

Peeling Bipartite Networks for Dense Subgraph Discovery

Ahmet Erdem Sariyüce
University at Buffalo
erdem@buffalo.edu

Ali Pinar
Sandia National Laboratories
apinar@sandia.gov

ABSTRACT

Finding dense bipartite subgraphs and detecting the relations among them is an important problem for affiliation networks that arise in a range of domains, such as social network analysis, word-document clustering, the science of science, internet advertising, and bioinformatics. However, most dense subgraph discovery algorithms are designed for classic, unipartite graphs. Subsequently, studies on affiliation networks are conducted on the co-occurrence graphs (e.g., co-author and co-purchase) that project the bipartite structure to a unipartite structure by connecting two entities if they share an affiliation. Despite their convenience, co-occurrence networks come at a cost of loss of information and an explosion in graph sizes, which limit the quality and the efficiency of solutions. We study the dense subgraph discovery problem on bipartite graphs. We define a framework of bipartite subgraphs based on the butterfly motif (2,2-biclique) to model the dense regions in a hierarchical structure. We introduce efficient peeling algorithms to find the dense subgraphs and build relations among them. We can identify denser structures compared to the state-of-the-art algorithms on co-occurrence graphs in real-world data. Our analyses on an author-paper network and a user-product network yield interesting subgraphs and hierarchical relations such as the groups of collaborators in the same institution and spammers that give fake ratings.

ACM Reference format:

Ahmet Erdem Sariyüce and Ali Pinar. 2018. Peeling Bipartite Networks for Dense Subgraph Discovery. In *Proceedings of WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, February 5–9, 2018 (WSDM 2018)*, 11 pages. DOI: 10.1145/3159652.3159678

1 INTRODUCTION

Many real-world systems are naturally modeled as affiliation, two-mode, or bipartite networks [9, 26]. In a bipartite network, vertices are decomposed into two disjoint sets, *primary* and *secondary*, such that edges can only connect vertices from different sets. For example authors and papers can be the primary and secondary vertex sets, with an edge representing authorship. Finding dense subgraphs in the real-world affiliation networks, and relating them to each other has been shown to be useful across different domains. Literature is rich with the examples such as spam group detection in web [21], word and document clustering [16], and sponsored search advertising on webpages [18]. Despite their representation

ACM acknowledges that this contribution was authored or co-authored by an employee, or contractor of the national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. Permission to make digital or hard copies for personal or classroom use is granted. Copies must bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. To copy otherwise, distribute, republish, or post, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org. WSDM 2018, Marina Del Rey, CA, USA
© 2018 ACM. 978-1-4503-5581-0/18/02...\$15.00
DOI: 10.1145/3159652.3159678

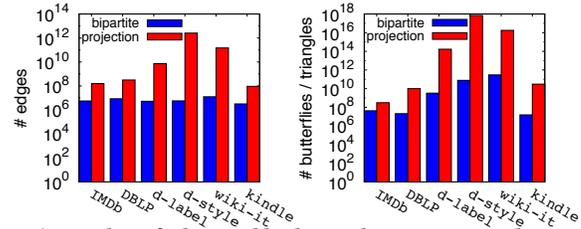


Figure 1: Number of edges and higher-order structures in bipartite graphs and their projections

power, bipartite graphs are underutilized, since most graph mining algorithms, including dense subgraph discovery, are studied on the traditional unipartite graphs. For this reason, affiliation networks are projected to co-occurrence graphs, such that two vertices in the primary set are connected by an edge if they share an affiliation. For instance, an author-paper network can be transformed into a co-authorship network, where two authors are connected if they co-authored a paper. However, this transformation comes at a cost of information loss and inflated graph size, as we will discuss in more detail later. Therefore, designing algorithms that can work directly on the bipartite graph, which provides an accurate representation of the underlying system, is essential.

This paper studies finding dense subgraphs in a *bipartite* graph and detecting the relations among them. Our approach is inspired by the k -core [30, 42] and k -truss [15, 38, 46, 48] decompositions in unipartite networks, which are instances of *peeling algorithms*. They have been shown to be effective to detect dense subgraphs with hierarchical relations [41].

1.1 Problem and Challenges

Our aim is to find many, if not all, dense regions in bipartite graphs and determine the relations among them. Despite a few successful studies that directly focus on bipartite networks [8, 13], a common practice in the literature for working with bipartite graphs has been creating co-occurrence (projection) graphs. Although the projection enables the use of well-studied unipartite graph mining algorithms [9], it has significant drawbacks:

- **Information loss and ambiguity:** Bipartite graphs comprise one-to-many relationship information, but this information is reduced to pairwise ties when projected to a weighted or unweighted unipartite form. Those pairwise ties are treated independently, which distorts the original information. In addition, projections are not bijective irrespective of the projection technique being used, which creates ambiguity.
- **Size inflation:** A secondary vertex in a bipartite network with degree d results in a d -clique in the projected graph. Thus, the number of edges in the projected graph can be as many as $\sum_{v \in V} \binom{d_v}{2}$, whereas it is only $\sum_{v \in V} d_v$ in the bipartite network, where V is the set of secondary vertices. Fig. 1 (left) shows

Table 1: Notations

$G = (U, V, E)$	bipartite graph with vertices in U and V , and edges E
$N(u, G), N(u)$	set of vertices that are connected to vertex u in G
$d(u)$	degree of vertex u , i.e., $ N(u) $
(a, b) -biclique	complete bipartite graph where $ U = a$ and $ V = b$
$G_p = (U, E_p)$	unweighted projection of G , as in Def. 1
$G_{wp} = (U, E_{wp})$	weighted projection of G , as in Def. 1
Δ_p	number of triangles in the projected graph G_p
\bowtie	butterfly or $(2, 2)$ -biclique
$\theta(u)$	tip number of vertex u , as in Def. 6
$\psi(e)$	wing number of edge e , as in Def. 8

the difference between some bipartite networks and their projections – we observe up to 6 orders of magnitude increase in size, which degrades the performance. This also artificially boosts the clustering coefficients and the local density measures in the projected graph by creating many triangles. Regarding the smallest higher-order structures, a projection can have up to 699 quadrillion triangles whereas its original bipartite network has 77 billion butterflies ($(2, 2)$ -biclique), as in Fig. 1 (right).

Given the drawbacks of the projection approaches, we work directly on the bipartite graph to discover the dense structures. It has been shown that the higher-order structures (motifs, graphlets) offer deeper insights for analyzing real-world networks and detecting dense regions in a better way [7, 23, 41, 44, 45]. Peeling algorithms, k -core and k -truss decompositions, find dense regions in unipartite graphs and determine the relations among them [22, 23, 48]. Nucleus decomposition [41] is a generalization of these two approaches and can work on higher-order structures such as 4-cliques. However, none of them are applicable for the bipartite networks. k -core decomposition assumes that all vertices represent the same kind of entity, which does not hold for bipartite graphs. k -truss decomposition works on triangles, which do not exist in bipartite graphs. Nucleus decomposition uses small-cliques, which are also nonexistent in bipartite graphs. Thus, we need **higher-order structures that capture the cohesiveness in bipartite graphs**. Then the peeling algorithms can be adapted to run on these structures to find the dense regions with hierarchical relations.

1.2 Contributions

We introduce new algorithms to efficiently find dense bipartite subgraphs with hierarchy of relations. Our contributions can be summarized as follows:

- **Introducing k -tip and k -wing bipartite subgraphs:** We survey attempts to define higher-order structures in bipartite graphs, and use the *butterfly* structure ($(2, 2)$ -biclique) as the simplest super-edge motif. Building on that, we define the k -tip and k -wing subgraphs based on the involvements of vertices and edges in butterflies, respectively.
- **Extension of peeling algorithms:** We introduce peeling algorithms to efficiently find all the k -tip and k -wing subgraphs. Our algorithms are inspired by the degeneracy based decompositions for unipartite graphs. We present detailed pseudocodes and analyze their complexities.
- **Evaluation on real-world data:** We evaluate our proposed techniques on real-world networks. Fig. 2 presents a glance of results on the IMDb movie-actor with 1.6M vertices and 5.6M edges. Our algorithms are able to extract larger and denser subgraphs of various sizes. We also examine the ratings data for the Amazon Kindle books and analyze the author-paper network

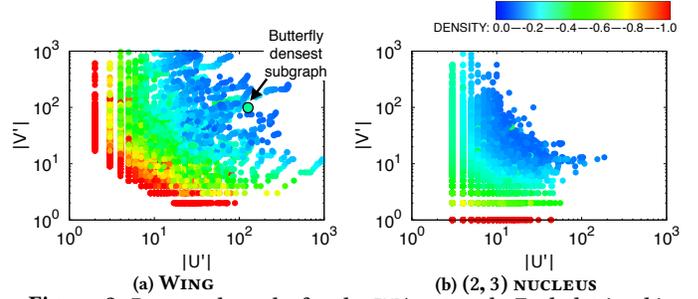


Figure 2: Dense subgraphs for the IMDb network. Each dot is a bipartite subgraph and edge density is color coded. U' and V' are the primary and secondary vertex sets and their sizes are given on the x- and y-axes. Wing decomposition results in 36 bipartite subgraphs with ≥ 0.9 edge density that have at least 10 vertices in each side, and perform competitive to the butterfly densest subgraph [31]. Other algorithms working on projections, on the right, cannot report any bipartite subgraphs in that quality.

of the top database conferences. We highlight the interesting subgraphs and hierarchies we detect that cannot be discovered by the existing approaches. Finally, we present the runtime performances.

2 BACKGROUND

This section reviews basics about the bipartite networks and the peeling algorithms. We present our notation in Table 1.

Let $G = (U, V, E)$ be an undirected, unweighted, simple (no loop, no multi-edge) bipartite graph. U is the set of primary vertices, V is the set of secondary vertices, and E is the set of edges s.t. $\forall (u, v) \in E, u \in U \wedge v \in V$. $N(u, G)$ denotes the neighbor set of a vertex u in the bipartite graph. We abuse the notation by using $N(u)$ when G is obvious. $d(u)$ is the degree of the vertex u and defined as $|N(u)|$. We define the density of a bipartite subgraph $G = (U, V, E)$ as the ratio of the number of existing edges over the number of all possible edges, i.e., $\frac{|E|}{|U| \cdot |V|}$. $H = (U', V', E')$ is an **induced subgraph** of the bipartite graph $G = (U, V, E)$, if $U' \subseteq U, V' \subseteq V$, and $V' = \cup_{u \in U'} N(u, G), E' = \cup_{u \in U'} \cup_{v \in V'} (u, v)$. $G = (U, V, E)$ is an **(a, b) -biclique** if it is a complete graph between a vertices on one side and b vertices on the other.

We present two ways to convert a bipartite graph to a unipartite graph [33, 34], also illustrated in Fig. 3a. Weighted projection is built by assigning weights to the edges. Weights are computed in proportion to the number of vertices connected to each affiliation in the bipartite graph.

DEFINITION 1. Given a bipartite graph $G = (U, V, E)$, its **weighted projection** is an edge-weighted unipartite graph $G_{wp} = (V_{wp}, E_{wp})$ s.t. $V_{wp} = U, E_{wp} = \{(u_1, u_2, w) \mid N(u_1) \cap N(u_2) \neq \emptyset \wedge w = \sum_{v \in (N(u_1) \cap N(u_2))} \frac{1}{|N(v)|}\}$. **Unweighted projection** is the same as weighted projection where the edge weights are 1.

k -core [30, 42] and k -truss [15, 38, 46, 48] subgraphs, which inspire our approach, are defined as follows:

DEFINITION 2. A connected subgraph, H , of G is a **k -core** if every vertex in H has at least degree k and no other subgraph of G that subsumes H is a k -core. **Core number** of a vertex u is the maximum k such that there is a k -core subgraph that contains u .

To find k -cores, vertices with degree $< k$ and their edges are removed from the graph, until no such vertex remains. For a full

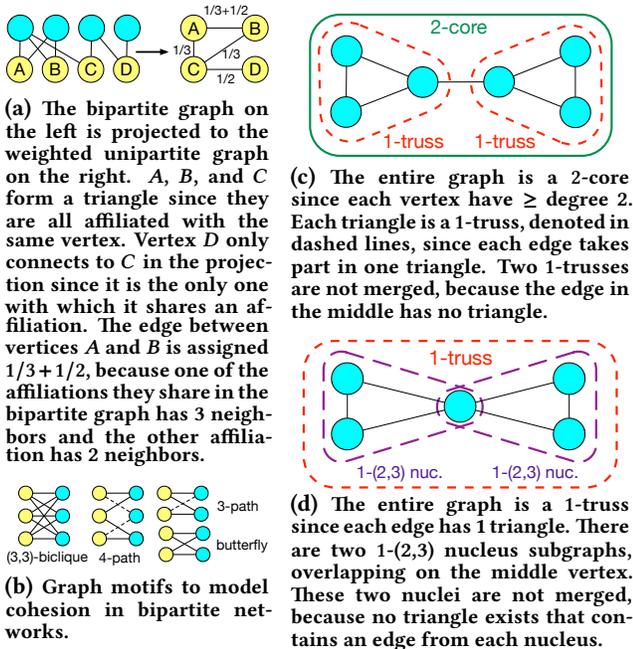


Figure 3: Examples for projections, and k -core, k -truss and k -(2,3) nucleus

decomposition, we increment k at each step, and assign k as the core number the removed vertex. This process is called as ‘peeling’, and it works in $O(|E|)$ time [6].

DEFINITION 3. A connected subgraph, H , of G is a **k -truss** if each edge in H takes part in $\geq k$ triangles and no other subgraph of G that subsumes H is a k -truss.

Nucleus decomposition is a generalization of k -core and k -truss decompositions. Instead of vertex-edge or edge-triangle relations, nucleus decomposition works on any clique relations. The idea is described in [41], but we restrict presentation to a specific case for brevity. Here we only define k -(2,3) nucleus to highlight its stronger connectedness than the k -truss. It is also referred to as ‘ k -truss community’ in [23].

DEFINITION 4. A subgraph $H=(V,E)$ of G is a **k -(2,3)-nucleus**, iff

- each edge takes part in at least k triangles;
- any pair of edges in E is connected by series of triangles;
- no other subgraph of G that subsumes H is a **k -(2,3)-nucleus**.

Here (2, 3) refers to the 2-clique (edge) and 3-clique (triangle) relations. Two edges e and f are connected by series of triangles if there exists a sequence of edges $e = e_1, e_2, \dots, e_k = f$ such that for each i , some triangle contains e_i and e_{i+1} . In Fig. 3c, entire graph is a 2-core. Two separate 1-trusses appear since the middle edge has no triangle. In Fig. 3d, each edge takes part in 1 triangle, making the entire graph 1-truss. However, two separate 1-(2, 3) nuclei exist, because there is no triangle that connect edges from each nucleus.

3 RELATED WORK

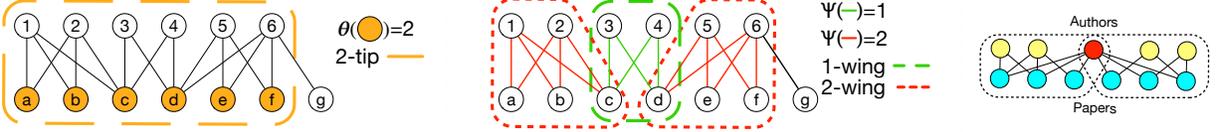
Literature on bipartite network analysis has two main thrusts: extending unipartite graph concepts to bipartite graphs and defining new projection methods to get unipartite representations. Centrality and the density metrics [9], clustering coefficients [37], matrix

partitioning [10] and clustering [47] algorithms are adapted for bipartite graphs. Regarding the projections, Newman introduced the weighted projection for scientific collaboration networks [33, 34] and Everett and Borgatti proposed to use dual projections [17]. We focus on using the actual bipartite graph (without projection) to find many dense subgraphs and their relations to each other, which have not been explored thoroughly.

Bipartite graph motifs: Capturing the smallest unit of cohesion provides a structural way to find the dense regions. Various locality patterns [9, 36, 37] and density measures [26] have been proposed for bipartite graphs (see Fig. 3b). Borgatti and Everett considered the (3, 3)-biclique to define cohesiveness [9]. Opsahl proposed the closed 4-path, which is also a (3, 3)-biclique [36]. Robins and Alexander used the (2, 2)-biclique to model the cohesion [37], and looked for 3-paths, which consists of three edges with two vertices from each set. This approach is also adopted in a recent work by Aksoy *et al.* [3] to generate bipartite graphs with community structure, where (2, 2)-biclique is called a butterfly.

Bipartite dense subgraphs: Fake likes, ratings and reviews are prevalent in online social networks and can be modeled by the dense regions in the underlying bipartite network [8]. Regarding quantifying the dense regions, Borgatti and Everett proposed the biclique as a dense subgraph [9]. Kumar *et al.* used bicliques of various sizes to analyze web graphs [24]. Enumerating all the maximal bicliques and quasi-cliques is studied by Sim *et al.* [43], and Mukherjee and Tirthapura [32]. However, biclique definition is regarded as too strict, not tolerating even a single missing edge, and is also expensive to compute. More recently, Tsourakakis *et al.* [31] used sampling to find (P, Q) -biclique densest subgraph in bipartite networks. Their algorithm results in a single subgraph that has the most number of (P, Q) -bicliques in the entire network. Butterfly ((2, 2)-biclique) densest subgraph serves as a baseline and we check how our results compare in Sec. 6. However, we do not focus on finding a single, densest subgraph, but aim to find many dense subgraphs with hierarchical relations.

Peeling bipartite networks: There have been some attempts to adapt peeling algorithms or k -core [42] like subgraphs to bipartite graphs. Cerinsek and Batagelj [11] adapted the generalized core idea [5] to bipartite networks. However, their definition is not suitable to construct a hierarchy since it is not clear how to define a comparison function for (p, q) pairs. Giatsidis *et al.* [20] worked on the scientific collaboration networks to find dense regions. They used the weighted projections and adapted the k -core decomposition for weighted networks to detect the hierarchy. They defined the fractional k -core as a maximal subgraph where every vertex has at least weight k (vertex weight is the sum of edge weights that are connected). Li *et al.* [29] introduced a k -truss like definition for bipartite networks. They insert artificial edges between vertex pairs that share a neighbor and apply the peeling algorithm on those artificial edges and their triangle counts. This is actually identical to creating the projection, and applying the k -truss decomposition using the triangle counts [15]. In addition, Chen and Saad [13] proposed algorithms to find many dense structures in bipartite graphs by constructing a dendrogram of the vertices, where the subgraphs are not overlapping. *Unlike the previous work, we introduce peeling algorithms that work directly on the bipartite network and identify possible overlapping dense regions in a hierarchy.*



(a) We check the vertices on the bottom to find k -tips. g has no butterfly. Vertices $a, b, e,$ and f take part in two butterflies while c and d are involved in three. But c and d cannot have a tip number of 3 since their induced subgraph has just one butterfly. Thus, vertices a to f forms a 2-tip. Each vertex gets the largest k -tip that they are part of; $\theta(a-f)=2$.

(b) k -wings on the same graph. Edge $(g, 6)$, $g6$ in short, has no butterfly. Each of the four edges in the middle, $c3, c4, d3, d4$, participate in only one butterfly, thus each edge has a wing number of 1 and they form a 1-wing. There are also two $(3, 2)$ -bicliques; $abc12$ and $def56$. Each edge in those bicliques takes part in two butterflies. So, each is a 2-wing, and all the edges in those have a wing number of 2. Overall, k -wings can reveal denser regions than the k -tips on the given toy graph.

(c) In an author-paper network, the author shown in red cannot be considered in a single research community because she collaborates with different researchers on distinct sets of papers. Each affiliation of the author should be considered independently to better detect the communities she is involved in.

Figure 4: Illustrations of the k -tip and k -wing subgraphs, and tip (θ) and wing (ψ) numbers.

4 DENSE BIPARTITE SUBGRAPHS

In many real-world networks, the underlying structure that yields cohesive and close-knit subgraphs is the triangle. Triangle is the smallest unit of cohesion in uni-partite networks and butterfly can be considered as the **triangle of bipartite networks** in the sense that it is the smallest cohesive bipartite unit. Conversely, a bipartite graph does not have a community structure without butterflies. *Motivated by that, we use the butterfly as the main higher-order structure in our bipartite subgraph models.* It is the smallest structure with multiple vertices at each side, and also cheaper to enumerate than the larger bicliques. Our aim is to discover the bipartite subgraphs with many butterflies and construct relations among them. k -core and k -truss decompositions reveal the hierarchical relations among dense regions and we follow a similar methodology in our models. We introduce two bipartite dense subgraph models that have different trade-offs between the subgraph density and the computation time to find.

4.1 Tip decomposition: the Definition

We introduce the k -tip to identify the vertex-induced subgraphs with many butterflies. Our approach enables building hierarchical relations among the subgraphs which results in a global tree structure that represents the significantly dense regions in the graph with various sizes and densities. k -tip measures the intensity of vertex participations in the butterfly structures. It is defined as follows:

DEFINITION 5. A bipartite subgraph $H = (U, V, E) \subseteq G$, induced on U , is a **k -tip** iff

- each vertex $u \in U$ takes part in at least k butterflies,
- each vertex pair $(u, v) \in U$ is connected by series of butterflies,
- H is maximal, i.e., there is no other k -tip that subsumes H .

Two vertices a and $b \in U$ are connected by a *series of butterflies* if there exists a sequence of vertices $a = u_1, u_2, \dots, u_k = b$ such that some butterfly contains u_i and u_{i+1} , for each i . This connectivity condition helps to separate the dense regions that are only connected by an edge, which are otherwise considered as a single combined subgraph. Butterfly density of a k -tip is obtained by ensuring a lower bound on the number of butterflies that each vertex participates. Note that a vertex with many butterflies does

not imply a dense region by itself; it should also be surrounded by other vertices incident to many butterflies.

Fig. 4a illustrates k -tip examples, where U and V cover the vertices on the bottom ($a - g$) and top ($1 - 6$), respectively. There are seven butterflies in total: $ab12, ac12, bc12, cd34, de56, df56,$ and $ef56$. Vertex g has no butterflies. Vertices c and d participate in three butterflies while the others in U (but g) are involved in two. Thus, vertices $a - f$ forms a 2-tip, shown with the dashed orange line. c and d cannot form a 3-tip by themselves or including any other vertex.

Storing all the k -tips is not convenient since they are in a hierarchy and have full overlaps. Instead, we define the **tip number** of a vertex, denoted as θ , in a similar spirit to the core number (Def. 2), to find any k -tip with a traversal operation.

DEFINITION 6. **Tip number**, $\theta(u)$, of vertex u , is the largest t such that there exists a t -tip that contains u . **Tip decomposition** of a graph $G = (U, V, E)$ is finding the tip numbers of vertices in U .

Going back to Fig. 4a, we see that all the vertices except g has tip number of 2 (in orange). A k -tip is found by starting at a vertex with the tip number of k and doing BFS-like traversal on set U . At each step, a vertex that resides in a common butterfly is visited if its tip number is at least k . Traversal continues until no such vertex appears. Set of visited vertices forms the k -tip.

4.2 Wing decomposition: the Definition

Consider the bipartite network of authors and their papers. When an author collaborates with different groups on different topics (as in Fig. 4c), k -tip cannot handle the situation since the dense regions identified by the k -tip are defined to be disjoint. To allow the vertex overlaps, which better captures the dense regions, we introduce another generic subgraph model, **k -wing**. Its definition is similar to the k -tip with one subtle difference: focus is on the edges, not vertices. Distinguishing the edges that are connected to the same vertex is the key to get overlaps on the vertices. We define the **k -wing** bipartite subgraph as follows:

DEFINITION 7. A bipartite subgraph $H = (U, V, E) \subseteq G$ is a **k -wing** if,

- every edge $(u, v) \in E$ takes part in at least k butterflies,
- each edge pair $(u_1, v_1), (u_2, v_2) \in E$ is connected by series of butterflies,

Algorithm 1: TIP DECOMPOSITION ($G = (U, V, E)$)

```
Function D2 ( $u, G$ )
   $D \leftarrow \bigcup_{v \in N(u)} \left( \bigcup_{d \in N(v)} \{d\} \right)$  // combine dist-2 neigs
   $c \leftarrow$  multiplicities of  $d \in D$  // counts of unique items
  return  $c, D$ 
// find the number of  $\bowtie$  each vertex  $u \in U$  participates
1  $D \leftarrow$  list,  $L \leftarrow$  list,  $\beta(\cdot) \leftarrow 0$ , butterfly counts  $\forall u \in U$ 
2 for each  $u \in U$  do
3    $c, D \leftarrow$  D2 ( $u, G$ ) // dist-2 neigs ( $D$ ), multiplicity ( $c$ )
4    $\beta(u) \leftarrow \sum_{d \in D} \binom{c_d}{2}$  // update  $u$ 
// find the tip numbers of vertices  $u \in U$  by peeling
5  $\theta(u) \leftarrow$  tip numbers  $\forall u \in U$ 
6 for each  $u$  with minimum  $\beta(u)$  s.t.  $\theta(u)$  is unassigned do
7    $\theta(u) \leftarrow \beta(u)$  // assign the tip number
8    $c, D \leftarrow$  D2 ( $u, G$ ) // dist-2 neigs ( $D$ ), multiplicity ( $c$ )
9   for each  $d \in D$  s.t.  $\theta(d)$  is unassigned do
// decrease by the num. of common  $\bowtie$ , if possible
10    if  $\beta(d) - \binom{c_d}{2} < \beta(u)$  then  $\beta(d) \leftarrow \beta(u)$ 
11    else  $\beta(d) \leftarrow \beta(d) - \binom{c_d}{2}$ 
12 return array  $\theta(\cdot)$ 
```

- H is maximal, i.e., there is no other k -wing that subsumes H .

Again we use the butterfly connectedness condition that is stronger than the traditional connectedness and helps to distinguish different regions that are traditionally connected, as shown in Fig. 4c. Two edges a and $b \in E$ are connected by a *series of butterflies* if there exists a sequence of edges $a = e_1, e_2, \dots, e_k = b$ such that some butterfly contains e_i and e_{i+1} for each i .

Fig. 4b presents the k -wings in the same toy graph that we checked in the previous section. Edge $(g, 6)$, $g6$ in short, does not participate in any butterfly. Each of the four edges in the middle, $c3, c4, d3, d4$, participate in only one butterfly, thus each edge has a wing number of 1 and they form a 1-wing, marked by the green region. There are also two $(3, 2)$ bicliques: $abc12$ and $def56$. Each edge in those subgraphs participates in exactly two butterflies. So, each is a 2-wing, shown in dashed red lines. Overall, k -wings can extract denser regions that cannot be seen by the k -tips. Similar to the tip numbers, we associate each edge with a number to index the subgraphs. We define the **wing number** of an edge, denoted by ψ , as follows.

DEFINITION 8. Wing number, $\psi(e)$, of an edge $e \in E$ is the largest w such that there is a w -wing that contains e . Wing decomposition is the problem of finding the wing numbers of edges in G .

In Fig. 4b, wing numbers of the edges are shown by the corresponding colors. $\psi(c3) = \psi(c4) = \psi(d3) = \psi(d4) = 1$, the green edges, which belong to the 1-wing, but not the 2-wing. All the other edges, except $g6$, has wing number of 2, in red. A k -wing can be found by a similar traversal operation explained for the k -tip. This time we traverse on edges. We start at an edge with the wing number of k and at each step we visit an edge that resides in a common butterfly and has a wing number of at least k . Traversal continues until no such edge remains and the set of visited edges forms the k -tip.

Algorithm 2: WING DECOMPOSITION ($G = (U, V, E)$)

```
// find the number of  $\bowtie$  each edge  $e \in E$  participates
1  $\beta(e) \leftarrow 0$ , butterfly counts  $\forall e \in E$ 
2 for each  $u \in U$  in order do
3   for each  $v_1, v_2$  pair  $\in N(u)$  do
4      $I \leftarrow N(v_1) \cap N(v_2)$  s.t.  $i > u \forall i \in I$ 
5     for each  $i \in I$  do
6        $\beta(e)++$ ,  $\forall e \in \bowtie(u, i, v_1, v_2)$  // each  $\bowtie$  visited
// find the wing numbers of edges  $e \in E$  by peeling
7  $\psi(e) \leftarrow$  wing numbers  $\forall e \in E$ 
8 for each  $e$  with minimum  $\beta(e)$  s.t.  $\psi(e)$  is unassigned do
9    $\psi(e) \leftarrow \beta(e)$  // assign the wing number
10   $\mathcal{B} \leftarrow$  set of  $\bowtie$ s containing  $e$ 
11  for each  $\bowtie \in \mathcal{B}$  do
12    for each edge  $f$  in  $\bowtie$  s.t.  $f \neq e$  do
13      if  $\beta(f) > \beta(e)$  then  $\beta(f) \leftarrow \beta(e)$  // neigs updated
14   $E \leftarrow E \setminus e$  // edge is removed from the graph
15 return array  $\psi(\cdot)$ 
```

5 PEELING BUTTERFLIES

In this section, we present algorithms to find the tip numbers of vertices, $\theta(\cdot)$, and the wing numbers of edges, $\psi(\cdot)$. k -tips and k -wings with the hierarchical relations can be located by the $\theta(\cdot)$ and $\psi(\cdot)$ values, as explained above.

5.1 Tip decomposition: the Algorithm

We start by counting the number of butterflies that each vertex participates in, and then apply the iterative peeling process where the tip numbers are assigned in a non-decreasing order. At each step, we find the vertex that has the minimum number of butterflies, assign its current butterfly count as the tip number, say k , and decrement the butterfly counts of the vertices (if $> k$) in U that has a common butterfly.

TIP DECOMPOSITION, presented in Algorithm 1, finds the tip numbers of the vertices in U . To locate the k -tips and construct the hierarchy, we use the disjoint-set forest heuristic that is introduced in a recent work [39]. We construct the subgraphs after finding the tip numbers of vertices, and adaptation of [39] for the tip decomposition is straightforward – we do not give the details for brevity.

Algorithm 1 has two phases. First, we determine the number of butterflies that each $u \in U$ participates, in lines 1 to 4. A simple way to count the butterflies that a vertex participates is to collect its distance-2 neighbors in a multiset by using a hashmap. If a vertex d appears $c_d > 1$ times in the multiset D (except u itself), then u and d have c common neighbors, and the number of their mutual butterflies will be $\binom{c_d}{2}$, which makes $\beta(u) = \sum_{d \in D} \binom{c_d}{2}$. This phase has $O(\sum_{v \in V} d(v)^2)$ time complexity, since a vertex $v \in V$ is accessed by each of its neighbors, and each time all the vertices in $N(v)$ are accessed. In the worst case, it is $O(|U|^2)$, where the graph is a biclique. We also used an ordering heuristic which is applied in the D2 and although it does not reduce the complexity, it enables to keep the c and D smaller, in line 3, and results in less work in the following line. We omit the details for brevity.

In the peeling process (lines 5 to 11), we assign the tip numbers of the vertices, $\theta(\cdot)$, in a non-decreasing order. We leverage a bucket data structure to efficiently retrieve the vertex with the fewest

butterflies at each step. Main operation at each step is to first assign the current butterfly number of u as its tip number, find the vertices that has a common butterfly with u , and decrement their butterfly numbers, if larger than the last assigned tip number. Since two vertices in U can have multiple mutual butterflies, we find those counts and decrease them at once. This again requires to collect the distance-2 neighbors in a multiset and we use D2 for that purpose in line 8. For each vertex that is not assigned a tip number yet, we decrease the butterfly number by $\binom{c}{2}$, where c is the cardinality of the common neighbors. However, the butterfly number cannot be less than the last assigned tip number, and we check that in line 10. Time complexity is again characterized by $O(\sum_{v \in V} d(v)^2)$ due to the D2 ($O(|U|^2)$ in the worst case). All of the additional data structures in both phases are in at most $O(|U|)$ size.

THEOREM 5.1. *Given a bipartite graph $G = (U, V, E)$, Algorithm 1 finds the tip numbers, $\theta(\cdot)$, of all $u \in U$ (Proof is omitted here, available in [40]).*

PROOF. We define that $u, v \in U$ are friend vertices if they participate in a common butterfly. $\theta(u) = k$ indicates that there are at least k butterflies which contains u and a friend vertex v s.t. $\theta(v) \geq \theta(u)$. This is enforced by the lines 10-11, where the butterfly count of a friend vertex is decreased if it will not be smaller than the tip number assigned at that step. In other words, any friend vertex with a smaller tip number does not contribute to the tip number of the vertex of interest. If Algorithm 1 finds $\theta(u) = k$ for a vertex $u \in U$, then by Def. 6, we need to show that (i) \exists a k -tip $H \ni u$, (ii) \nexists a k^+ -tip $H \ni u$ ($k^+ > k$).

(i) Once $\theta(u) = k$ is found in Algorithm 1, stop and construct an induced subgraph $H \subset G$ by traversal as follows: Initially H has only u . At each step, visit a vertex $v \in U$ s.t. v participates in a butterfly with a vertex from H . If $\theta(v)$ is unassigned yet or equal to k , add v to H . Continue the traversal until no such vertex v can be found. At the end, H is a k -tip since (1) each vertex has tip number of at least k , which means each has $\geq k$ butterflies, since the vertices are processed in the non-decreasing order of the butterfly counts (line 6), (2) all the vertices are connected to each other with butterflies thanks to the butterfly based traversal operation, and (3) H is maximal since it is the largest subgraph that can be found by the traversal.

(ii) u cannot be in a k^+ -tip. Assume it is. Then it should take part in at least k^+ butterflies and each butterfly contains a friend vertex that has at least k^+ butterflies. But, this implies that $\theta(u) = k^+$, contradiction. \square

5.2 Wing decomposition: the Algorithm

We apply a similar peeling approach to find the wing numbers of the edges in E , $\psi(\cdot)$. Instead of looking at the vertex-butterfly relations, we investigate the involvements of edges in butterflies. There are again two phases; counting the butterflies for each edge and the peeling process to find the wing numbers. Locating the k -wings and building the hierarchy is again straightforward by the disjoint-set data structures [39] and not included here for brevity.

Algorithm 2 outlines the WING DECOMPOSITION. Butterfly counting for each edge is done in lines 1 to 6. Note that it is different than the first phase of TIP DECOMPOSITION, and also more expensive, since we need to enumerate the butterflies to find the participating

edges. We compute intersections for each pair in the neighborhoods of vertices in U . We utilize a total ordering of the vertices for efficient computation. All the vertices $u \in U$ are processed in order (line 2), and in each intersection operation, we only take the vertices that succeed u (line 4). This enables to visit each butterfly only once. Total complexity is $O(\sum_{u \in U} \sum_{v_1, v_2 \in N(u)} \max(d(v_1), d(v_2)))$, and it is $O(|U||V||E|)$ in the worst case where the graph is a biclique.

In the peeling phase (lines 7 to 14), wing numbers are assigned in a non-decreasing order. Bucket structure helps to get the edge with the least number of butterflies at each step. Similar to the TIP DECOMPOSITION, we assign the updated butterfly number of the edge e as its wing number, say k , find the butterflies that e participates, and decrement the butterfly counts of the other edges in those butterflies, if greater than k . Note that there are four edges in a butterfly, thus we need to check the other three edges. Lastly, we remove the edge e from the graph. Peeling phase has $O(\sum_{(u,v) \in E} \sum_{w \in N(v)} \max(d(u), d(w)))$ time complexity, and $O(|U||V||E|)$ in the worst case. Additional space complexity is $O(|E|)$, since we store the butterfly and wing numbers for each edge.

THEOREM 5.2. *Given a bipartite graph $G = (U, V, E)$, Algorithm 2 finds the wing numbers, $\psi(\cdot)$, of all $e \in E$ (Proof is omitted here, available in [40]).*

PROOF. We define that $e, f \in E$ are friend edges if they participate in a common butterfly. $\psi(e) = k$ indicates that there are at least k butterflies which contains e and three other friend edges f, g, h s.t. $\psi(f) \geq \psi(e)$, $\psi(g) \geq \psi(e)$, and $\psi(h) \geq \psi(e)$. This is enforced by the line 13, where the butterfly count of an edge is decremented if larger than the wing number assigned at that step. In other words, any friend edge with a smaller wing number does not contribute to the wing number of the edge of interest. If Algorithm 2 finds $\psi(e) = k$ for a vertex $e \in E$, then by Def. 8, we need to show that (i) \exists a k -wing $H \ni e$, (ii) \nexists a k^+ -wing $H \ni e$ ($k^+ > k$).

(i) Once $\psi(e) = k$ is found in Algorithm 2, stop and take the remaining subgraph $H \subset G$. H is a k -wing since (1) each edge has wing number of at least k , which means each has $\geq k$ butterflies, since the edges are processed in the non-decreasing order of their butterfly counts (line 8), (2) all the edges are connected to each other with butterflies thanks to the butterfly based traversal operation, and (3) H is maximal since it is the largest subgraph that can be found by the traversal.

(ii) e cannot be in a k^+ -wing. Assume it is. Then it should take part in at least k^+ butterflies and each butterfly contains neighbor edges with at least k^+ butterflies. But, this implies that $\psi(e) = k^+$, contradiction. \square

6 EXPERIMENTS

We evaluate our algorithms on real-world unweighted simple bipartite networks from SNAP [27], ICON [14], and Konect [25]. Table 2 shows the important statistics for our dataset. condmat is the author-paper network for the arXiv preprints about condensed matter physics, published between 1995 and 1999 [35]. dbconf is another author-paper network that we constructed with the proceedings in the three top database conferences; VLDB, SIGMOD,

Table 2: Statistics for the real-world bipartite graphs (in black) and their projections (in red). First three columns show the number of primary vertices, secondary vertices, and edges for the bipartite graphs, and the fifth column is the number of butterflies. Fourth and last columns (in red) are the number of edges and triangles in the projections, which applied on U .

network	$ U $	$ V $	$ E $	$ E_p $	$ \mathbb{X} $	$ \Delta_p $
condmat	16.73 K	22.02 K	58.60 K	95.19 K	70.55 K	68.04 K
dbconf	11.19 K	8.92 K	30.72 K	84.79 K	34.55 K	95.66 K
github	56.56 K	123.35 K	440.24 K	44.56 M	50.89 M	962.55 M
marvel	6.49 K	12.94 K	96.66 K	336.53 K	10.71 M	3.26 M
IMDb	1.23 M	419.66 K	5.60 M	157.56 M	42.49 M	312.08 M
DBLP	4.00 M	1.43 M	8.65 M	315.89 M	21.04 M	9.99 B
d-label	1.75 M	270.77 K	5.30 M	> 7.20 B	3.26 B	> 167.99 T
d-style	1.62 M	0.38 K	5.74 M	> 2.56 T	77.38 B	> 699.01 Q
wiki-it	2.26 M	137.69 K	12.64 M	> 148.48 B	298.49 B	> 17.50 Q
kindle	430.53 K	1.41 M	3.21 M	93.19 M	15.51 M	10.14 B

and ICDE [28]. `github` is the network between the users and repositories in the GitHub [12]. `marvel` is the occurrence relations between the Marvel characters and the comic books [4]. IMDb links the actors and the movies they played in [1]. DBLP is the author and paper network for all the papers in the DBLP website [28]. `d-label` and `d-style` networks are obtained from an online music database [2] and the former consists of the relations between artists and the production companies, whereas the latter is the network of the artists and the styles of their albums. `wiki-it` is the edit network of the Italian Wikipedia, having the users and the pages they edit. `kindle` is the network between the Kindle books and the users who rated those books. In Table 2, second to fourth columns show the number of primary vertices, secondary vertices, and edges in each network. We assume that the primary vertices are the ones that drive the connections, which are authors, users, characters, actors, and artists. In the fifth column, the number of edges in the projected graphs (E_p) are given. We applied the projections as described in Def. 1. Last two columns are the butterfly (\mathbb{X}) counts in each bipartite network and the triangle (Δ_p) counts in the projected unipartite graph. For the `d-label`, `d-style`, and `wiki-it` networks, projections cannot be computed in 36 hours, so we give lower bounds for their edge and triangle counts, based on the degrees of the secondary vertices. Our implementation is in C++ and available¹. We used gcc 5.2.0 at -O2 optimization level. All experiments are performed on a Linux operating system running on a machine with Intel Xeon Haswell E5-2698 2.30 GHz processor with 128GB RAM.

We compared the TIPDECOMPOSITION (TIP in short, Algorithm 1) and the WINGDECOMPOSITION (WING in short, Algorithm 2) with the previous studies that find dense subgraphs with (or without) hierarchical relations in bipartite networks and or their projections.

- For the **unweighted projection**, we use two algorithms: k -core decomposition (Def. 2) and (2, 3) nucleus decomposition (Def. 4). (2, 3) nuclei subgraphs have been shown to be quite effective to find dense regions with detailed hierarchical relations [23, 41].
- For the **weighted projection**, we use the fractional k -cores [20], described in Sec. 3. To the best of our knowledge, it is the only

peeling adaptation that works on weighted networks. It is designed to handle the bipartite author-paper networks by using their weighted projections.

- Regarding the algorithms that directly focus on the **bipartite** data, Li *et al.* [29] proposed a k -truss adaptation, as explained at the end of Sec. 3. Although the focus is on the bipartite connections, their algorithm relies on inserting edges between the vertices in the same set, and computes the k -trusses on those new edges and new triangles, which is essentially the same as the (2, 3) nucleus decomposition.
- Apart from those, we also check the **(P, Q)-biclique densest subgraphs**, proposed by Mitzenmacher *et al.* [31]. We obtain the (2, 2)-biclique (butterfly) densest region for each bipartite network in our dataset.

For each bipartite subgraph, we report the size of primary and secondary vertex sets, and the edge density, i.e. $\frac{|E|}{|U| \cdot |V|}$. For the k -core, fractional k -core, and (2, 3) nucleus decompositions, we find the nuclei/cores in the projections (G_p), and then report the induced bipartite subgraphs using the vertices in those nuclei/cores.

6.1 Dense subgraph profiles

We compare the size and density of the bipartite subgraphs found by our algorithms and the previous works mentioned above. Fig. 5 (`condmat`), 7 (`marvel`), and also 2 (IMDb) (in Sec. 1) summarize the results. In all charts, each dot is a bipartite subgraph with at least 0.1 edge density. $|U|$ and $|V|$ are given on the x- and y-axes, and the density is color coded.

Overall, we observe that many dense bipartite subgraphs with nontrivial sizes (on both sides) can be obtained with WING. For IMDb and `marvel` networks, those subgraphs exhibit competitive quality (high density and large size) with respect to the butterfly densest subgraphs reported by [31]. TIP also performs well on some instances with respect to other alternatives, but not as good as the WING. As we will show in Sec. 6.4, TIP is faster than the WING, and the fair quality of the TIP can be preferred for applications with strict performance requirements. We omit the results for k -cores since they consistently have lower densities and larger sizes than the (2, 3) NUCLEUS and we observe that the densest subgraphs reported by the previous works are concentrated on the axes, meaning that they have a single vertex in either side, which is trivial.

For `condmat` network, in Fig. 5, WING yields 416 subgraphs with at least 5 authors, 5 papers, and 0.5 density. However, it cannot find a subgraph that is as good as the butterfly densest subgraph [31], which has 11 and 13 vertices on each side with more than 0.8 density. TIP can find 59 such subgraphs, whereas the (2, 3) NUCLEUS and FRACTIONAL k -CORE can only detect 14 and 20, respectively. TIP and WING are also effective to find detailed hierarchical relations; respectively, they report 77 and 164 subgraphs that contain multiple other subgraphs. (2, 3) nucleus can yield only 30 such subgraphs and the other decompositions perform worse.

`marvel` network, as seen in Fig. 7, shows more striking differences between WING and the others; 42 subgraphs appear with more than 0.7 density and at least 5 vertices on each side by using WING, while no such subgraphs can be obtained with other algorithms, including TIP. Furthermore, WING results in pretty competitive subgraphs with respect to the butterfly densest subgraph.

¹<http://sariyuze.com/bnd.tar>

6.2 Books in Amazon Kindle ratings

We analyze the ratings data for the Amazon Kindle books. The unweighted user-item bipartite graph has users on one side, items on the other side, and edges connect the users to the items they rated. The first striking difference appears in the number of reported

subgraphs: WING gives 169 distinct subgraphs that have at least 5 vertices on each side and more than 0.5 edge density. TIP reports 25 such structures whereas (2, 3) NUCLEUS only gives 6.

One example group that WING identifies while all the others cannot is a set of 38 books that are mostly on the self-improvement

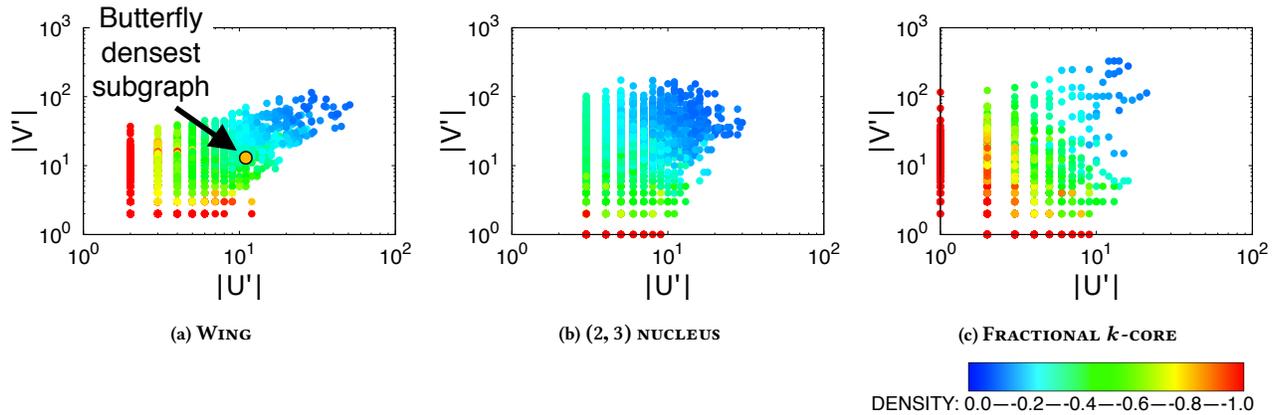


Figure 5: Dense subgraphs in condmat. Each dot is a bipartite subgraph, the density, $|E'|/(|U'||V'|)$, is color coded, and $|U'|$ and $|V'|$ are given on the x- and y-axes. WING results in 416 subgraphs with ≥ 0.5 density and at least 5 vertices in each side. Although it has many subgraphs with different qualities, none is as good as the butterfly densest subgraph, reported by [31].

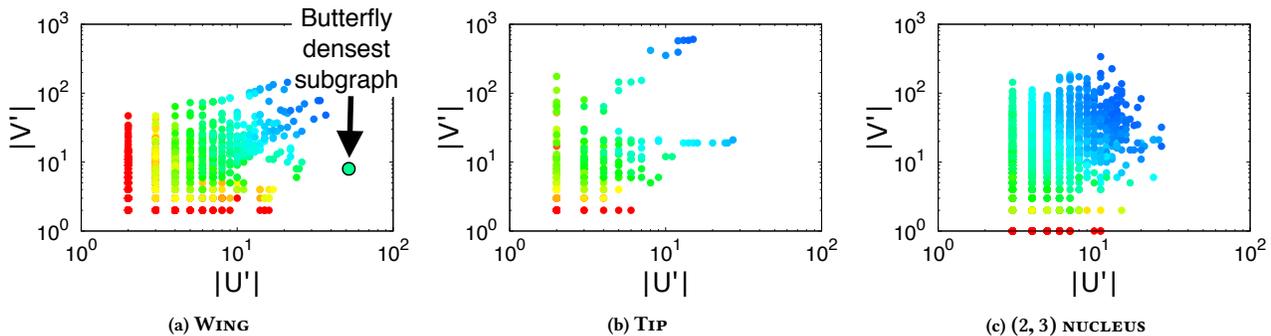


Figure 6: dbconf network. Most dense structures in (2, 3) nuclei and fractional k -cores have only one vertex in either vertex set (the red dots along x- and y- axes). Those subgraphs represent the collaborations of many authors in a single paper. We observe that they are mostly papers on a software-product authored by a large group in a company. In most cases those authors do not have any other papers, which makes the subgraph less informative.

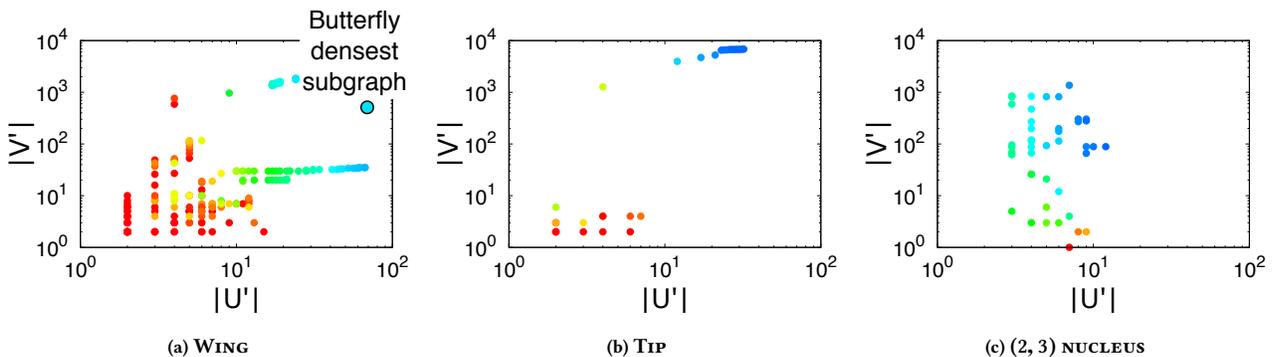


Figure 7: Dense subgraphs for marvel network. WING provides many subgraphs with ≥ 0.5 density. 57 of those have at least 5 vertices in each side and 11 have 10 vertices in each. No other algorithm can get such subgraphs.

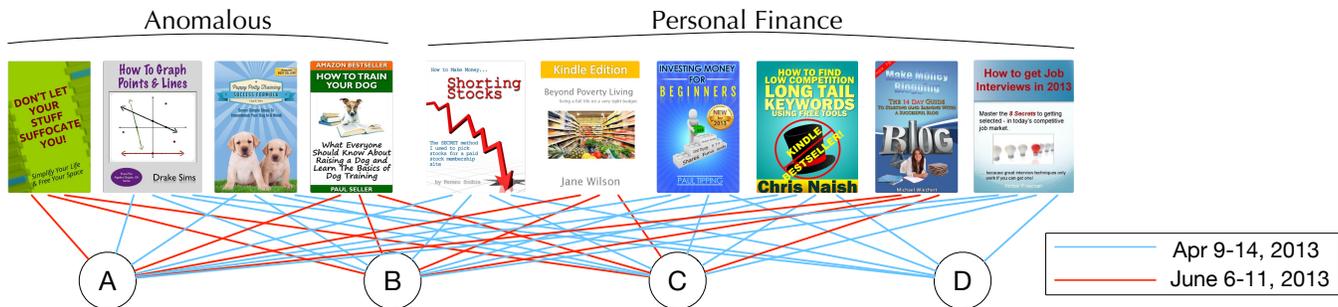


Figure 8: 10 books rated by 4 users, reported by WING. 6 books on the right are on personal finance whereas the 4 on the left are unrelated. Further investigation reveals that those 4 users rated the books in the same days and always gave 5 stars, which are strong indicators for the fake reviews.

theme (“how to” books, guides on relationships, healthy diets and personal finance) rated by 12 users. Furthermore, there are three smaller groups within this group. The first has 6 books and all are rated by 5 users. Those books are about the relationships between spouses, like “*Finding The Ins And Outs Of Relationships*”. Second group is a collection of 20 books on guides for healthy diets and self-meditation. The third group, shown in Fig. 8, reveals a different picture. There are 10 books that are rated by 4 users, and 6 of them are about the personal finance. However, the other 4 are totally unrelated; 2 on dog training, 1 on the elementary-level algebra and 1 is about creating more space at home. Checking the rating scores for those shows that that almost all the ratings are 5 stars. Furthermore, the dates those books are rated are very close: 2 books are rated between April 9 and 14 in 2013, and the other 2 are between June 6 and 11 in the same year. Bursty reviews is an important indicator for the fake reviews, as [19] notes that “*Reviewers and reviews appearing in a burst are often related in the sense that spammers tend to work with other spammers and genuine reviewers tend to appear together with other genuine reviewers*”. Overall, WING can reveal the sets of books on the same theme and also point to the anomalous behaviors, like the fake reviews, in bipartite networks. It does this by using only the graph topology *without any metadata*.

6.3 Authors in the top database conferences

Here we highlight some interesting subgraphs and hierarchical structures found in the dbconf network. We report on what WING and TIP can find but others cannot, and also what we cannot find which can be identified by other algorithms.

Fig. 6 shows the dense subgraph profiles for the dbconf network. Trends are similar to the other graphs explained above. WING finds dense subgraphs with non-trivial sizes and also vying with the butterfly densest subgraph, TIP cannot perform as good as the WING, and other algorithms mainly result in dense regions with a single vertex on either side, red dots along x- and y-axes. In (2, 3) NUCLEUS case, the red dots on the x-axis mostly correspond to the papers about a software product of a company and authored by a large group of researchers. In most cases those authors do not have any other papers, thus no butterfly structures exist around them and cannot be found by our WING and TIP algorithms. One example is the paper entitled “*Comdb2: Bloomberg’s Highly Available Relational Database System*” in ICDE’10 is written by a large group of people in Bloomberg LP. On the other hand, in the FRACTIONAL

k-CORE case, there are also red dots appearing along the y-axis. Those are the subgraphs with a single author and many papers. However, they are isolated because the fractional *k*-core computation assigns a large weight to this vertex and there is no other vertex around with a close weight. Divesh Srivastava² is one such prolific researcher appearing on the y-axis with 144 papers. But, there is no other dense subgraph in fractional *k*-cores that contains him, showing the weakness of the projection-based approach. **Note that, none of those trivial subgraphs are reported by the TIP or WING algorithms, since no butterflies exist.** We also identified some structures that can be *only identified by the TIP and/or WING algorithms*.

Philip Yu and IBM Research: We also checked the densest subgraphs that contain a prolific researcher who is expected to participate in multiple structures. We choose Philip Yu³ for this purpose. Tracing the hierarchical relations from his subgraph(s) in the TIP and WING results gives interesting information about the Streams group at IBM Research, denoted in Fig. 9. The bottom branch in the figure can be identified by both TIP and WING whereas all other projection-based methods fail. The densest subgraph on the right-most is a biclique, including Kun-Lung Wu, manager at IBM Research, and P. Yu, who used to work in the same group, and 13 papers they co-authored. They are contained in another subgraph which has 0.82 density and includes Buğra Gedik (ex-staff member) in addition to K. L. Wu and P. Yu. Looking at another parent subgraph in the chain, we find Ling Liu (B. Gedik’s Ph.D. advisor) joins B. Gedik, K. L. Wu and P. Yu. The further parent subgraphs in the chain reveals other researchers, H. Andrade and G. Jacques-Silva, who used to work in the same group. WING also reports another branch that contains P. Yu. There is a biclique of Charu Aggarwal (staff member), P. Yu and their 14 papers. Its parent subgraph has two more authors, Joel Wolf and Daniel Dias (staff members), and the further parent subgraphs in the chain include other people at the same institution. This chain and the other chain described above merges in a subgraph that has 76 authors, who are the researchers and ex-interns in the Streams group.⁴

One weakness observed in the TIP and WING is that they tend to identify pairs of authors first, and then enlarge those

²<http://dblp.uni-trier.de/pers/hd/s/Srivastava:Divesh>

³<https://www.cs.uic.edu/PSYu/>

⁴All the information is obtained from the websites of the aforementioned people.

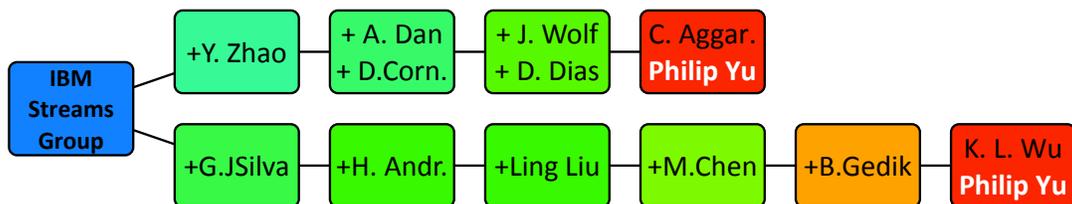


Figure 9: The subgraphs reported by WING in the dbconf network that contains Philip Yu. Densities are color coded. Two chains show the collaborations in the IBM Research’s Streams group, where the parent subgraph (left-most) contains 76 researchers that contains staff members and intern students.

pairs to larger subgraphs. This results in long chains in the hierarchy which needs further filtering to extract large and dense structures. Main reason for this behavior is the quadratic increase in the number butterflies for author pairs with common papers, i.e., n common neighbors gives $\binom{n}{2}$ butterflies.

6.4 Runtime performances

Lastly we check the runtimes. Table 3 presents the results. As pointed in Sec. 5, WING does more work than TIP and it is also verified by the experiments; TIP can be up to 6 times faster. (2, 3) NUCLEUS gives the most significant subgraphs among the projection based algorithms, however it has $O(|\Delta|)$ complexity on the projected graph and suffers from the explosion in the number of triangles. Note that, a k -clique is created for each secondary vertex with degree k , and the number of triangles increases **cubically**, for instance d-style network has less than 2M vertices and around 5.5M edges, but its projection has more than 699 quadrillion triangles, which is impossible to process in a reasonable time. Thus, WING is orders of magnitude faster than the (2, 3) NUCLEUS. Regarding the densest butterfly subgraph [31] runtimes, we run [31] without sampling for all the graphs, except d-label, d-style, and wiki-it, and show the runtimes for butterfly enumeration and maximum flow computation phases. Runtimes are mostly dominated by the maximum flow computation and significantly larger than the WING runtimes. For d-label, d-style, and wiki-it graphs, the ones with the most butterflies, we set the sampling probabilities such that the maximum flow operation is run on 50M butterflies (shown in red in Table 3). We observe that the butterfly enumeration, which does not depend on the sampling, takes the most time and it can be up to 33% of the total WING time. For the wiki-it graph, it takes more than 24K seconds to report a single densest butterfly subgraph while 97K seconds is enough for the WING to find many in a large spectrum.

7 DISCUSSION

Our algorithms for bipartite networks can find many dense substructures with the hierarchical relations. Butterfly based definitions enable us to extract the meaningful regions in the graph and the use cases in Sec. 6.3 and 6.2 also verify this. One weakness that we observed is that, number of butterflies per vertex or edge can quadratically increase when there is a large biclique. This results in dense but smaller subgraphs focusing around a few vertices. Handling those regions is an interesting direction. Another promising direction is to find efficient heuristics to count and enumerate the butterflies and other small bicliques. There is a huge body of work

Table 3: Runtimes of all the algorithms in seconds. WING is orders of magnitude faster than the (2, 3) NUCLEUS due to the increase in the number of triangles in the projection. CORE and FRACTIONAL CORE are pretty fast, but not effective to report dense structures.

(in sec.)	bipartite		unipartite (projection)			densest \times [31]	
	TIP	WING	CORE	FR. CORE	(2, 3) NUC.	enum	max flow
github	4.09	35.27	0.43	0.68	121,643	13.89	985
IMDb	11.97	72.42	3.35	3.74	8,831	21.78	668
DBLP	25.63	11.04	5.85	10.30	9,378	3.96	206
d-label	1,328	3,789	> 36hrs	> 36hrs	> 36hrs	1,266	1,146
d-style	22,488	25,758	> 36hrs	> 36hrs	> 36hrs	7,264	833
wiki-it	63,187	97,052	> 36hrs	> 36hrs	> 36hrs	23,564	859
kindle	7.19	33.09	1.80	2.02	29,057	10.06	165

on triangle counting and enumeration, but studies are immature for the bipartite structures. Indeed, butterflies can easily reach to billions in a bipartite graph with a few million edges, whereas that many triangles in unipartite networks can be observed for hundreds of millions of edges.

Acknowledgements: Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-NA-0003525. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

REFERENCES

- [1] 2016. IMDb. (2016). (www.imdb.com/interfaces).
- [2] 2017. Konec network dataset. (2017). (<http://www.discogs.com/>).
- [3] S. Aksoy, T. G. Kolda, and A. Pinar. 2017. Measuring and Modeling Bipartite Graphs with Community Structure. *Journal of Complex Networks* 5, 4 (2017), 581–603.
- [4] R. Alberich, J. Miro-Julia, and F. Rosselló. 2002. Marvel Universe looks almost like a real social network. *CoRR abs/cond-mat/0202174* (2002).
- [5] Vladimir Batagelj and Matjaz Zaversnik. 2002. Generalized Cores. *CoRR cs.DS/0202039* (2002).
- [6] V. Batagelj and M. Zaversnik. 2003. An $O(m)$ Algorithm for Cores Decomposition of Networks. *CoRR cs/0310049* (2003).
- [7] A. R. Benson, D. F. Gleich, and J. Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166.
- [8] A. Beutel, W. Xu, V. Guruswami, C. Palow, and C. Faloutsos. 2013. CopyCatch: Stopping Group Attacks by Spotting Lockstep Behavior in Social Networks. In *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*. 119–130.
- [9] S. P. Borgatti and M. G. Everett. 1997. Network analysis of 2-mode data. *Social Networks* 19, 3 (1997), 243 – 269.
- [10] Ü. V. Çatalyürek and C. Aykanat. 1999. Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication. *IEEE Transactions on*

- Parallel and Distributed Systems* 10, 7 (1999), 673–693.
- [11] M. Cerinsek and V. Batagelj. 2015. Generalized two-mode cores. *Social Networks* 42 (2015), 80 – 87.
 - [12] S. Chacon. 2009. The 2009 GitHub Contest. (2009). (github.com/blog/466-the-2009-github-contest).
 - [13] J. Chen and Y. Saad. 2012. Dense Subgraph Extraction with Application to Community Detection. *IEEE Transactions on Knowledge & Data Engineering* 24, 7 (2012), 1216–1230.
 - [14] A. Clauset, E. Tucker, and M. Sainz. 2016. The Colorado Index of Complex Networks. (2016). (icon.colorado.edu).
 - [15] J. Cohen. 2008. Trusses: Cohesive subgraphs for social network analysis. National Security Agency Technical Report (2008).
 - [16] I.S. Dhillon. 2001. Co-clustering Documents and Words Using Bipartite Spectral Graph Partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*. 269–274.
 - [17] M.G. Everett and S.P. Borgatti. 2013. The dual-projection approach for two-mode networks. *Social Networks* 35, 2 (2013), 204 – 210.
 - [18] D. C. Fain and J. O. Pedersen. 2006. Sponsored search: A brief history. *Bulletin of the American Society for Information Science and Technology* 32, 2 (2006), 12–13.
 - [19] G. Fei, A. Mukherjee, B. Liu, M. Hsu, M. Castellanos, and R Ghosh. 2013. Exploiting Burstiness in Reviews for Review Spammer Detection. In *ICWSM*. The AAAI Press.
 - [20] C. Giatsidis, D. M. Thilikos, and M. Vazirgiannis. 2011. Evaluating Cooperation in Communities with the k -Core Structure. In *ASONAM*. 87–93.
 - [21] D. Gibson, R. Kumar, and A. Tomkins. 2005. Discovering Large Dense Subgraphs in Massive Graphs. In *VLDB*. 721–732.
 - [22] E. Gregori, L. Lenzini, and C. Orsini. 2011. k -dense communities in the internet AS-level topology. In *International Conf. on Communication Systems and Networks (COMSNETS)*. 1–10.
 - [23] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. 2014. Querying K-truss Community in Large and Dynamic Graphs. In *Proc. of the ACM SIGMOD International Conf. on Management of Data*. 1311–1322.
 - [24] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. 1999. Trawling the Web for Emerging Cyber-communities. In *WWW*. 1481–1493.
 - [25] Jerome Kunegis. 2017. Konect network dataset. (2017). (<http://konect.uni-koblenz.de>).
 - [26] M. Latapy, C. Magnien, and N. Del Vecchio. 2008. Basic notions for the analysis of large two-mode networks. *Social Networks* 30, 1 (2008), 31 – 48.
 - [27] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. (June 2014). (snap.stanford.edu/data).
 - [28] Michael Ley. 2016. DBLP computer science bibliography. (Sept. 2016). (dblp.uni-trier.de).
 - [29] Y. Li, T. Kuboyama, and H. Sakamoto. 2013. Truss Decomposition for Extracting Communities in Bipartite Graph. In *IMMM 2013 : The Third International Conference on Advances in Information Mining and Management*.
 - [30] D. Matula and L. Beck. 1983. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of ACM* 30, 3 (1983), 417–427.
 - [31] M. Mitzenmacher, J. Pachocki, R. Peng, C. Tsourakakis, and S. C. Xu. 2015. Scalable Large Near-Clique Detection in Large-Scale Networks via Sampling. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. 815–824.
 - [32] A. P. Mukherjee and S. Tirthapura. 2014. Enumerating Maximal Bicliques from a Large Graph Using MapReduce. In *Proceedings of the 2014 IEEE International Congress on Big Data (BIGDATAACONGRESS '14)*. 707–716.
 - [33] M. E. J. Newman. 2001. Scientific collaboration networks. I. Network construction and fundamental results. *Phys. Rev. E* 64 (2001), 016131. Issue 1.
 - [34] M. E. J. Newman. 2001. Scientific collaboration networks. II. Shortest paths, weighted networks, and centrality. *Phys. Rev. E* 64 (2001), 016132. Issue 1.
 - [35] M. E. J. Newman. 2001. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences* 98, 2 (2001), 404–409.
 - [36] T. Opsahl. 2013. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks* 35, 2 (2013), 159 – 167.
 - [37] G. Robins and M. Alexander. 2004. Small Worlds Among Interlocking Directors: Network Structure and Distance in Bipartite Graphs. *Computational & Mathematical Organization Theory* 10, 1 (2004), 69–94.
 - [38] K. Saito and T. Yamada. 2006. Extracting Communities from Complex Networks by the k -dense Method. In *IEEE International Conf. on Data Mining Workshops, ICDMW*. 300–304.
 - [39] A. E. Sarıyüce and A. Pinar. 2016. Fast Hierarchy Construction for Dense Subgraphs. *Proc. VLDB Endow.* 10, 3 (Nov. 2016), 97–108.
 - [40] A. E. Sarıyüce and A. Pinar. 2017. Peeling Bipartite Networks for Dense Subgraph Discovery. *CoRR* 1611.02756 (2017). (Extended version).
 - [41] A. E. Sarıyüce, C. Seshadhri, A. Pinar, and Ü. V. Çatalyürek. 2015. Finding the Hierarchy of Dense Subgraphs Using Nucleus Decompositions. In *Proc. of the International Conf. on World Wide Web (WWW)*. 927–937.
 - [42] S. B. Seidman. 1983. Network structure and minimum degree. *Social Networks* 5, 3 (1983), 269–287.
 - [43] K. Sim, J. Li, V. Gopalkrishnan, and G. Liu. 2009. Mining maximal quasi-bicliques: Novel algorithm and applications in the stock market and protein networks. *Statistical Analysis and Data Mining* 2, 4 (2009), 255–273.
 - [44] C. Tsourakakis. 2015. The K -clique Densest Subgraph Problem. In *Proc. of the 24th International Conf. on World Wide Web (WWW '15)*. 1122–1132.
 - [45] C. E. Tsourakakis, J. Pachocki, and M. Mitzenmacher. 2017. Scalable Motif-aware Graph Clustering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. 1451–1460.
 - [46] A. Verma and S. Butenko. 2012. Network clustering via clique relaxations: A community based approach. In *Graph Partitioning and Clustering, DIMACS Workshop*. 129–140.
 - [47] M. M. Wolf, A. M. Klinvex, and D. M. Dunlavy. 2016. Advantages to Modeling Relational Data using Hypergraphs versus Graphs. In *IEEE High Performance Extreme Computing Conference, HPEC*.
 - [48] Y. Zhang and S. Parthasarathy. 2012. Extracting Analyzing and Visualizing Triangle K -Core Motifs Within Networks. In *Proc. of the IEEE International Conf. on Data Engineering (ICDE)*. 1049–1060.