition for bipartite graphs. We propose an algorithm called Weighted Linking that identifies denser and more compact hierarchies than the state of the art approach. We also propose a new score to help choose the best between two hierarchical decompositions of the same graph. The proposed algorithm was evaluated experimentally using six real-world datasets and identified smaller and denser hierarchies on most of them.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications

Keywords: bipartite graphs, dense subgraphs, graph mining, hierarchical decomposition

1. INTRODUCTION

Dense subgraphs identification is a well-studied subject in graph theory [Lee et al. 2010]. However, organizing a graph as a hierarchy of dense subgraphs goes beyond simple clustering, as it allows for network analysis at various scales. Hierarchical decomposition of graphs applies to various scenarios, such as data visualization [Abello and Korn 2002], [Alvarez-hamelin et al. 2006], anomaly detection [Shin et al. 2016], dense subgraphs discovery [Andersen and Chellapilla 2009], community detection [Giatsidis et al. 2011], and influential spreaders identification [Kitsak et al. 2010], [Al-garadi et al. 2017], among others. Let us consider, for instance, the author-paper bipartite graph, where authors who work together more consistently, representing research groups, are located at the hierarchy's deepest levels. As we move up to ancestor levels, we find authors who bridge different groups or collaborate less frequently.

Given a plain graph $G = (V, E)$, with node set V and edge set E, and a subgraph $G' = (V', E')$, where $G' \subseteq G$, there are two popular density measurements in the literature, namely *mean degree density* and *edge density*. *Mean degree density* (d_{deg}) is the number of edges divided by the number of nodes. *Edge density* (d_{edge}), on the other hand, is the number of existing edges divided by the number of the edges of a clique [1] in the same node set. Since a trivial subgraph with two nodes and one edge presents the maximum density by d_{edge}, and it is known to find the maximum clique in a graph is an NP problem [Tsourakakis et al. 2013], the authors work with approximated algorithms for this problem [Lee et al. 2010]. When considering density as d_{deg}, however, there exists a polynomial-time

[1]In graph theory, a clique is a graph where there is an edge connecting every two nodes

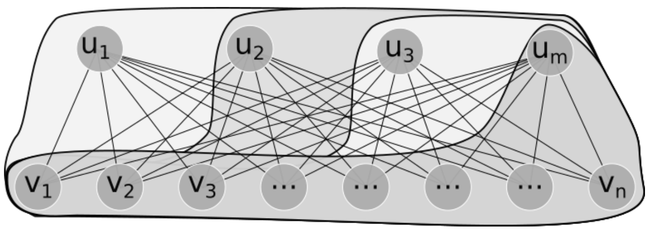


Fig. 1. A possible hierarchical decomposition of a complete bipartite graph. The hatched areas represent subgraphs. Each subgraph in the hierarchy adds a single node from the partition U.

solution for the problem [Goldberg 1984].

In order to illustrate the challenge associated with the hierarchical decomposition of a bipartite graph, let us consider such decomposition of a complete bipartite graph $H_c = (U, V, E)$, shown in Fig. 1, with node set $U \cup V$ and edge set E, $|U| = m$, $|V| = n$ in m subgraphs such that the outermost subgraph contains all nodes in $U \cup V$, the subgraph one level down contains $m-1$ nodes from U and all nodes from V, and so forth. Since all subgraphs have $d_{edge} = 1$, the sum of all subgraphs densities will be large, as well as the height of the hierarchy, which is not desirable.

In this work, we address the problem of decomposing a plain, undirected, and unweighted bipartite graph as a hierarchy of dense subgraphs. Our objective is to determine a hierarchical decomposition that generates denser subgraphs with less hierarchical levels when compared against the state of the art approach [Sariyüce and Pinar 2018]. Furthermore, we need to assess the quality of the generated hierarchies. Since, as far as we know, there is no work dedicated to this problem, we also propose a score to identify the best hierarchical decomposition of a bipartite graph. We use the terms graph and network throughout the text with the same meaning, as well as edge and link.

Our contributions in this work may be summarized as follows:

—**Algorithm for the hierarchical decomposition of bipartite graphs:** we propose the algorithm *Weighted Linking* for the hierarchical decomposition of bipartite graphs that allows the definition of the desired trade-off between hierarchy height and subgraph density.
—**Score to evaluate hierarchical decompositions**: we propose a new score to help choose the best between two hierarchical decompositions for the same bipartite graph.

We extend in this paper a preliminary conference publication [Moreira et al. 2019], in particular, with respect to:

—We improve the discussion in the related work section, covering a broader scope of studies.
—We add the pseudocode of our approach and discuss it.
—We propose a new score to compare the quality of two hierarchical decompositions.
—We include, in the experiments, one small and one large dataset in terms of both the number of nodes and the number of edges.
—We improve the qualitative analysis of the DBLP case study.
—We perform a qualitative analysis of the recently added datasets.
—We improve the discussion in the conclusion and future work sections.

This paper is organized as follows. In Section 2, we present related work to dense subgraph detection and hierarchical decomposition of graphs. In Section 3, we define the problem. In Section 4, we provide the intuition about our algorithm and the decomposition algorithm itself. In Section 5, we present both quantitative and qualitative analyses obtained experimentally. Finally, in Section 6, we present our conclusions and future work.

2. RELATED WORK

This section discusses the most relevant related work, particularly on graph core decomposition and dense subgraphs detection.

2.1 Core decomposition

[Seidman 1983] proposes an approach to determine the network cohesion based on a minimum degree that also produces a hierarchy of increasing cohesion. The author introduces the *k-core* of a plain unipartite graph $G = (V, E)$, $|V| = n$, which is the maximal subgraph $G' \subseteq G$, $G' = (V_{G'}, E_{G'}) \mid \forall v \in V_{G'} : deg_{G'}(v) \geq k$, where $deg_{G'}(v)$ is the degree of node v in G'. The *core number* of a node is the largest value k of a *k-core* containing v. The *k-tip* [Sariyüce and Pinar 2018] concept used in our work is similar to the *k-core* definition.

Core decomposition has been extended to scenarios beyond unipartite graphs, such as uncertain graphs [Bonchi et al. 2014], temporal graphs [Wu et al. 2015], directed graphs [Giatsidis et al. 2013], bipartite graphs [Batagelj and Zaversnik 2002], graph streams [Sarıyüce et al. 2016], and multilayer graphs [Galimberti et al. 2017], among others [Malliaros et al. 2019]. Applications of hierarchical decomposition of graphs include data visualization [Abello and Korn 2002], [Alvarez-hamelin et al. 2006], anomaly detection [Shin et al. 2016], dense subgraphs discovery [Andersen and Chellapilla 2009], community detection [Giatsidis et al. 2011], and influential spreaders identification [Kitsak et al. 2010], [Al-garadi et al. 2017], among others [Malliaros et al. 2019]. The core decomposition principle is also applied in this work.

[Brown and Feng 2011] propose a *k-core* decomposition on Twitter using a logarithmic evaluation of node degrees. Thus, the number of shells is reduced as well as the execution time. The shells interpretability is also improved. In [Wu et al. 2015], the authors define the *k,h-core* for temporal graphs, as well as an efficient distributed algorithm to compute it. The idea is to consider the number of times the edges appear over time. The *k-core* definition is enforced so that the neighbors in the subgraph of the same *k-core* are connected by at least h temporal edges. However, these algorithms are not directly applicable to the bipartite case since they are designed to work in one mode graph. In our work, we also propose to reduce the number of subgraphs, besides improving the interpretability.

[Ban 2018] proposes a linear complexity algorithm to find a maximal half isolated biclique, a maximal biclique such that at most one partition allows edges to outside nodes. The approach is motivated by the fraud detection task. The key observation is that it is easy for fraudsters to create edges from their account to other nodes, including legitimate nodes, but it is difficult for fraud nodes to gain edges from legitimate users. Despite being the densest subgraphs in a bipartite graph, bicliques are very restrictive since all possible edges between the two partitions have to be present. In our work, we allow for dense subgraphs to be detected even if possible edges are absent.

Influential spreaders in social networks are the research focus of [Al-garadi et al. 2017]. A weighted degree is assigned to each node using the node degree itself and the interaction with neighbors given by the propagation strength and engaging strength. The peeling is made based on the node weighted degree, which is updated as nodes are removed. In this work, we also apply the weighted peeling strategy, although the weight is computed differently.

2.2 Dense subgraphs detection in static graphs

Several methods deal with finding the densest subgraphs taking the density as the mean degree of the subgraph nodes. [Goldberg 1984] proposes an algorithm to compute the exact solution for this problem by computing $O(log n)$ *min-cut*. [Charikar 2000] proposes a greedy strategy to approximate the densest subgraph in $O(n)$. [Khuller and Saha 2009] extend the algorithm *min-cut* to find the densest subgraphs in directed graphs. They also propose a greedy strategy similar to [Charikar

2000]. The authors also prove that the algorithm is a 2-approximation for the densest subgraph problem in directed graphs. In this work, we apply a similar greedy strategy to decompose the bipartite graph.

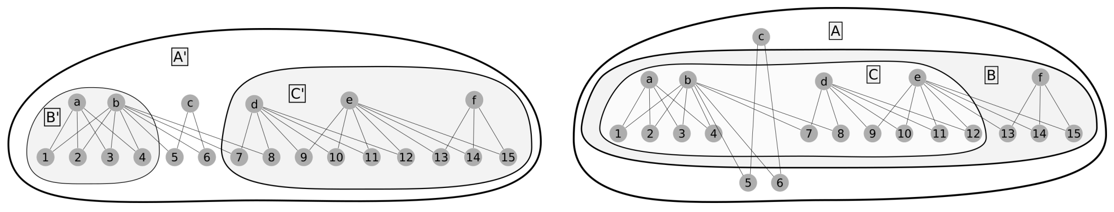
[Tsourakakis et al. 2013] claim that dense subgraphs found by taking density as the mean degree tends to be larger and sparser (low edge density). To overcome this issue, they define an objective function to find the best *α-quasi-clique*, that is, the best node set maximizing the function. They propose a greedy algorithm that extends the one described by [Asahiro et al. 2000]. They also provide a local search heuristic, which is proved to be locally optimal - if any node is added or removed from the subgraph, the value for the objective function is decreased. Although applying the mean degree density at an intermediate step, we propose to find dense hierarchies for bipartite graphs by taking the edge density as a measure. We also focus on balancing the subgraphs densities and sizes, besides dealing with the bipartite case.

The algorithm proposed by [Sariyüce and Pinar 2016] is designed for building the dense subgraphs hierarchy by traversing the graph only once. The algorithm *FastNucleusDecomposition* first sorts the nodes according to their *r-clique (K_r)* [2]. The nodes are then inspected one by one in increasing order of K_r. Let v be the current node and $\lambda(v)$ the maximum *r-clique* value associated to v. The inspection process starts by assigning the current $K_r(v)$ as $\lambda(v)$. The $K_r(u)$ for all v's neighbors u that have not been processed yet (that means $\lambda(u)$ is undefined) are decremented by one unless $K_r(u) = K_r(v)$. Up to this point, this is a generalization of the *k-core* decomposition. The interesting strategy the authors propose is to use the already computed $\lambda(u)$ to build the subgraphs and their parent-child relationship. If $\lambda(v) = \lambda(u)$, they are marked as belonging to the same subgraph. Otherwise, the only possibility is $\lambda(v) > \lambda(u)$, so the subgraph containing u is marked as the parent of the subgraph containing v. The algorithm uses the union-find data structure [Cormen et al. 2001] to facilitate the subgraphs merging operation. At the end of the peeling process, an extra step over the structure is made to build the final hierarchy. Despite building dense subgraphs, the strategy used to join nodes in the same subgraph while building the hierarchy does not consider the connection strength between them.

[Sariyüce and Pinar 2018] describe two algorithms to find dense subgraphs, as well as for building the hierarchy, for the bipartite case, named *Tip Decomposition* and *Wing Decomposition*. For the bipartite graph $H = (U, V, E)$, *Tip Decomposition* starts by computing the butterflies connecting nodes in the partition U. A butterfly is a biclique having exactly two nodes in each partition. For each $u \in U$, the butterflies count for u, $\beta(u)$, is computed. The peeling process is done based on $\beta(u)$. For each u with smaller $\beta(u)$, the *tip number*, $\theta(u)$, is assigned to u with the same value of $\beta(u)$. For all v participating in u's butterflies and for which $\theta(v)$ has not been computed yet, $\beta(v)$ is decremented by the number of butterflies containing both u and v, bounded by $\theta(u)$. The process for *Wing Decomposition* is similar. However, instead of computing butterflies for nodes, butterflies are accounted for edges. The number of butterflies, $\beta(e)$, each edge participates in is computed first. The algorithm then processes each edge $e \in E$ in ascending order of $\beta(e)$. The *wing number*, $\psi(e)$, is assigned to e as the current value of $\beta(e)$. For each butterfly containing e, the three other edges are inspected, and $\beta(.)$ is decremented by one if $\psi(.)$ has not been computed yet, bounded by $\psi(e)$. In either case, the dense subgraphs hierarchy is built using the disjoint-set data structure they propose [Sariyüce and Pinar 2016].

In this work, we propose an extension of [Sariyüce and Pinar 2018] to overcome some issues on building denser subgraphs, as we discuss in Section 4.

[2]An *r-clique*, or K_r, of a node is defined as the number of cliques of size r the node participates. A *2-clique* of a node, for instance, is equivalent to its degree.



(a) h_1 (b) h_2

Fig. 2. Two hierarchical decompositions, h_1 and h_2, for the same bipartite graph. For the subgraphs of h_1, A', B', and C', we have the levels *1, 2,* and *2*, respectively, and the densities *0.33, 1,* and *0.59*. The total height for h_1 is *5*, the total density is *1.92*, the mean height is *1.67*, and the mean density is *0.64*. For the subgraphs of h_2, A, B, and C, we have levels *1, 2,* and *3*, respectively, and the densities *0.33, 0.4,* and *0.5*. The total height for h_2 is *6*, the total density is *1.23*, the mean height is *2*, and the mean density is *0.41*.

3. PROBLEM DEFINITION

Let $H = (U, V, E)$ denote a bipartite graph, where U and V are sets of nodes, $U \cap V = \emptyset$, and $\{E \mid \forall\, e \in E : e = (u, v) \wedge u \in U \wedge v \in V\}$ is the edge set.

Let $H' = (U_{H'}, V_{H'}, E_{H'}) \subseteq H$, denote a subgraph from H. A hierarchical decomposition h of H is composed by k nested subgraphs, $H'_{1..k}$, which satisfy the restriction that $H'_i \subset H'_j \vee H'_j \subset H'_i \vee H'_i \cap H'_j = \emptyset\ \forall\, i, j \in 1..k$.

The level of a subgraph H'_i ($H'_1 = H$) in the hierarchical decomposition of H is given by

$$l(H'_i) = \begin{cases} 1, \textit{ if } i = 1 \\ l(H'_{i-1}) + 1, \textit{if } H'_i \subset H'_{i-1} \wedge \nexists\, H'_t \mid H'_i \subset H'_t \subset H'_{i-1}\ \forall\, t \in 1..k \end{cases}$$

We define the total height of hierarchy h as $A_h = \sum_{i=1}^{k} l(H'_i)$ and the edge density of subgraph H'_i as $d_{edge}(H'_i) = \frac{|E_{H'_i}|}{|U_{H'_i}| \times |V_{H'_i}|}$. The total density of a hierarchy h is given by $D_h = \sum_{i=1}^{k} d_{edge}(H'_i)$. The mean height of h is $\overline{A_h} = \frac{A_h}{k}$ and its mean density is $\overline{D_h} = \frac{D_h}{k}$.

For a better understanding of the definitions, we show in Fig. 2 two hierarchical decompositions for the same bipartite graph as well as the total height, total density, mean height, and mean density for each. We recall that the density and the size of a subgraph are computed taking into account all the nodes in the subgraph hierarchy. For instance, for the subgraph B in Fig. 2b, the density is computed taking the ratio between the sum of the internal edges to B, the internal edges to C, and the edges that connect C to B, which are 3, 20, and 3, respectively, over the total possible edges for the bipartite subgraph, considering all the nodes from C and B. The density is, thus, $\frac{3+20+3}{5\times13} = 0.4$, and the size is the sum of the number of nodes of the subgraphs B and C, $5 + 13 = 18$. Since A is the first and comprises the whole graph, $l(A) = 1$. For subgraph B, which is an immediate subgraph from A, its level is given by $l(B) = l(A) + 1 = 2$.

In this work, we aim to find the hierarchical decomposition h of H such that the mean height, $\overline{A_h}$, is minimized, and the mean density, $\overline{D_h}$, is maximized. It is worth mentioning that hierarchical decomposition differs from hierarchical clustering since the union of the subgraphs at the leaf of the hierarchy does not necessarily reconstruct H, as in the latter case.

4. DENSE HIERARCHY DECOMPOSITION BY *WEIGHTED LINKING*

In this section we present our algorithm, *Weighted Linking*, which extends *Tip Decomposition*. As mentioned in Section 2, *Tip Decomposition* and *Wing Decomposition* decompose the graph based on the

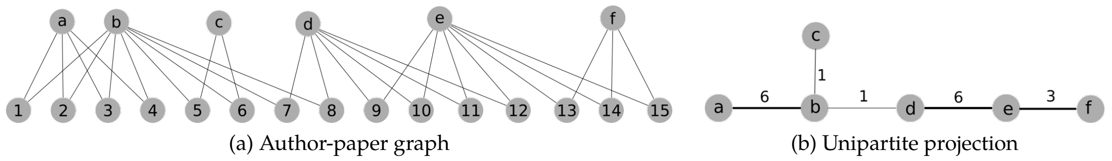


Fig. 3. Author-paper graph and its unipartite projection considering shared butterflies.

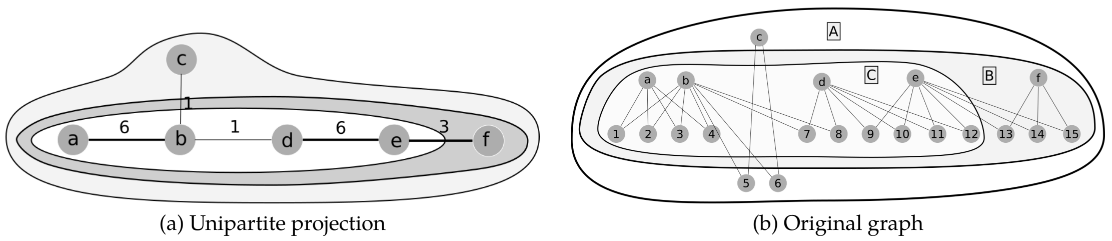


Fig. 4. Hierarchical decomposition of the graph shown in Fig. 3a by *Tip Decomposition* for the projected and original graphs
.

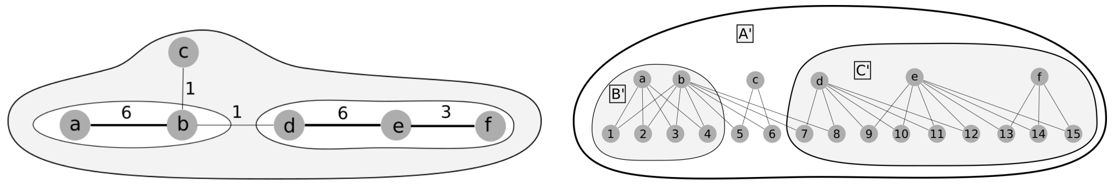
butterflies (2,2-biclique) shared by the nodes, in the former case, or by the edges, in the latter case. Both methods apply the heuristic proposed by [Sariyüce and Pinar 2016] to define the subgraphs and the hierarchy. Although this heuristic is useful for building the hierarchy, it does not consider the connection strength between dense subgraphs, and a less dense subgraph may result of the merge of denser ones. Because we aim to build a hierarchy of graph nodes, we consider only the *Tip Decomposition* algorithm in this work.

We can think of *Tip Decomposition* as a decomposition of the bipartite graph projection. The unipartite projection of the bipartite graph is constructed by taking the nodes from one partition and connecting them if they share butterflies. A weight is assigned to each edge as the number of butterflies shared by the end nodes. The node's weighted degree is computed by adding the weights of its associated edges. The decomposition process removes from the graph the node with the smallest weighted degree at each step, assigning the *tip number* as the current value of its weighted degree. The removal of a node forces the update of the neighboring nodes weighted degree by discounting the weight of the removed edges. The process stops when all nodes are processed. Connected nodes having the same *tip number* (or *tipness*), regardless of the edge weight connecting them, are added to the subgraph. Nodes having distinct *tip numbers*, regardless of the edge weight connecting them, are placed in different hierarchy levels.

Consider the author-paper graph of Fig. 3a, where {*a, b, c, d, e, f*} represent authors and {*1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15*} denote publications. The unipartite projection is shown in Fig. 3b. The edge weight shown in the projection means the number of butterflies shared between two authors. For instance, nodes *a* and *b* share nodes {*1, 2, 3, 4*}. Hence, the number of butterflies shared by *a* and *b* may be computed as $\binom{4}{2}$, which is equals to 6.

By running the *Tip Decomposition* algorithm, we obtain the hierarchy shown in Fig. 4a in the unipartite graph. The tip value of nodes a, b, c, d, e, and f are 6, 6, 1, 6, 6, and 3, respectively. The hierarchy in the original graph is shown in Fig. 4b. The subgraph levels and densities are shown in Table I. The total height is 6, and the total density is 1.23. Thus, the mean height is $\frac{6}{3} = 2$, and the mean density is $\frac{1.23}{3} = 0.41$.

There are, however, two well defined dense subgraphs that have been merged, induced by $\{a, b\}$ and $\{d, e\}$. Notice also the extra level induced by $\{f\}$ (Fig. 4a), even f being strongly linked to e. To overcome this issue, we propose an algorithm called *Weighted Linking*. *Weighted Linking* considers both the *tip number* of the end nodes and the link weight to build the subgraphs. Given two adjacent



(a) Unipartite projection (b) Original graph

Fig. 5. Hierarchical decomposition of the graph shown in Fig. 3a by *Weighted Linking* for the projected and original graphs
.

Subgraph	Density	Level
A	$\frac{30}{6\times 15} = 0.33$	1
B	$\frac{26}{5\times 13} = 0.4$	2
C	$\frac{20}{4\times 10} = 0.5$	3

Table I. Subgraph levels and densities by *Tip Decomposition* for each subgraph shown in Fig. 5

nodes, u and v, linked by an edge weighting p, *Weighted Linking* merges u and v in the same subgraph if $p > \alpha\frac{\theta(u)+\theta(v)}{2}$, where α regulates the minimum weight required for merging. We say an edge is weak if its weight does not satisfy the merging condition. Otherwise, we say the edge is strong.

The pseudocode for the *Weighted Linking* algorithm is shown in Algorithm 1. It extends the algorithm proposed by [Sariyüce and Pinar 2016]. We also use the same *subnucleus* structure to track nodes in the subgraph as in the original work. We now give a general view of the algorithm and focus on our contribution, more specifically lines 16-21 and 25-30. The unipartite projection is computed at line 2, keeping partition U nodes in the projected graph. The main loop (6-41) inspects each node, in ascending order of butterfly count (the number of butterflies the node participates in). At line 8, the current butterfly count of each node is assigned as its *tip number*. A new subnucleus used if the inspected node is not assigned to any existing one (36-39) is created at line 9. The neighbors inspection is performed at the code block from line 10 to 35. Nodes that have not been processed yet have their butterflies count decremented by the number of butterflies shared with the current node, bounded by the *tip number* of the current node (11-14). Nodes with the same *tip number* are inspected at lines 15-23.

As mentioned, our first contribution is at lines 16-21, as detailed next. In the original algorithm, the nodes are joined regardless of the edge weight. Our algorithm, on the other hand, places the nodes in the same subgraph only if they share enough butterflies (that is, they are connected by a strong edge), avoiding, thus, joining nodes that may decrease the subgraph density. The neighbors presenting the largest *tip number* are handled in lines 24-34. Our second contribution then comes in lines 25-30. In contrast with the original algorithm, where nodes having the largest *tip number* among all neighbors are set as direct children in the hierarchy, despite the edges' weights, we allow nodes in the same subgraph to be joined, even if they have different *tip numbers* , but only if they are strongly connected. We then avoid creating unnecessary hierarchy levels. If the edge is weak, the neighboring node is taken as a direct child in the hierarchy (line 32).

The complete parent-child relationship for the subnuclei is built at line 42. The subgraphs may be reconstructed by traversing both the hierarchy (*hrc*) and the subnucleus for each node (*comp*). Algorithms *Union-r* and *BuildHierarchy* are the same presented by [Sariyüce and Pinar 2016]. We remark that our modifications do not change the complexity of the original algorithm.

Fig. 5 shows the hierarchy for the graph of Fig. 3a built by the *Weighted Linking* algorithm by setting $\alpha = 0.6$. Since a weak edge links subgraphs induced by $\{a, b\}$ and $\{d, e\}$, they are placed in distinct subgraphs. Furthermore, the node f is added to the same subgraph induced by $\{d, e\}$, since it is related to e by an edge weighting above the threshold.

```
function WeightedLinkingDecomposition
    input  : $H = (U, V, E), \alpha$
    output: Hierarchy root node
    $K \leftarrow$ ButterflyCounts($H_U$);
    comp($u$) $\leftarrow$ $undefined$ $\forall\ u \in U$;
    $hrc \leftarrow \emptyset$;
    $ADJ \leftarrow \emptyset$;
    foreach unprocessed $u \in U$ with minimum butterfly count do
        $t \leftarrow$ ButterflyCount($u$);
        $\theta(u) \leftarrow t$;
        sn $\leftarrow$ subnucleus($t$);
        foreach $n \in$ Neighbors($u$) do
            foreach unprocessed $n$ do
                $w \leftarrow K[n, u]$;
                ButterflyCount($n$) $\leftarrow$ max(ButterflyCount($n$) $- w, t$);
            end
            foreach $n\ :\ \theta(n) = \theta(u)$ do
                if $K[n, u] > \alpha\frac{\theta(n)+\theta(u)}{2}$ then
                    if comp ($u$) is undefined then
                        comp($u$) $\leftarrow$ comp($n$);
                    else
                        Union-r(comp($u$), comp($n$));
                    end
                end
            end
            foreach $n\ :\ \theta(n) \neq \theta(u) \wedge \theta(n) =$ MaxTipness(Neighbors($u$)) do
                if $K[n, u] > \alpha\frac{\theta(n)+\theta(u)}{2}$ then
                    if comp ($u$) is undefined then
                        comp($u$) $\leftarrow$ comp($n$);
                    else
                        Union-r(comp($u$), comp($n$));
                    end
                else
                    $ADJ.add$(comp($u$), comp($n$));
                end
            end
        end
        if comp ($u$) is not set then
            comp($u$) $\leftarrow$ sn;
            $hrc.add$(sn);
        end
        Update all ($undefined$, .) $\in ADJ$ with (comp($u$), .);
    end
    BuildHierarchy($ADJ, hrc$);
    root $\leftarrow$ subnucleus(0);
    $hrc.add$(root);
    Report all the nuclei by hrc, comp;
end
```

Algorithm 1: Weighted linking decomposition.

Subgraph	Density	Level
A'	$\frac{30}{6\times15} = 0.33$	1
B'	$\frac{8}{2\times4} = 1$	2
C'	$\frac{16}{3\times9} = 0.59$	2

Table II. Subgraph levels and densities by *Weighted Linking* for each subgraph shown in Fig. 5
.

Graph	Nodes in partition 1	Nodes in partition 2	Total edges	Edges density
Condmat	16726	22015	58595	0.0001591292
Marvel	6486	12942	96662	0.001151536
DBLP	95580	93700	290365	0.00003242184
Github	56519	120867	440237	0.00006444427
DG-AssocMiner	7294	519	21357	0.005641663
Actor movies	511463	127823	1470404	0.000022491

Table III. Main characteristics of the databases used in the experiments
.

For the decomposition of Fig. 5, the subgraph levels and densities are shown in Table II. The total height is 5, and the total density is 1.92. The mean height is $\frac{5}{3} = 1.67$, and the mean density is $\frac{1.92}{3} = 0.64$. Therefore, we have a more compact and denser decomposition than that of Fig. 4.

5. EXPERIMENTAL RESULTS

5.1 Data sets

We evaluate *Weighted Linking* on six real-world datasets that are described next. Table III shows the main characteristics of the datasets used in the experiments, such as the number of nodes in each partition, the total number of edges, and the edge density.

DBLP[3] is a large bibliographic database on Computer Science with publications metadata from the leading conferences and journals on this topic. We build the bipartite graph author-publication by considering a subset with publications at 23 conferences, namely AAAI, IAAI, CIKM, CVPR, ECIR, ECML, PKDD, EDBT, ICDT, ICDE, ICDM, ICML, IJCAI, KDD, PAKDD, PODS, SDM, SIGIR, SIGMOD, SSDBM, VLDB, WSDM, and WWW. We consider the dataset of all publications up to August 2019. **Condmat**[4] [Kunegis 2013] is a bipartite graph modeling relations between authors and publications of the *Condensed Matter* section from *arXiv*[5]. One partition represents authors, and the other represents papers. An edge indicates the author collaborates in the paper authoring. **Github**[6] [Kunegis 2013] is a platform for source code-hosting that allows developers to collaborate on public and private projects. The bipartite network is modeled by joining developers, in one partition, to projects in the other. **Marvel**[7] [Alberich et al. 2002] models the occurrence of Marvel characters in comics. One partition represents characters, and the other represents comics. There is an edge connecting nodes in the two partitions whenever a character appears in a comic. **DG-AssocMiner**[8], which is part of the BioSNAP Dataset [Zitnik et al. 2018], is a disease-gene association network that contains information on genes associated with diseases. The partitions are genes and diseases; an edge connects a gene associated with a disease. **Actor movies**[9][Kunegis 2013] is a bipartite network

[3]`http://dblp.uni-trier.de/db/` (Date accessed October 06, 2020).
[4]`https://arxiv.org/archive/cond-mat` (Date accessed October 06, 2020)
[5]`https://arxiv.org/` (Date accessed October 06, 2020)
[6]`https://github.com/` (Date accessed October 06, 2020)
[7]`http://bioinfo.uib.es/~joemiro/marvel/porgat.txt` (Date accessed October 06, 2020)
[8]`https://snap.stanford.edu/biodata/datasets/10012/10012-DG-AssocMiner.html` (Date accessed October 06, 2020)
[9]`http://konect.uni-koblenz.de/networks/actor-movie` (Date accessed October 06, 2020)

of movies and actors. An edge exists between the two partitions if an actor portrays a character in a movie.

5.2 Analysis Methodology

One challenge in analyzing the effectiveness of different algorithms or algorithms' parameters is that there may be significantly different decompositions of the same graph.

The basic question for our analysis is how to determine the best decomposition for a given bipartite graph H. In particular, considering two different decompositions, h_1 and h_2, we need to rank them regarding the decomposition quality, which, in our case, means more compact decompositions. Thus, the desired subgraphs should be larger and denser, as well as located at lower levels.

As far as we know, no work in the literature measures the quality of the hierarchical decomposition of a graph. Then, to answer this question, we propose a *Pairwise Hierarchical Decomposition Quality Score ($PHDscore$)* between h_1 and h_2.

Our quality score is calculated at the node level, and the node comparison criteria may vary among decomposition tasks, but they always take into account features that characterize the nodes individually. In practice, for each node from H, we evaluate a set of criteria for both h_1 and h_2, and the decomposition that outperforms the other more frequently across all nodes is considered the best between them, as we detail next.

In the case of our hierarchical decomposition task, we assign features to the nodes according to the subgraph that contains them. In particular, we employ three features: (i) $f_{u,size}$, (ii) $f_{u,density}$, and (iii) $f_{u,level}$:

(i) $f_{u,size}$ is the number of nodes of the innermost subgraph containing node u. The larger the subgraph, the greater the $f_{u,size}$.
(ii) $f_{u,density}$ is the density of the innermost subgraph containing node u. The denser the subgraph, the greater the $f_{u,density}$.
(iii) $f_{u,level}$ is the opposite of the level of the innermost subgraph containing node u. The lower the level, the greater the $f_{u,level}$.

Formally, given two hierarchical decompositions, h_1 and h_2, and a set $F = \{f_i\}$ of features that are relevant to the decomposition task and are instantiated at the node level, we define the $PHDScore$ of h_1 w.r.t. h_2, as:

$$PHDscore(h_1, h_2, H) = \frac{\sum_{\forall u \in U} I(label(u) = h_1)}{\sum_{\forall u \in U} I(label(u) = h_1) + \sum_{\forall u \in U} I(label(u) = h_2)}$$

where

$$label(u) = \begin{cases} h_1,\ if\ \sum_{i=1}^{|F|} I(f_{u,i}(h_1, H) > f_{u,i}(h_2, H)) > \sum_{i=1}^{|F|} I(f_{u,i}(h_2, H) > f_{u,i}(h_1, H)), \\ h_2,\ if\ \sum_{i=1}^{|F|} I(f_{u,i}(h_2, H) > f_{u,i}(h_1, H)) > \sum_{i=1}^{|F|} I(f_{u,i}(h_1, H) > f_{u,i}(h_2, H)), \\ None,\ otherwise \end{cases}$$

$$I(x) = \begin{cases} 1,\ if\ x\ is\ true, \\ 0,\ otherwise \end{cases}$$

We define $PHDscore(h_1, h_2, H) = 0$ if $f_{u,i}(h_2, H) = f_{u,i}(h_1, H)\ \forall\ u \in U,\ f_i \in F$. We state that h_1 produces better results than h_2 for nodes features F for a given node if, for most features, h_1 performs better than h_2. The best decomposition for a graph is the one that produces the best

	h_1				h_2				
Node	**Subgraph**	$f_{.,size}$	$f_{.,density}$	$f_{.,level}$	**Subgraph**	$f_{.,size}$	$f_{.,density}$	$f_{.,level}$	**Label**
a	B′	6	**1**	**-2**	C	**14**	0.5	-3	h_1
b	B′	6	**1**	**-2**	C	**14**	0.5	-3	h_1
c	A′	21	0.33	-1	A	21	0.33	-1	-
d	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
e	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
f	C′	12	**0.59**	-2	B	**18**	0.4	-2	-
1	B′	6	**1**	**-2**	C	**14**	0.5	-3	h_1
2	B′	6	**1**	**-2**	C	**14**	0.5	-3	h_1
3	B′	6	**1**	**-2**	C	**14**	0.5	-3	h_1
4	B′	6	**1**	**-2**	C	**14**	0.5	-3	h_1
5	A′	21	0.33	-1	A	21	0.33	-1	-
6	A′	21	0.33	-1	A	21	0.33	-1	-
7	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
8	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
9	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
10	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
11	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
12	C′	12	**0.59**	**-2**	C	**14**	0.5	-3	h_1
13	C′	12	**0.59**	-2	B	**18**	0.4	-2	-
14	C′	12	**0.59**	-2	B	**18**	0.4	-2	-
15	C′	12	**0.59**	-2	B	**18**	0.4	-2	-

Table IV. Size, density, and opposite of the level of the inner most subgraph where each node of Fig. 2 appears, for hierarchies h_1 and h_2. Bold values indicate the features where each hierarchy outperforms the other for each node. The last column shows the label for each node. In case no hierarchy performs better than the other for the node, no label is assigned to it (denoted by '-' in the table).

result on most of its nodes. Thus, if $PHDscore(h_1, h_2, H) > 0.5$, h_1 should be ranked better than h_2. Otherwise, h_2 should come first. Notice that $PHDscore(h_2, h_1, H) = 1 - PHDscore(h_1, h_2, H)$, since the denominator is the same for both scores and just the decomposition used as reference changes.

The rationale of the PHDScore is that it prioritizes the decomposition where the nodes perform better individually, considering the chosen feature set (F).

For the sake of clarity, we show, in Table IV, the label for each node of the graph of Fig. 2, when comparing hierarchies h_1 and h_2, considering the feature defined previously. For the node a, for instance, we have $f_{a,density}(h_1, H) > f_{a,density}(h_2, H)$, $f_{a,level}(h_1, H) > f_{a,level}(h_2, H)$, and $f_{a,size}(h_1, H) < f_{a,size}(h_2, H)$. The label for the node a is, thus, h_1, since h_1 wins in more features (2) than h_2 (1). The score for h_1 in respect to h_2, $score(h_1, h_2, H)$, is $\frac{14}{14+0}$, that equals to *1*. Then, h_1 must be chosen instead of h_2.

5.3 Experimental Results

To evaluate our algorithm, we compare it with the state of the art approach, *Tip Decomposition*. We search for the best parameter α for *Weighted Linking* that outperforms *Tip Decomposition* on most nodes. Then, we discuss the semantics of some subgraphs found by *Tip Decomposition* and *Weighted Linking* for the best parameter configuration.

We show in Fig. 6 the $PHDscores$ for our approach (WL α) compared to *Tip Decomposition* for all data sets and several α values. We run *Weighted Linking* and vary α as percentages $\{1, 2, 3, 4, 5, 10, 15, 20, 40, 60, 80, 100\}$. A score greater than 0.5 indicates that our method outperforms *Tip Decomposition* for the associated α. We chose the WL α with the highest score as the best decomposition by *Weighted Linking*. In case all configurations score below 0.5, *Tip Decomposition* is chosen. According to our score, *Weighted Linking* generates the best hierarchy for the data sets *Condmat*, *Marvel*, *DBLP*, *Github*, and *Actor movies*; *Tip Decomposition*, on the other hand, performs better than *Weighted Linking* for *DG-AssocMiner*. We show in Table V a summary of the best α value for *Weighted Linking* compared to *Tip Decomposition* and the associated $PHDscore$.

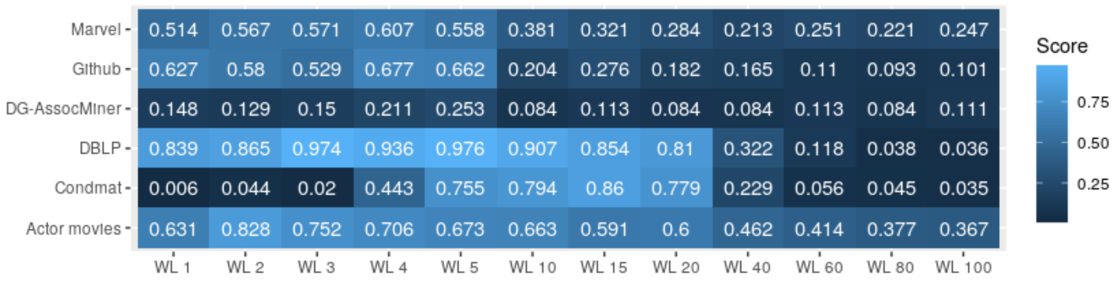


Fig. 6. *PHDscore* for *Weighted Linking* (WL) concerning *Tip Decomposition*. Each cell represents the score for *Weighted Linking* relative to *Tip Decomposition* for each dataset. The columns represent *Weighted Linking* decomposition with different α. The rows indicate the datasets. The greater the score, the better the *Weighted Linking* decomposition.

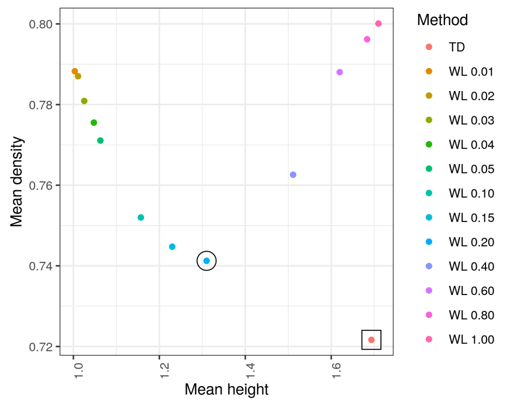
Graph	α (%)	PHDscore
Condmat	15	0.86
Marvel	4	0.607
DBLP	5	0.976
Github	4	0.677
DG-AssocMiner	5	0.253
Actor movies	2	0.828

Table V. Best α value for *Weighted Linking* and the corresponding *PHDscore* concerning *Tip Decomposition*.

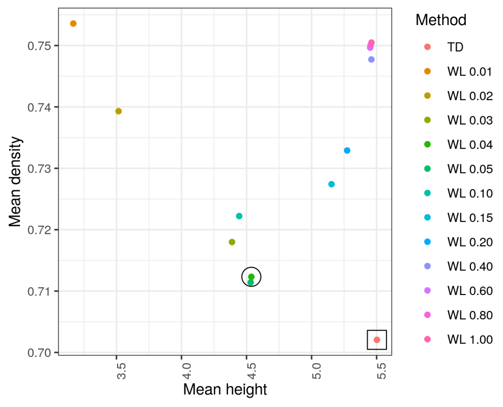
In Fig 6, we can also observe the concave behavior of the score value as the α value grows. For smaller α values, *Weighted Linking* is more permissive in joining nodes in the same subgraph, inducing larger and more sparse subgraphs and a smaller hierarchy as a consequence. On the other hand, for higher α values, *Weighted Linking* inhibits the union of the nodes in the same subgraph, leading to smaller and denser subgraphs. For intermediate α values, however, we have a better balance between the density and size of the subgraphs, as well as the height of the hierarchy. We can also observe for each data set that, despite the same general behavior of the score as a function of α, the optimal score is reached for different α values. This behavior is due to the particular topological structure of each graph.

We show in Fig. 7 the mean density and mean size of the subgraphs for the hierarchical decomposition of *Condmat*, *Marvel*, *DBLP*, *Github*, *Actor movies*, and *DC-AssocMiner* by *Weighted Linking*, with α values in percentages $\{1, 2, 3, 4, 5, 10, 15, 20, 40, 60, 80, 100\}$, and also by *Tip Decomposition*. The best *Weighted Linking* decomposition is marked with a circle, and *Tip Decomposition* is highlighted with a square. Smaller α values tend to generate more compact hierarchies with sparser subgraphs, while higher values generate higher hierarchy trees and denser subgraphs. The reason for this behavior is that, for smaller α values, the count of butterflies (or edge weights in the projected graph) needed to push nodes to the same subgraph is smaller, favoring the appearance of sparser subgraphs. On the other hand, higher α values require stronger links to put nodes together, producing smaller and denser subgraphs while promoting more splits and increasing the hierarchy levels. The higher mean density for smaller α values is justified by dense and isolated subgraphs that cannot be merged with any other. For the *Condmat*, *Marvel*, *DBLP*, and *Actor movies*, we can observe that the best decomposition has both smaller mean density and smaller mean hierarchy height than *Tip Decomposition*.

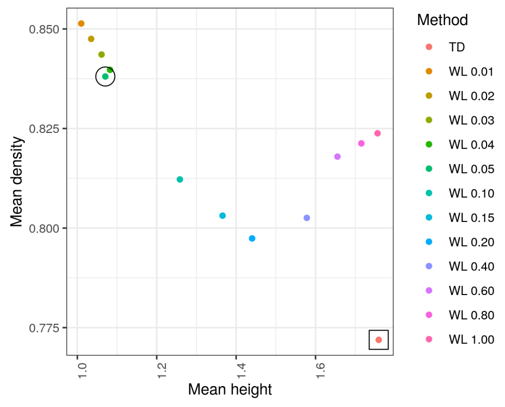
Weighted Linking also performs better than *Tip Decomposition* for *Github*. However, when α is set to 100%, we can observe that the mean height is smaller than for α at 1%. Since there are many weakly connected subgraphs at lower levels of the hierarchy, the increase in α produces more subgraphs at the same lower levels. The mean height computation is then dominated by the increase in the denominator (number of subgraphs).



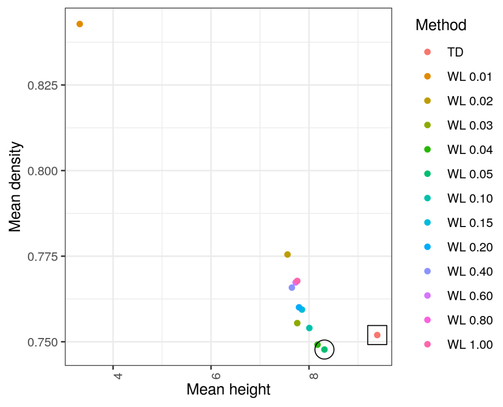
(a) *Condmat*



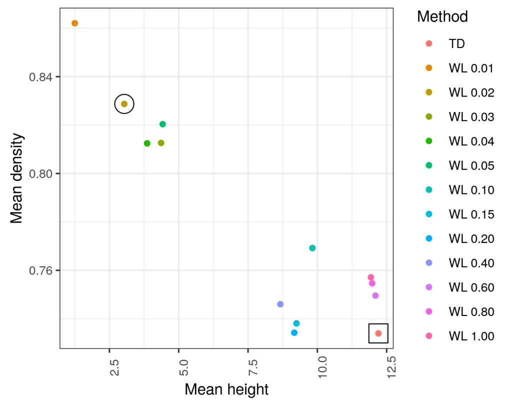
(b) *Marvel*



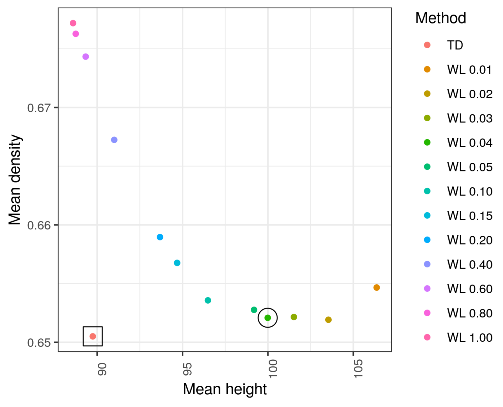
(c) *DBLP*



(d) *DG-AssocMiner*



(e) *Actor movies*



(f) *Github*

Fig. 7. Mean height versus mean density of the subgraphs for the hierarchical decomposition by *Weighted Linking* (WL), with different α values, and *Tip Decomposition* (TD). The square marker is for TD, and the circle highlights the result for the best WL execution.

For *DG-AssocMiner*, however, *Tip Decomposition* gives a better result than *Weighted Linking*. The reason is that there are many genes associated almost exclusively with a few diseases. Then, the edges connecting such genes in the projected graph become weak since the number of shared butterflies between nodes is small, even when the nodes' *tip number* is high. In this case, *Weighted Linking* would only place the nodes in the same subgraph if we set α to a very small value. The side effect of reducing the threshold is that nodes with smaller *tip numbers* would also be merged, leading to

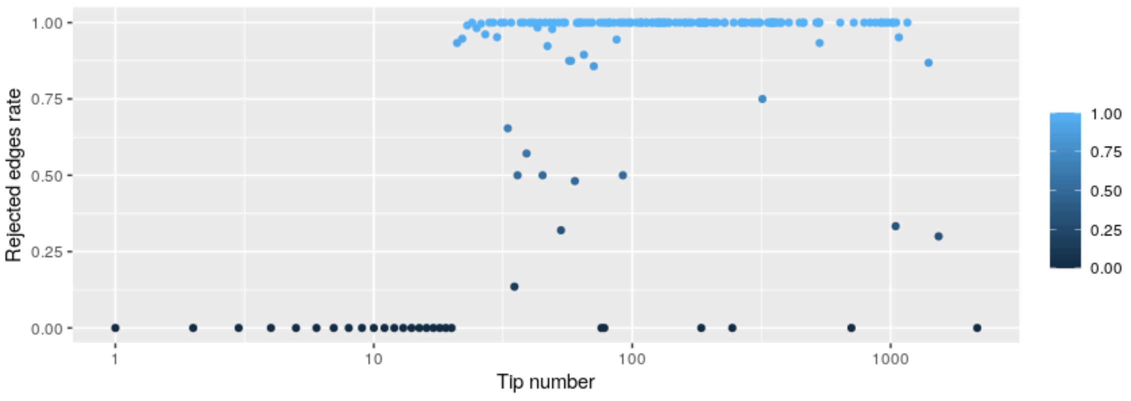
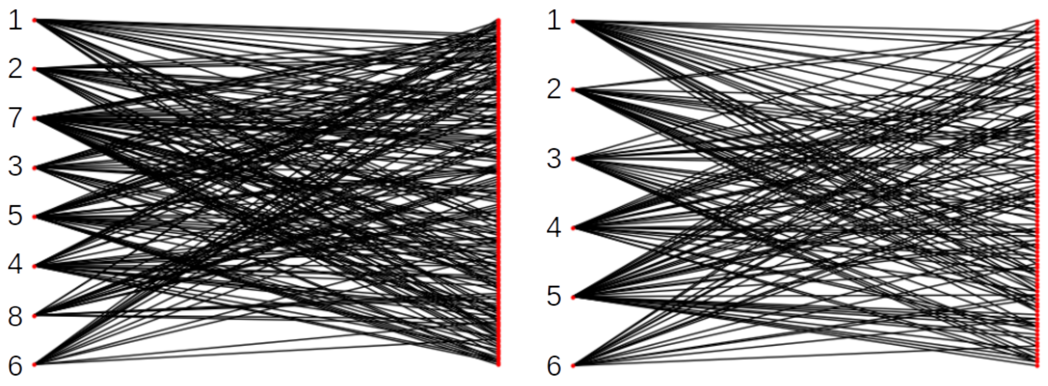


Fig. 8. Rejected edges rate for each *tip number* on *DG-AssocMiner* data set.

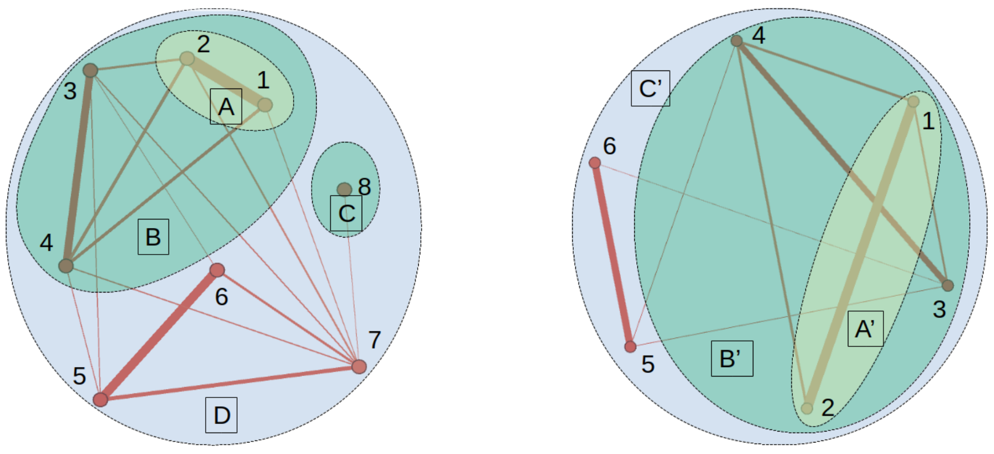


(a) *Tip decomposition* (b) *Weighted linking*

Fig. 9. First subgraph extracted from *DBLP* where node 6 appears. Nodes represent authors, and links represent co-authorship on at least two publications (at least one butterfly). Link thickness represents the number of shared butterflies between two authors.

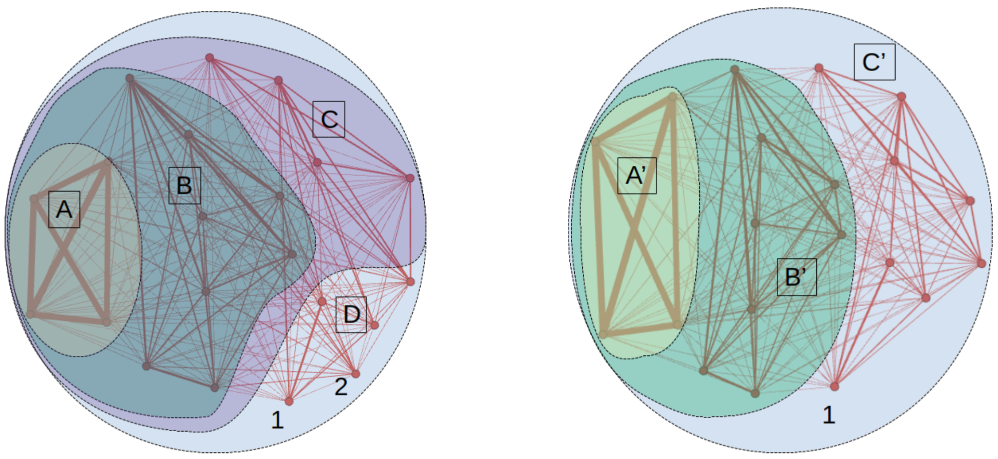
sparser subgraphs. We show in Fig. 8 the rejected edges rate, that is, the rate of the edges weight that are smaller than the threshold, for each *tip number* on the *DG-AssocMiner* data set when running *Weighted Linking* with α set to 5%. The rejected rate around 1 for higher *tip numbers* suggests that we are missing many large bicliques.

We show in Fig. 9 two most internal subgraphs from DBLP, one generated by *Tip Decomposition* and the other by *Weighted Linking*. The numbered nodes on the left represent authors, and the nodes on the right represent publications. We show the innermost subgraph containing the author represented by node 6. The subgraph produced by *Tip Decomposition* (Fig. 9a) has 8 authors, 137 publications and, 214 author-publication relationships, so the density is 0.195. The subgraph produced by *Weighted Linking (WL 5)* (Fig. 9b) comprises 6 authors, 89 publications, and 146 author-publication relations, giving a density of 0.273. We also show in Fig. 10 the subgraphs hierarchy for both decompositions in the projected unipartite graph. Nodes represent authors, and edges represent the co-authorships. The link weight represents the co-authorship strength, computed as the butterfly count involving the two authors. Fig. 10a shows the *Tip Decomposition* hierarchy, where the nodes in each subgraph *A, B, C,* and *D* share the same tip number. The only criterion employed to build the hierarchy is the tip number. Notice that node 7 is placed in the same subgraph as nodes 5 and 6 because it has the same tip number, despite the edges connecting it to nodes 5 and 6 being weaker than between these two nodes. The inclusion of node 7 also brings node 8 to the subgraph, which is not connected to



(a) *Tip decomposition*   (b) *Weighted linking (α=3%)*

Fig. 10. Hierarchies by *Tip Decomposition* and *Weighted Linking* of the first subgraph extracted from DBLP where node 6 appears. Nodes represent authors and links represent co-authorship of, at least, two publications. Link thickness represents the number of butterflies shared between two authors. *Weighted Linking* improves the interpretability by producing denser subgraphs with fewer hierarchical levels.



(a) *Tip decomposition*   (b) *Weighted linking (4%)*

Fig. 11. Hierarchies of a subgraph extracted from Marvel. Nodes represent characters, and links represent co-occurrence in at least two comics (one butterfly). Link thickness represents the number of shared butterflies between two characters. Nodes 1 and 2 represent *Wolverine* and *Wonder Man*, respectively. *Weighted linking* adds *Wolverine* to the third level of the hierarchy, while *Tip Decomposition* adds it to the fourth. *Wonder man* is left out of *Wolverine*'s subgraph by *Weighted Linking*.

any other node in that context. The hierarchy built by *Weighted Linking* with α set to 5% is shown in Fig. 10b. Authors 7 and 8 are excluded because their main collaborators are outside this subgraph. Besides improving the density, we also have fewer hierarchy levels. Authors 1 and 2, at the deepest level of this hierarchy, represent authors who collaborate the most (subgraph A'). At the next level, we can find authors 3 and 4. They have written many papers together and collaborated, to some extent, with authors 1 and 2 (subgraph B'). Author 6, the focus of our analysis, together with author 5, is located at the outermost hierarchy level since they have co-authored fewer papers than the others (subgraph C').

Fig. 11 shows the subgraph hierarchy containing the character *Wolverine* (node 1) from the *Marvel* database. Both hierarchies agree on the subgraphs A and A', which represent *The Fantastic Four*;

subgraphs B and B' are also compatible as they add characters from the *Avengers*. Subgraph C', however, can be better interpreted as *The X-Men*, in contrast with subgraph C, which does not give a direct mapping to a well known Marvel group. Subgraph C' puts together all characters from *The X-Men* found in C along with *Professor X, Wolverine*, and *Storm*, which are known to be part of the group[10]. Note that node 2, which represents *Wonder Man*[11], is not present in the subgraph C'. One possible explanation for the absence is that *Wonder Man* first appeared in *The X-Men* comics but disappeared for several years. This fact justifies the placement of this character in the upper levels of the hierarchy since it has fewer co-occurrences with the other X-Men members. Notice also the extra level added in the subgraph of Fig. 11a without aggregating any new information. Indeed, the X-Men group is divided into subgraphs C and D by *Tip Decomposition*. Thus, *Tip decomposition* adds an unnecessary extra level and extra nodes, making the interpretation less obvious.

Hence, the subgraph generated by *Weighted Linking* is able to explain real-world scenarios better, demonstrating the effectiveness of our algorithm in generating more interpretable hierarchies.

6. CONCLUSION

Organizing a graph as a hierarchy of dense subgraphs is a challenging task, especially considering the edge density, since even a pair of connected vertices has a maximum density of 1. Besides, hierarchies with several levels lose the semantics of the subgraphs because they force the splitting of nodes that, together, explain real-world scenarios better. In this work, we show empirically that there is a trade-off between the number of hierarchical levels of decomposition and its overall density. We propose the *Weighted Linking* method to handle this problem, which allows controlling the compromise between larger densities and shorter hierarchies. We also provide a score to compare whether a hierarchical decomposition is better than another. We show, empirically, that our algorithm outperforms the state of the art approach in most of the presented real-world scenarios.

However, we need to investigate some aspects of the algorithm further. An interesting behavior that we were able to observe is that *Tip Decomposition* is able to detect a particular case of bicliques where *Weighted Linking* fails unless we set a very small value to α. Consider the case of many authors who are co-authors of a few papers, let us say two, and do not collaborate with other authors, in the author-paper graph. This subset of authors and papers produces a biclique subgraph with many nodes at the authors' partition and only two in the papers' partition. Hence, when we compute the unipartite projection, the projected subgraph will present a clique for the authors' nodes where all the links are weighted as one since all the authors share only one butterfly. Therefore, the nodes' weighted degrees will be large, even with each link having a small weight, and all the nodes will be assigned the same high *tip number*. Since all nodes have exactly the same *tip number*, and the weight of the link is not taken into account when building the subgraphs, *Tip Decomposition* joins all nodes in the same subgraph. On the other hand, since the edge weight is small, unless we set lower values to α in *Weighted Linking*, the nodes tend to be divided into different subgraphs. An improvement that we plan to make in the future is the ability to handle such situations.

We also intend to explore the parametric space to estimate the optimal value for α that produces a result closer to the optimal decomposition. In particular, we plan to compare several decompositions by *Weighted Linking* with different parameter settings to identify the decomposition that balance the best the subgraphs density and hierarchy height, according to the $PHDscore$.

We believe that the score we propose can also be applied to other configurations besides the hierarchical decomposition of bipartite graphs. We intend to extend it to unipartite, as well as to attributed graphs, as future work.

[10]https://en.wikipedia.org/wiki/X-Men (Date accessed October 06, 2020)
[11]https://en.wikipedia.org/wiki/Wonder_Man (Date accessed October 06, 2020)

ACKNOWLEDGMENT

All the authors would like to thank FAPEMIG, CNPq, and CAPES for their financial support. This work was also partially funded by projects InWeb, MASWeb, EUBra-BIGSEA, INCT-Cyber, and ATMOSPHERE.

REFERENCES

Abello, J. and Korn, J. Mgv: A system for visualizing massive multidigraphs. *IEEE Transactions on Visualization and Computer Graphics* 8 (1): 21–38, 2002.

Al-garadi, M. A., Varathan, K. D., and Ravana, S. D. Identification of influential spreaders in online social networks using interaction weighted k-core decomposition method. *Physica A: Statistical Mechanics and its Applications* vol. 468, pp. 278–288, 2017.

Alberich, R., Miro-Julia, J., and Rossello, F. Marvel universe looks almost like a real social network, 2002. cite arxiv:cond-mat/0202174Comment: 14 pages, 3 figures.

Alvarez-hamelin, J. I., asta, L. D., Barrat, A., and Vespignani, A. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in Neural Information Processing Systems 18*, Y. Weiss, B. Schölkopf, and J. C. Platt (Eds.). MIT Press, pp. 41–50, 2006.

Andersen, R. and Chellapilla, K. Finding dense subgraphs with size bounds. In *Algorithms and Models for the Web-Graph*, K. Avrachenkov, D. Donato, and N. Litvak (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 25–37, 2009.

Asahiro, Y., Iwama, K., Tamaki, H., and Tokuyama, T. Greedily finding a dense subgraph. *Journal of Algorithms* 34 (2): 203–221, 2000.

Ban, Y. On finding dense subgraphs in bipartite graphs: Linear algorithms with applications to fraud detection, 2018.

Batagelj, V. and Zaversnik, M. Generalized cores. *CoRR* vol. cs.DS/0202039, 2002.

Bonchi, F., Gullo, F., Kaltenbrunner, A., and Volkovich, Y. Core decomposition of uncertain graphs. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '14. ACM, New York, NY, USA, pp. 1316–1325, 2014.

Brown, P. E. and Feng, J. Measuring user influence on twitter using modified k-shell decomposition. In *Fifth international AAAI conference on weblogs and social media*, 2011.

Charikar, M. Greedy approximation algorithms for finding dense components in a graph. In *Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*. APPROX '00. Springer-Verlag, Berlin, Heidelberg, pp. 84–95, 2000.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., et al. Introduction to algorithms, 2001.

Galimberti, E., Bonchi, F., and Gullo, F. Core decomposition and densest subgraph in multilayer networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. ACM, pp. 1807–1816, 2017.

Giatsidis, C., Thilikos, D. M., and Vazirgiannis, M. Evaluating cooperation in communities with the k-core structure. In *2011 International Conference on Advances in Social Networks Analysis and Mining*. pp. 87–93, 2011.

Giatsidis, C., Thilikos, D. M., and Vazirgiannis, M. D-cores: measuring collaboration of directed graphs based on degeneracy. *Knowledge and information systems* 35 (2): 311–343, 2013.

Goldberg, A. V. Finding a maximum density subgraph. Tech. rep., Berkeley, CA, USA, 1984.

Khuller, S. and Saha, B. On finding dense subgraphs. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming: Part I*. ICALP '09. Springer-Verlag, Berlin, Heidelberg, pp. 597–608, 2009.

Kitsak, M., Gallos, L. K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H. E., and Makse, H. A. Identification of influential spreaders in complex networks. *Nature physics* 6 (11): 888, 2010.

Kunegis, J. Konect: The koblenz network collection. In *Proceedings of the 22Nd International Conference on World Wide Web*. WWW '13 Companion. ACM, New York, NY, USA, pp. 1343–1350, 2013.

Lee, V. E., Ruan, N., Jin, R., and Aggarwal, C. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*. Springer, pp. 303–336, 2010.

Malliaros, F. D., Giatsidis, C., Papadopoulos, A. N., and Vazirgiannis, M. The core decomposition of networks: theory, algorithms and applications. *The VLDB Journal*, 2019.

Moreira, E., Campos, G. O., and Meira Jr, W. Dense hierarchy decomposition for bipartite graphs. In *Anais do VII Symposium on Knowledge Discovery, Mining and Learning*. SBC, pp. 105–112, 2019.

Sariyüce, A. E., Gedik, B., Jacques-Silva, G., Wu, K.-L., and Çatalyürek, Ü. V. Incremental k-core decomposition: algorithms and evaluation. *The VLDB Journal—The International Journal on Very Large Data Bases* 25 (3): 425–447, 2016.

Sariyüce, A. E. and Pinar, A. Fast hierarchy construction for dense subgraphs. *Proceedings of the VLDB Endowment* 10 (3): 97–108, 2016.

Sariyüce, A. E. and Pinar, A. Peeling bipartite networks for dense subgraph discovery. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. ACM, New York, NY, USA, pp. 504–512, 2018.

Seidman, S. B. Network structure and minimum degree. *Social networks* 5 (3): 269–287, 1983.

Shin, K., Eliassi-Rad, T., and Faloutsos, C. Corescope: Graph mining using k-core analysis — patterns, anomalies and algorithms. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. pp. 469–478, 2016.

Tsourakakis, C., Bonchi, F., Gionis, A., Gullo, F., and Tsiarli, M. Denser than the densest subgraph: Extracting optimal quasi-cliques with quality guarantees. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '13. ACM, New York, NY, USA, pp. 104–112, 2013.

Wu, H., Cheng, J., Lu, Y., Ke, Y., Huang, Y., Yan, D., and Wu, H. Core decomposition in large temporal graphs. In *2015 IEEE International Conference on Big Data (Big Data)*. pp. 649–658, 2015.

Zitnik, M., Sosič, R., Maheshwari, S., and Leskovec, J. BioSNAP Datasets: Stanford biomedical network dataset collection. `http://snap.stanford.edu/biodata`, 2018.