Noname manuscript No.
(will be inserted by the editor)

Finding Lasting Dense Subgraphs

Konstantinos Semertzidis [1] · **Evaggelia Pitoura** [1] · **Evimaria Terzi** [2] · **Panayiotis Tsaparas** [1]

Received: date / Accepted: date

Abstract Graphs form a natural model for relationships and interactions between entities, for example, between people in social and cooperation networks, servers in computer networks, or tags and words in documents and tweets. But, which of these relationships or interactions are the most lasting ones? In this paper, we study the following problem: given a set of graph snapshots, which may correspond to the state of an evolving graph at different time instances, identify the set of nodes that are the most densely connected in all snapshots. We call this problem the *Best Friends Forever* (BFF) problem. We provide definitions for density over multiple graph snapshots, that capture different semantics of connectedness over time, and we study the corresponding variants of the BFF problem. We then look at the *On-Off* BFF (O^2BFF) problem that relaxes the requirement of nodes being connected in all snapshots, and asks for the densest set of nodes in at least k of a given set of graph snapshots. We show that this problem is NP-complete for all definitions of density, and we propose a set of efficient algorithms. Finally, we present experiments with synthetic and real datasets that show both the efficiency of our algorithms and the usefulness of the BFF and the O^2BFF problems.

Keywords Dense Subgraphs · Graph History · Social Networks

1 Introduction

Graphs offer a natural model for capturing the interactions and relationships among entities. Oftentimes, multiple snapshots of a graph are available; for example, these snapshots may correspond to the states of a dynamic graph at different time instances, or the states of a complex system at different conditions. We call such sets of graph snapshots, a *graph history*. Analysis of the graph history finds

[1]Dept. of Computer Science and Engineering, University of Ioannina, Greece
E-mail: {ksemer,pitoura,tsap}@cs.uoi.gr

[2]Dept. of Computer Science, Boston University, USA
E-mail: evimaria@cs.bu.edu

a large spectrum of applications, ranging from social-network marketing, to virus propagation and digital forensics. A central question in this context is: *which interactions, or relationships in a graph history are the most lasting ones?* In this paper, we formalize this question and we design algorithms that effectively identify such relationships.

In particular, given a graph history, we introduce the problem of efficiently finding the set of nodes, that remains the most tightly connected through history. We call this problem the *Best Friends Forever* (BFF) problem. We formulate the BFF problem as the problem of locating the set of nodes that have the maximum *aggregate density* in the graph history. We provide different definitions for the aggregate density that capture different notions of connectedness over time, and result in four variants of the BFF problem.

We then extend the BFF problem to capture the cases where subsets of nodes are densely connected for only a subset of the snapshots. Consider for example, a set of collaborators that work intensely together for some years and then they drift apart, or, a set of friends in a social network that stop interacting for a few snapshots and then, they reconnect with each other. To identify such subsets of nodes, we define the *On-Off* BFF problem, or O^2BFF for short. In the O^2BFF problem, we ask for a set of nodes and a set of k snapshots such that the aggregate density of the nodes over these snapshots is maximized.

The BFF and the O^2BFF problems find many applications. For example, in collaboration and social networks, the nodes that belong to lasting dense subgraphs correspond to well-acquainted individuals. Such individuals can be chosen to form teams, or organize professional or social events, since usually the success of such events depends on whether the participants are well-acquainted with each other. Identifying groups of collaborators or friends may also help in improving our understanding of such networks. For example, using the *DBLP* co-authorship graph, we were able to identify lasting collaborations among authors in database and data mining conferences. In particular, for a specific definition of aggregate density, we identified a group of authors that although there was no paper in which they are all co-authors, they have co-authored papers with each other in many snapshots.

Furthermore, in a network where nodes are words or tags and edges correspond to their co-occurrences in documents or microposts published during a specific period of time, identifying BFF nodes may serve as a first step in topic identification, tag recommendation and other types of analysis. For example, using a *Twitter* dataset of tweets published in a period of two weeks, we were able, by locating O^2BFFs, to identify both trending topics and the dates within these two weeks when these topics were popular. The topics we discovered correspond to real events that attracted attention world-wide. Yet another application of BFFs is in computer networks. For instance, locating servers that communicate heavily over time may be useful in identifying potential attacks, or bottlenecks. Finally, there are many applications in biological networks. For example, in a protein-interaction network, one could apply the BFF problem to locate protein complexes that are densely interacting at different states, thus indicating a possible underlying regulatory mechanism.

The problem of identifying a dense subgraph in a static (i.e., single-snapshot) graph has received a lot of attention (e.g., Charikar (2000); Goldberg (1984); Khuller and Saha (2009)). There has been also work on finding dense subgraphs

in dynamic graphs (e.g., Epasto et al. (2015)). However, in this line of work, the goal is to efficiently locate the densest subgraph in the current graph snapshot, whereas we are interested in locating subgraphs that remain dense in the whole graph history. To the best of our knowledge, we are the first to systematically introduce and study density in a graph history, and define the BFF and O^2BFF problems. The most related work to ours is by Jethava and Beerenwinkel (2015) where they study just one of the four variants of the BFF problem in the context of graph databases. We compare the performance of our algorithms for this variant with the algorithm proposed by Jethava and Beerenwinkel (2015) experimentally.

We study the complexity of the different variants of the BFF and O^2BFF problems. Two of the BFF variants can be solved optimally, while the O^2BFF is NP-hard. We propose a generic algorithmic framework for solving our problems, that works in linear time. Experimental results with real and synthetic datasets show the efficiency and effectiveness of our algorithms in discovering lasting dense subgraphs. Two case studies on bibliographic collaboration networks, and hashtag co-occurrence networks in *Twitter* validate our approach.

To summarize, the main contributions of this work are the following:

- We introduce the novel BFF and O^2BFF problems of identifying a subset of nodes that define dense subgraphs in a graph history. To this end, we extend the notion of density for graph histories, and provide definitions that capture different semantics of density over time leading to four variants of our problems.
- We study the complexity of the variants of the BFF and O^2BFF problems and propose appropriate algorithms. We prove the optimality, or the approximation factor of our algorithms whenever possible.
- We perform experiments with both real and synthetic datasets and demonstrate that our problem definitions are meaningful, and that our algorithms work well in identifying dense subgraphs in practice.

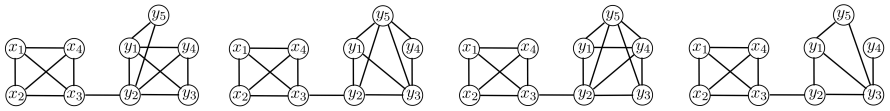
Roadmap: In Section 2, we provide definitions of aggregate density. We introduce the BFF problem and its algorithms in Section 3, and the O^2BFF problem and its algorithms in Section 4. Our experimental evaluation is presented in Section 5 and comparison with related work in Section 6. Section 7 concludes the paper.

2 Aggregate density

We assume that we are given as input multiple graph snapshots over the same set of nodes. Snapshots may be ordered, for example, when the snapshots correspond to the states of a dynamic graph. We may also have an unordered collection of graphs, for example, when the snapshots correspond to graphs collected as a result of some scientific experiments. We refer to such a graph collection as a *graph history*.

Definition 1 (Graph History) A graph history $\mathcal{G} = \{G_1, G_2, \ldots, G_\tau\}$ is a collection of τ graph snapshots, where each snapshot $G_t = (V, E_t)$, $t \in [1, \tau]$, is defined over the same set of nodes V.

An example of a graph history with four snapshots is shown in Figure 1. Note that our definition is applicable to graph snapshots with different set of nodes by considering V as their union.



(a) G_1 (b) G_2 (c) G_3 (d) G_4

Fig. 1: A graph history $\mathcal{G} = \{G_1, \ldots, G_4\}$ consisting of four snapshots.

We will now define the notion of density of a set of nodes in a graph history. We start by reviewing two basic definitions of graph density of a set of nodes in a single graph snapshot (Charikar (2000)). Given an undirected graph $G = (V, E)$ and a node u in V, let *degree*(u, G) denote the degree of u in G. Let $S \subseteq V$ be a subset of nodes in the graph $G = (V, E)$, and let $G[S] = (S, E(S))$ in G be the induced subgraph for the set S, where $E(S) = \{(u, v) \in E : u \in S, v \in S\}$. We define the *average density*, $d_a(S, G)$, of the set S to be the average degree of the nodes in S, in the induced subgraph $G[S]$:

$$d_a(S, G) = \frac{1}{|S|} \sum_{u \in S} \textit{degree}(u, G[S]) = \frac{2|E(S)|}{|S|}$$

We define the *minimum density*, $d_m(S, G)$, of the set S to be the minimum degree of any node in S, in the induced subgraph $G[S]$:

$$d_m(S, G) = \min_{u \in S} \textit{degree}(u, G[S]).$$

Intuitively, for a given set of nodes S and the connections between them in $E(S)$, d_m is defined by a single node, the one that is least connected in the induced subgraph, while d_a looks at the average connectivity of the nodes in S. For example, for snapshot G_1 in Figure 1, for $S_x = \{x_1, x_2, x_3, x_4\}$, $d_m(S_x, G_1) = d_a(S_x, G_1) = 3$, while for $S_y = \{y_1, y_2, y_3, y_4, y_5\}$, $d_m(S_y, G_1) = 2$ and $d_a(S_y, G_1) = 16/5$. Between S_x and S_y, S_x has the highest minimum density, whereas S_y the highest average density. Clearly, d_m is a lower bound for d_a. From now on, when the subscript of d is ignored, density can be either d_a or d_m. Abusing the notation, we will sometimes use $d(G[S])$ to denote the density $d(S, G)$ of S in G.

To define the density of a set of nodes S on a graph history, we need a way to aggregate the density of a set of nodes over multiple graph snapshots.

Aggregating density sequences: Given a graph history $\mathcal{G} = \{G_1, \ldots, G_\tau\}$, we will use $d(S, \mathcal{G}) = \{d(S, G_1), \ldots, d(S, G_\tau)\}$ to denote the sequence of density values for the graphs induced by the set S in the graph snapshots. We consider two definitions for an *aggregation function* $g(d(S, \mathcal{G}))$ that aggregates the densities over snapshots: The first, g_m, computes the minimum density over all snapshots:

$$g_m(d(S, \mathcal{G})) = \min_{G_t \in \mathcal{G}} d(S, G_t).$$

The second, g_a, computes the average density over all snapshots:

$$g_a(d(S, \mathcal{G})) = \frac{1}{|\mathcal{G}|} \sum_{G_t \in \mathcal{G}} d(S, G_t).$$

Intuitively, the minimum aggregation function requires high density in each and every snapshot, while the average aggregation function looks at the snapshots as a whole. Again, we use g to collectively refer to g_m or g_a. We can now define the *aggregate density* f.

Definition 2 (Aggregate Density) Given a graph history $\mathcal{G} = \{G_1, \ldots, G_\tau\}$ defined over a set of nodes V and $S \subseteq V$, we define the *aggregate density* $f(S, \mathcal{G})$ to be $f(S, \mathcal{G}) = g(d(S, \mathcal{G}))$. Depending on the choice of the density function d and the aggregation function g, we have the following four versions of f: (a) $f_{mm}(S, \mathcal{G}) = g_m(d_m(S, \mathcal{G}))$, (b) $f_{ma}(S, \mathcal{G}) = g_m(d_a(S, \mathcal{G}))$, (c) $f_{am}(S, \mathcal{G}) = g_a(d_m(S, \mathcal{G}))$, and (d) $f_{aa}(S, \mathcal{G}) = g_a(d_a(S, \mathcal{G}))$.

Each density definition associates different semantics to density among nodes in a graph history. Large values of $f_{mm}(S, \mathcal{G})$ correspond to groups of nodes S where each member of the group is connected with a large number of other members of the group at each snapshot. A group ceases to be considered dense if a single node loses touch with the other members in the group, even for a single snapshot.

Large values of $f_{ma}(S, \mathcal{G})$ are achieved for groups with high average density at each snapshot $G \in \mathcal{G}$. Contrary to $f_{mm}(S, \mathcal{G})$, where the requirement is placed at each member of the group, large values of $f_{ma}(S, \mathcal{G})$ are indicative that the group S has persistently high density as a whole.

The $f_{am}(S, \mathcal{G})$ metric takes the average in time of the minimum degree of the nodes in group S, thus is less sensitive to the density of S at a single snapshot.

Lastly, the $f_{aa}(S, \mathcal{G})$ metric takes large values when the group S has many connections on average; thus, f_{aa} is more "loose" both in terms of consistency over time and in terms of requirements at the individual group member level.

For example, take S_x and S_y in the graph history $\mathcal{G}$ in Figure 1. All aggregate densities for S_x are equal to 3. However, for S_y, $f_{aa}(S_y, \mathcal{G}) = 31/10$, while $f_{ma}(S_y, \mathcal{G}) = 12/5$. That is, while $f_{aa}(S_y, \mathcal{G}) > f_{aa}(S_x, \mathcal{G})$, $f_{ma}(S_y, \mathcal{G}) < f_{ma}(S_x, \mathcal{G})$ due to the last instance. Note also, that $f_{mm}(S_y, \mathcal{G}) = 1$ due to just one node in just one snapshot, i.e., node y_4 in the last snapshot, while $f_{am}(S_y, \mathcal{G}) = 2$.

The average graph: Finally, let us define the *average graph* of a graph history $\mathcal{G}$ which is an edge-weighted graph where the weight of an edge is equal to the fraction of snapshots in $\mathcal{G}$ where the edge appears.

Definition 3 (Average Graph) Given a graph history $\mathcal{G} = \{G_1, \ldots, G_\tau\}$ on a set of nodes V, the average graph $\widehat{H}_\mathcal{G} = (V, \widehat{E}, \widehat{w})$ is a *weighted, undirected* graph on the set of nodes V, where $\widehat{E} = \cup_{i=1}^{\tau} E_i$, and for each $(u, v) \in \widehat{E}$, $\widehat{w}(u, v) = \frac{|G_t = (V, E_t) \in \mathcal{G} | (u,v) \in E_t|}{|\mathcal{G}|}$.

As usual, the degree of a node u in a weighted graph is defined as: $\mathit{degree}(u, \widehat{H}_\mathcal{G}) = \sum_{(u,v) \in \widehat{E}} \widehat{w}(u, v)$. The average graph performs aggregation on a per-node basis, in that, the degree of each node u in $\widehat{H}_\mathcal{G}$ is the average degree of u in time. With the average graph, we lose information regarding density at individual snapshots. With some algebraic manipulation, we can prove the following lemma that shows a connection between the average graph and the f_{aa} density function:

Lemma 1 *Let $\mathcal{G} = \{G_1, \ldots, G_\tau\}$ be a graph history over a set of nodes V and S a subset of nodes in V, it holds: $f_{aa}(S, \mathcal{G}) = d_a\left(S, \widehat{H}_\mathcal{G}\right)$.*

3 The BFF problem

In this section, we introduce the BFF problem, we study its hardness and propose appropriate algorithms.

3.1 Problem definition

Given the snapshots of a graph history $\mathcal{G}$, our goal is to identify a subset of nodes $S \subseteq V$ (the Best Friends Forever (BFF) set) that are densely connected in the graph history $\mathcal{G}$. Formally:

Problem 1 (The Best Friends Forever (BFF) Problem) Given a graph history $\mathcal{G}$ and an aggregate density function f, find a subset of nodes $S \subseteq V$, such that $f(S, \mathcal{G})$ is maximized.

By considering the four choices for the aggregate density function f, we have four variants of the BFF problem. Specifically, f_{mm}, f_{ma}, f_{am} and f_{aa} give rise to problems BFF-MM, BFF-MA, BFF-AM, and BFF-AA respectively.

3.2 BFF algorithms

We now introduce a generic algorithm for the BFF problem. The algorithm (shown in Algorithm 1) is a "greedy-like" algorithm inspired by a popular algorithm for the densest subgraph problem on a static graph (Asahiro et al. (2000); Charikar (2000)). We use $\mathcal{G}[S] = \{G_1[S], \ldots, G_\tau[S]\}$ to denote the sequence of the induced subgraphs of the set of nodes S. The algorithm starts with a set of nodes S_0 consisting of all nodes V, and then it performs $n-1$ steps, where at each step i it produces a set S_i by removing one of the nodes in the set S_{i-1}. It then returns the set S_i with the maximum aggregate density $f(S_i, \mathcal{G})$.

Algorithm 1 The FindBFF algorithm.

Input: Graph history $\mathcal{G} = \{G_1, \ldots, G_\tau\}$; aggregate density function f
Output: A subset of nodes S

1: $S_0 = V$
2: **for** $i = 1, \ldots, n-1$ **do**
3: $\quad v_i = \arg\min_{v \in S_{i-1}} score(v, \mathcal{G}[S_{i-1}])$
4: $\quad S_i = S_{i-1} \setminus \{v_i\}$
5: **return** $\arg\max_{i=0 \ldots n-1} f(S_i, \mathcal{G})$

The FindBFF algorithm forms the basis for the algorithms we propose for the four variants of the BFF problem. Interestingly, by defining appropriate scoring functions, $score(v, \mathcal{G}[S])$, (used in line 3 to select which node to remove), we can get effective algorithms for each of the variants.

3.2.1 Solving BFF-MM

For the BFF-MM problem, we define the score for a node v in S, $score_m$, as the minimum degree of v in the sequence $\mathcal{G}[S]$. That is,

$$score_m(v, \mathcal{G}[S]) = \min_{G_t \in \mathcal{G}} degree(v, G_t[S]).$$

Algorithm 2 The score$_m$ algorithm.

Input: Graph history $\mathcal{G} = \{G_1, \ldots, G_\tau\}$
Output: Node with the minimum score$_m$

```
1: L_t[d] ← list of nodes with degree d in G_t
2:
3: procedure ScoreAndUpdate()
4:     for t = 1, ..., τ do
5:         dmin_t ← smallest d s.t. L_t[d] ≠ ∅
6:     score_m = min_{t=1,...,τ} dmin_t
7:     t' = arg min_{t=1,...,τ} dmin_t
8:     u = L_{t'}[score_m].get()
9:     for each G_t ∈ G do
10:        L_t[degree(u, G_t)].remove(u)
11:        for each (u, v) ∈ E_t do
12:            L_t[degree(v, G_t)].remove(v)
13:            E_t = E_t − (u, v)   // update degree_{v∈V}(v, G_t)
14:            L_t[degree(v, G_t)].add(v)
15:    V = V \ {u}
16:    return u
```

Therefore, at the i-th iteration the FindBFF algorithm selects the node v_i with the minimum $score_m$ value. We call this instantiation of the FindBFF algorithm FindBFF$_{\mathrm{M}}$. Below we prove that FindBFF$_{\mathrm{M}}$ provides the optimal solution to the BFF-MM problem.

Proposition 1 *The* BFF-MM *problem can be solved optimally in polynomial time using the* FindBFF$_{\mathrm{M}}$ *algorithm.*

Proof Let i be the iteration of the FindBFF$_{\mathrm{M}}$ algorithm, where for the first time, a node that belongs to an optimal solution S^* is selected to be removed. Let v_i be this node. Clearly, $S^* \subseteq S_{i-1}$, and therefore $score_m(v_i, \mathcal{G}[S_{i-1}]) \geq score_m(v_i, \mathcal{G}[S^*])$. Since v_i is the node we pick at iteration i, every node $u \in S_{i-1}$ satisfies: $\min_{G_t \in \mathcal{G}} degree(u, G_t[S_{i-1}]) = score_m(u, \mathcal{G}[S_{i-1}]) \geq score_m(v_i, \mathcal{G}[S_{i-1}]) \geq score_m(v_i, \mathcal{G}[S^*])$. Since this is true for every node u, this means that S_{i-1} is indeed optimal and that our algorithm will find it.

The running time of FindBFF$_{\mathrm{M}}$ is $O(n\tau + M)$, where $n = |V|$, τ the number of snapshots in the history graph and $M = m_1 + m_2 + \ldots + m_\tau$ the total number of edges that appear in all snapshots. The node with the minimum $score_m$ value is computed by the procedure ScoreAndUpdate shown in Algorithm 2, which also removes the node and its edges from all snapshots. For each snapshot G_t, we keep the list of nodes $\mathcal{L}_t[d]$ with degree d (line 1 in Algorithm 2); these lists can be

constructed in time $O(n\tau + M)$. Furthermore, each position of the list $\mathcal{L}_t[d]$ points to a hash-based data structure which stores nodes with degree d in order to handle additions and deletions in constant time. Given these lists, the time required to find the node with the minimum $score_m$ is $O(\tau)$ (lines 4–8). Finding the minimum degree $dmin_t$ for each snapshot G_t at each step of the algorithm takes constant time, as the minimum degree can only decrease by at most one in each G_t. Now in all snapshots, the neighbors of the removed node need to be moved from their position in the τ lists (lines 9–14); the degree of every neighbor of the removed node is decreased by one. Throughout the execution of the algorithm at most $O(M)$ such moves can happen. Therefore, the total running time of FindBFF$_{\mathrm{M}}$ is $O(n\tau + M)$. Note that an algorithm that iteratively removes from a graph G the node with the minimum degree was first studied by Asahiro et al. (2000) and shown to compute a 2-approximation of the densest subgraph problem for the $d_a(G)$ density by Charikar (2000) and the optimal for the $d_m(G)$ density by Sozio and Gionis (2010).

3.2.2 Solving BFF-AA

To solve the BFF-AA problem, we shall use the average graph $\widehat{H}_{\mathcal{G}}$ of $\mathcal{G}$. Lemma 1 shows that $f_{aa}(S, \mathcal{G}) = d_a\left(\widehat{H}_{\mathcal{G}}[S]\right)$. Thus, based on the results of Charikar (2000) and Goldberg (1984), we conclude that:

Proposition 2 *The* BFF-AA *problem can be solved optimally in polynomial time.*

Although there exists a polynomial-time optimal algorithms for BFF-AA, the computational complexity of these algorithms (e.g., $O(|V||\widehat{E}|^2)$ for the case of the max-flow algorithm of Goldberg (1984)), makes them hard to use for large-scale real graphs. Therefore, instead of these algorithm we use the FindBFF algorithm, where we define the score of a node v in S, $score_a$, to be equal to its average degree of v in graph history $\mathcal{G}[S]$. That is,

$$score_a\left(v, \mathcal{G}[S]\right) = \frac{1}{|\mathcal{G}|} \sum_{G_t \in \mathcal{G}} degree\left(v, G_t[S]\right).$$

At the i-th iteration, we select the node v_i with the *minimum average degree in* $\mathcal{G}[S]$. We will refer to this instantiation of the FindBFF, as FindBFF$_{\mathrm{A}}$. Using Lemma 1 and the results of Charikar (2000) we have the following:

Proposition 3 FindBFF$_{\mathrm{A}}$ *is a* $\frac{1}{2}$*-approximation algorithm for the* BFF-AA *problem.*

Proof It is easy to see that FindBFF$_{\mathrm{A}}$ removes the node with the minimum density in $\widehat{H}_{\mathcal{G}}[S]$. Charikar (2000) has shown that an algorithm that iteratively removes from a graph the node with minimum density provides a $\frac{1}{2}$-approximation for finding the subset of nodes that maximizes the average density on a single (weighted) graph snapshot. Given the equivalence we established in Lemma 1, FindBFF$_{\mathrm{A}}$ is also a $\frac{1}{2}$-approximation algorithm for BFF-AA.

Using list of nodes with average degree d similarly to Algorithm 2 but on the average graph, we can efficiently find the minimum $score_a$ value and achieve an $O(n\tau + M)$ total running time for FindBFF$_{\mathrm{A}}$.

3.2.3 Solving BFF-MA *and* BFF-AM

We prove the following theorem of the complexity of the BFF-AM problem.

Theorem 1 *The* BFF-AM *problem is NP-hard.*

Proof The reduction is from the k-Clique problem, which, given a graph G, asks if the graph contains a clique of size at least k. The decision version of BFF-AM, given a graph history $\mathcal{G} = \{G_1, \ldots, G_\tau\}$, asks if there exists a subset of nodes S $f_{am}(S, \mathcal{G}) \geq \theta$ for some value θ.

Given a graph $G = (V, E)$ with $|V| = n$ nodes that is input to the k-Clique problem, we construct a graph history $\mathcal{G}$ with $\tau = n$ snapshots. All snapshots are defined over the vertex set V. There is a snapshot G_i for each node $i \in V$, consisting of a star-graph with node i as the center, and edges to all the neighbors of i in G. We will prove that there exists a clique of size at least k in graph G if and only if there exists a set of nodes S with $f_{am}(S, \mathcal{G}) \geq k/n$. The forward direction is easy; if there exists a subset of nodes S in G, with $|S| \geq k$, that form a clique, then for this set of nodes S, $d_m(S, G_i) = 1$ for all $i \in S$; therefore, $f_{am}(S, \mathcal{G}) \geq k/n$. To prove the other direction, we observe that all our snapshots consist of star graphs, and a collection of disconnected nodes. Given a set S, $d_m(S, G_i) = 1$, if $i \in S$ and all nodes in S are connected to the center node i, and zero otherwise. Therefore, if $f_{am}(S, \mathcal{C}_k) \geq k/n$, then this implies that $d_m(S, G_i) = 1$ for k snapshots $G_i \in \mathcal{G}$, which means that the k centers of the star graphs in these snapshots belong to S and they are connected to all nodes in S. Therefore, all nodes in S are connected to the k star centers, and hence the k star centers for a clique of size k in the graph G.

The complexity of BFF-MA is an open problem. Jethava and Beerenwinkel (2015) conjecture that it is NP-hard, yet they do not provide a proof.

We consider the application of FindBFF$_{\text{M}}$ and FindBFF$_{\text{A}}$ algorithms for the two problems. In the following propositions, we prove that the two algorithms cannot guarantee a good approximation ratio for all inputs for any of the two problems. Recall that all our problems are maximization problems, and, therefore, the lower the approximation ratio, the worse the performance of the algorithm. We construct instances of the problems for which the algorithms achieve approximation ratio that can be arbitrarily small as a function of the input size. Due to space limitations, we only give a sketch of the proofs; the complete proofs can be found in the full version of this paper (Semertzidis et al. (2016)).

Proposition 4 *The approximation ratio of algorithm* FindBFF$_{\text{M}}$ *is at most* $O\left(\frac{1}{n}\right)$*, for the* BFF-AM *problem, and at most* $O\left(\frac{1}{\sqrt{n}}\right)$*, for the* BFF-MA *problem, where* n *is the number of nodes.*

Proof (Sketch) The intuition behind the proof is to construct an input where there is a dense set of nodes A in the graph history $\mathcal{G} = \{G_1, ..., G_\tau\}$ that maximizes the density, but the nodes of the set have low degree in a single snapshot. FindBFF$_{\text{M}}$ will remove the nodes from this set, and thus never return it as a candidate solution.

For the BFF-AM problem, we construct a counter-example graph sequence as follows. The first $\tau - 1$ snapshots consist of a set A with $n - 1$ nodes that form

a full clique, plus an additional node v that is connected with a single node u in A. The last snapshot consists of just the edge (v, u). FINDBFF$_{\mathrm{M}}$ iteratively removes nodes from A, until left with edge (v, u) and thus always yields candidate solutions S_i that include v and have $f_{am}(S_i) = 1$, while the optimal solution is A with $f_{am}(A) = O(n)$.

For the BFF-MA problem, we construct a counter-example graph sequence with m snapshots. Each snapshot consists of two sets of nodes A and B of size m and m^2 respectively. The nodes in B form a cycle. The nodes in a A form a clique with all nodes except one node, different in each snapshot. The FINDBFF$_{\mathrm{M}}$ iteratively removes the nodes in A, and thus always yields candidate solutions S_i with $f_{ma}(S_i) = \Theta(1)$, while the optimal solution is A with $f_{ma}(A) = \Theta(m) = \Theta(\sqrt{n})$.

Proposition 5 *The approximation ratio of algorithm* FINDBFF$_{\mathrm{A}}$ *is at most* $O\left(\frac{1}{n}\right)$ *for the* BFF-AM *problem, and at most* $O\left(\frac{1}{\sqrt{n}}\right)$*, for the* BFF-MA *problem, where* n *is the number of nodes.*

Proof (Sketch) The counter-example for the BFF-AM problem is complex and hard to explain in brief. The intuition is that we can construct an input where there are two nodes that have very low degree in (different) half of the snapshots, yet high average degree overall. As a result they are never removed by the FINDBFF$_{\mathrm{A}}$ algorithm, resulting in constant f_{am} value, while removing them gives $f_{am}(S) = \Theta(n)$.

The counter-example for the BFF-MA problem is a graph sequence of m graph snapshots. Each snapshot consist of two set of nodes A and B of size m and m^2 respectively. The nodes in A form a clique in all snapshots, while the nodes in B form a clique in all but one snapshot, in which, they are disconnected. FINDBFF$_{\mathrm{A}}$ iteratively removes the nodes in A, thus yielding candidate solutions S_i with $f_{ma}(S_i) = \Theta(1)$, while the optimal solution is A with $f_{ma}(A) = \Theta(m) = \Theta(\sqrt{n})$.

Given that FINDBFF$_{\mathrm{A}}$ and FINDBFF$_{\mathrm{M}}$ have no theoretical guarantees, we also investigate a *greedy* approach, which selects which node to remove based on the objective function of the problem at hand. This greedy approach is again an instance of the iterative algorithm shown in Algorithm 1. More specifically, for a target function f (either f_{am} or f_{ma}), given a set S_{i-1}, we define the score $score_g(v, \mathcal{G}[S_i])$ of node $v \in S_i$ as follows:

$$score_g(v, \mathcal{G}[S_{i-1}]) = f\left(S_{i-1}, \mathcal{G}\right) - f\left(S_{i-1} \setminus \{v\}, \mathcal{G}\right).$$

At iteration i, the algorithm selects the node v_i that causes the smallest decrease, or the largest increase in the target function f. We refer to this algorithm as FINDBFF$_{\mathrm{G}}$. FINDBFF$_{\mathrm{G}}$ complexity is $O(n^2\tau + nM)$ since it requires to check all nodes when choosing which node to remove at each step.

4 The O^{2}BFF problem

In this section, we relax the requirement that density is computed over all snapshots of the graph history. Instead, we ask for a set of k snapshots and a set of nodes such that the aggregate density over these snapshots is maximized. We call

this problem *On-Off* BFF (O^2BFF) problem. We formally define O^2BFF, we show that it is NP-hard, and develop two general types of algorithms for efficiently solving it in practice.

4.1 Problem definition

In the O^2BFF problem, we seek to find a collection $\mathcal{C}_k$ of k graph snapshots, and a set of nodes $S \subseteq V$, such that the subgraphs induced by S in $\mathcal{C}_k$ have high aggregate density. Formally, the O^2BFF problem is defined as follows:

Problem 2 (The On-Off BFF (O^2BFF) Problem) Given a graph history $\mathcal{G} = \{G_1, G_2, \ldots, G_\tau\}$, an aggregate density function f, and an integer k, find a subset of nodes $S \subseteq V$, and a subset $\mathcal{C}_k$ of $\mathcal{G}$ of size k, such that $f(S, \mathcal{C}_k)$ is maximized.

As with the BFF problem, depending on the choice of the aggregate density function f, we have four variants of the O^2BFF problem, namely O^2BFF-MM, O^2BFF-MA, O^2BFF-AM and O^2BFF-AA.

Note that the subcollection of graphs $\mathcal{C}_k \subset \mathcal{G}$ does not need to consist of contiguous graph snapshots. If this were the case, then the problem could be solved easily by considering all possible contiguous subsets of $[1, \tau]$ and outputting the one with the highest density. However, all four variants of the O^2BFF become NP-hard if we drop the constraint for consecutive graph snapshots.

Theorem 2 *The O^2BFF problem is NP-hard for any definition of the aggregate density function f.*

Proof For all aggregate density functions, the reduction is from the k-Clique problem, which, given a graph G, asks if the graph contains a clique of size at least k. The decision version of O^2BFF, given a graph history $\mathcal{G} = \{G_1, \ldots, G_\tau\}$, asks if there exists a subset of nodes S and a subset $\mathcal{C}_k$ of k snapshots, such that $f(S, \mathcal{C}_k) \geq \theta$ for some value θ.

The reduction differs depending on the definition of f. In the case of f_{mm} and f_{am}, the construction and proof is the same as that of Theorem 1. Given a graph $G = (V, E)$ with $|V| = n$ nodes that is input to the k-Clique problem, we construct a graph history $\mathcal{G}$ with $\tau = n$ snapshots, where snapshot G_i is a star-graph with node i as the center, and edges to all the neighbors of i in G.

We will prove that there exists a clique of size at least k in graph G if and only if there exists a set of nodes S and a subset $\mathcal{C}_k \subseteq \mathcal{G}$ of k snapshots, with $f(S, \mathcal{C}_k) \geq 1$. The forward direction is easy; if there exists a subset of nodes S in G, with $|S| \geq k$, that form a clique, then selecting this set of nodes S, and a subset $\mathcal{C}_k$ of k snapshots that correspond to nodes in S will wield $f_{mm}(S, \mathcal{C}_k) = f_{am}(S, \mathcal{C}_k) = 1$. This follows from the fact that every snapshot is a complete star where $d_m(S, G_i) = 1$ for all $G_i \in \mathcal{C}_k$. To prove the other direction, we observe that all our snapshots consist of a star graph, and a collection of disconnected nodes. Given a set S, $d_m(S, G_i) = 1$, if $i \in S$ and all nodes in S are connected to the center node i, and zero otherwise. Therefore, if $f_{mm}(S, \mathcal{C}_k) = 1$ or $f_{am}(S, \mathcal{C}_k) = 1$, then this implies that $d_m(S, G_i) = 1$ for all $G_i \in \mathcal{C}_k$, which means that the k centers of the graph snapshots in $\mathcal{C}_k$ are connected to all nodes in S, and hence to each other. Therefore, they form a clique of size k in the graph G.

Algorithm 3 The Iterative (ITR) FindO^{2}BFF algorithm.

Input: Graph history $\mathcal{G} = \{G_1, \dots G_\tau\}$; an aggregate-density function f; integer k
Output: A subset of nodes S and a subset of snapshots $\mathcal{C}_k \subseteq \mathcal{G}$.

converged = False
$(\mathcal{C}_k^0, S^0) =$ Initialize $(\mathcal{G}, f)$
$ds^0 = f(S^0, \mathcal{C}_k^0)$
while not converged **do**
  $\mathcal{C}_k =$ BestSnapshots(S^0, f)
  $S =$ FindBFF$(\mathcal{C}_k, f)$
  $ds = f(S, \mathcal{C}_k)$
  if $ds \leq ds^0$ **then**
    $(S, \mathcal{C}_k) = (S^0, \mathcal{C}_k^0)$
    Converged = True
  else
    $(ds^0, S^0, \mathcal{C}_k^0) = (ds, S, \mathcal{C}_k)$
return $S, \mathcal{C}_k$

In the case of f_{aa} and f_{ma} the construction proceeds as follows: given the graph $G = (V, E)$, with $|E| = m$ edges, we construct a graph history $\mathcal{G} = \{G_1, \dots, G_\tau\}$ with $\tau = m$ snapshots. All snapshots are defined over the vertex set V. There is a snapshot G_e for each edge $e \in E$, consisting of the single edge e.

We will prove that there exists a clique of size at least k in graph G if and only if there exists a set of nodes S and a subset $\mathcal{C}_K \subseteq \mathcal{G}$ of $K = k(k-1)/2$ snapshots, with $f(S, \mathcal{C}_K) \geq 1/k$. The forward direction is easy. If there exists a subset of nodes S in G, with $|S| = k$, that form a clique, then selecting this set of nodes S, and the $\binom{k}{2}$ snapshots $\mathcal{C}_K$ in $\mathcal{G}$ that correspond to the edges between the nodes in S will yield $f_{aa}(S, \mathcal{C}_K) = f_{ma}(S, \mathcal{C}_K) = 1/k$.

To prove the other direction, assume that there is no clique of size greater or equal to k in G. Let $\mathcal{C}_K$ be any subset of $K = k(k-1)/2$ snapshots, and let S be the union of the endpoints of the edges in $\mathcal{C}_K$. Since S cannot be a clique, it follows that $|S| = \ell > k$. Therefore, $f_{aa}(S, \mathcal{C}_K) = f_{ma}(S, \mathcal{C}_K) = 1/\ell < 1/k$.

4.2 O^{2}BFF algorithms

We consider two general types of algorithms: iterative and incremental ones. The *iterative algorithms* start with an initial solution of the problem and improve it, whereas the *incremental algorithms* build the solution incrementally, adding one snapshot at a time. Next, we describe these two types of algorithms in detail. Note that in each of the algorithms, depending on which of the O^{2}BFF-MM, O^{2}BFF-MA, O^{2}BFF-AM or O^{2}BFF-AA problems we are solving, we use the appropriate version of the FindBFF algorithm.

Iterative algorithm. The iterative (ITR) algorithm (shown in Algorithm 3) starts with an initial collection of k snapshots $\mathcal{C}_k^0$ and set of nodes S^0 (routine Initialize). At each iteration, given a set S, ITR finds the best collection of k graph snapshots for S; this is done by BestSnapshots. BestSnapshots computes the density $d(S, G_i)$ of S in each snapshot $G_i \in \mathcal{G}$ and outputs the k snapshots $\mathcal{C}_k$ with the largest density. Then, given the collection $\mathcal{C}_k$, the algorithm finds the best set S

Algorithm 4 The Incremental Density (INC_D) FindO^{2}BFF algorithm.

Input: Graph history $\mathcal{G} = \{G_1, \dots G_\tau\}$; aggregate-density function f; integer k
Output: A subset of nodes S and a subset of snapshots $\mathcal{C}_k \subseteq \mathcal{G}$.

$S_{ij} = \text{FindBFF}(\{G_i, G_j\}, f)$, $\forall G_i, G_j \in \mathcal{G}$
$\mathcal{C}_2 = \arg\max_{G_i, G_j \in \mathcal{G}} f(S_{ij}, \{G_i, G_j\})$
for $i = 3$; $i \leq k$ **do**
  for each $G_t \in \mathcal{G} \setminus \mathcal{C}_{i-1}$ **do**
    $S_t = \text{FindBFF}(\mathcal{C}_{i-1} \cup \{G_t\}, f)$
  $G_m = \arg\max_{G_t} f(S_t, \mathcal{C}_{i-1} \cup \{G_t\})$
  $\mathcal{C}_i = \mathcal{C}_{i-1} \cup \{G_m\}$
return $S, \mathcal{C}_k$

for $\mathcal{C}_k$, that is, the set $S \subseteq V$ such that $f(S, \mathcal{C}_k)$ is maximized. This step essentially solves Problem 1 on input $\mathcal{C}_k$ for aggregate density function f using the FindBFF algorithm. ITR keeps iterating between collections $\mathcal{C}_k$ and dense sets of nodes S until no further iterations can improve the score $f(S, \mathcal{C}_k)$.

An important step of the iterative FindO^{2}BFF algorithm is the initialization of $\mathcal{C}_k^0$ and S^0. We consider three alternative initializations.

Random initialization (ITR_R): In this initialization, we randomly pick k snapshots $\mathcal{C}_k^0$ from $\mathcal{G}$ and use them to produce $S^0 = \text{FindBFF}(\mathcal{C}_k^0, f)$.

Contiguous initialization (ITR_C): The motivation behind contiguous initialization is that in many real world datasets, such as in those modeling collaboration networks that evolve with time, there is temporal locality. Thus, we expect that the dense subgraphs will appear in nearby snapshots. Consequently, given $\mathcal{G} = \{G_1, \dots, G_\tau\}$, we go over all the $O(\tau)$ contiguous sets of k snapshots from $\mathcal{G}$, and find the set of k snapshots $\mathcal{C}_k^0$ and corresponding set of nodes S^0 that maximize $f(S^0, \mathcal{C}_k^0)$.

At least-k initialization (ITR_K): With this initialization, our aim is to include in the initial set S^0 the nodes that appear to be densely connected in many snapshots. Thus, we solve the BFF problem independently in each snapshot $G_i \in \mathcal{G}$. This results in τ sets $S_i \subseteq V$, one for each G_i. S^0 includes the nodes that appear in at least k of the τ sets S_i. We also experimented with other natural alternatives, such as the union: $S^0 = \cup_{i=1\dots\tau} S_i$ and the intersection: $S^0 = \cap_{i=1\dots\tau} S_i$; the at least-$k$ approach seems to strike a balance between the two.

The running time of the iterative FindO^{2}BFF algorithm is $O(I(n\tau + M))$, where I is the number of iterations required until convergence, and $(n\tau + M)$ comes from the running time of FindBFF, assuming that we use FindBFF_M or FindBFF_A (which can be accordingly modified for FindBFF_G). In practice, we observed that the algorithm converges in at most 6 iterations.

Incremental algorithm. The incremental algorithm starts with a collection $\mathcal{C}_2$ with two snapshots and incrementally adds snapshots to it until a collection $\mathcal{C}_k$ with k snapshots is formed. Then, the appropriate FindBFF algorithm is used to compute the most dense subset of nodes S in $\mathcal{C}_k$. We use two different policies for selecting snapshots. The first one, termed *incremental density* (INC_D) (shown in Algorithm 4), adds snapshots so as to maximize density, whereas the second one,

Algorithm 5 The Incremental Overlap (INC_O) FindO^{2}BFF algorithm.

Input: Graph history $\mathcal{G} = \{G_1, \dots G_\tau\}$; aggregate-density function f; integer k
Output: A subset of nodes S and a subset of snapshots $\mathcal{C}_k \subseteq \mathcal{G}$.

1: $S_i = \text{FindBFF}(G_i, f)$, $\forall G_i \in \mathcal{G}$
2: $\mathcal{C}_2 = \arg \max_{G_i, G_j \in \mathcal{G}} \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$
3: **for** $i = 3$; $i \leq k$ **do**
4: $\quad S_C = \text{FindBFF}(\mathcal{C}_{i-1}, f)$
5: $\quad G_m = \arg\max_{G_t} \frac{|S_t \cap S_C|}{|S_t \cup S_C|}$
6: $\quad \mathcal{C}_i = \mathcal{C}_{i-1} \cup \{G_m\}$
7: $S = \text{FindBFF}(\mathcal{C}_k, f)$
8: **return** $S, \mathcal{C}_k$

termed *incremental overlap* (INC_O) (shown in Algorithm 5), adds snapshots so as to maximize the similarity of their dense subgraphs.

Incremental density (INC_D): To select the pair of snapshots to form the initial collection $\mathcal{C}_2$, we solve the BFF problem independently for each pair of snapshots $G_i, G_j \in \mathcal{G}$. This gives us $\binom{\tau}{2}$ dense sets S_{ij} as solutions. We select the pair of snapshots whose dense subgraph S_{ij} has the largest density (lines 1–2). INC_D then builds the solution incrementally in $k-2$ iterations by adding at each iteration the snapshot whose addition maximizes density. Specifically, in the i-th iteration, we construct $\mathcal{C}_i$ by adding to $\mathcal{C}_{i-1}$ the graph snapshot $G_m = \arg\max_{G_t} f(S_t, \mathcal{C}_{i-1} \cup \{G_t\})$, over all G_t in $\mathcal{G} \setminus \mathcal{C}_{i-1}$. The running time is $O\left(\tau^2(n+M) + k\tau\left(n\tau + M\right)\right)$, where the first term is due to the initialization step (again assuming that we use FindBFF_M or FindBFF_A).

Incremental overlap (INC_O): Our goal is to find snapshots whose dense subgraphs have many nodes in common. To form the initial collection $\mathcal{C}_2$, we solve the BFF problem independently in each snapshot $G_i \in \mathcal{G}$. This gives us τ different sets $S_i \subseteq V$, where S_i is the most dense subgraph in G_i. The algorithm selects from these τ sets the two most similar ones, S_i and S_j, using Jaccard similarity, and initializes $\mathcal{C}_2$ with the corresponding snapshots G_i and G_j (lines 1–2). To form $\mathcal{C}_i$ from $\mathcal{C}_{i-1}$, the algorithm first solves the BFF problem in $\mathcal{C}_{i-1}$. Let S_C be the solution. Then, it selects from the remaining snapshots and adds to $\mathcal{C}_{i-1}$ the snapshot G_m whose dense set S_t is the most similar with S_C (lines 3–6). The running time is $O(k\left(n\tau + M\right))$ (again assuming that we use FindBFF_M or FindBFF_A).

Note that the incremental algorithm can be easily modified so as, instead of the number k of snapshots being an input to the algorithm, an appropriate value of k is determined in the course of the algorithm. For example, snapshots could be added to the solution until density drops below a given threshold value.

5 Experimental evaluation

The goal of our experimental evaluation is threefold. First, we want to evaluate the performance of our algorithms for the BFF and the O^{2}BFF problems in terms of the quality of the solutions and running time. Second, we want to compare the different variants of the aggregate density functions. Third, we want to show the

usefulness of the problem, by presenting results of BFF's and O^2BFF's in two real datasets, namely research collaborators in *DBLP* and hashtags in Twitter.

Table 1: Real dataset characteristics

Dataset	# Nodes	# Edges (aver. per snapshot)	# Snapshots
$DBLP_{10}$	2,625	1,143	10
$Oregon_1$	11,492	22,569	9
$Oregon_2$	11,806	31,559	9
Caida	31,379	45,833	122
Twitter	849	100	15
AS	7,716	7,783	733

Datasets and setting. To evaluate our approach, we use both real and synthetic datasets. We use six real graph histories where the graphs correspond to collaboration, computer, and concept networks (summarized in Table 1). The $DBLP_{10}$[1] dataset contains yearly snapshots of the co-authorship graph in the 2006-2015 interval for 11 top database and data mining conferences. There is an edge between two authors in a graph snapshot, if they co-authored a paper in the corresponding year and more than two papers in total. The $Oregon_1$[2] dataset consists of nine graph snapshots of autonomous systems (AS) peering information inferred from Oregon route-views (one snapshot per week), while the $Oregon_2$[3] dataset includes in addition to route-views looking glass data and routing registry, all combined. The *Caida*[4] dataset contains 122 AS graphs, derived from a set of route views BGP-table instances. The *Twitter* dataset (Tsantarliotis and Pitoura (2015)) contains 15 daily snapshots from October 27, 2013 to November 10, 2013, where the nodes are hashtags and there is an edge between two nodes if the corresponding hashtags co-appear in a tweet. The *AS*[5] dataset consists of 733 daily snapshots representing a communication network of who-talks-to-whom from the BGP (Border Gateway Protocol) logs.

We also use synthetic datasets. In particular, we create graph snapshots using the forest fire model (Leskovec et al. (2007)), a well-known model for creating evolving networks, using the default forward and backward burning probabilities of 0.35. Then, we plant dense subgraphs in these snapshots, by randomly selecting a set $X \subset V$ of the nodes and creating additional edges between them, different at each snapshot. In all experiments, we create 100 such graph histories and report average values.

We ran our experiments on a system with a quad-core Intel Core i7-3820 3.6 GHz processor, with 64 GB memory. We only used one core in all experiments.

Both the code and the datasets used in our experiments are publicly available[6].

[1] http://dblp.uni-trier.de/
[2] https://snap.stanford.edu/data/oregon1.html
[3] https://snap.stanford.edu/data/oregon2.html
[4] http://www.caida.org/data/as-relationships/
[5] https://snap.stanford.edu/data/as.html
[6] https://github.com/ksemer/BestFriendsForever-BFF-

5.1 BFF evaluation

In terms of algorithms, for the BFF-MM and BFF-AA problems, FindBFF$_{\mathrm{M}}$ and FindBFF$_{\mathrm{A}}$ provide provably good solutions respectively (as shown in Section 3.2), thus we only consider these algorithms for these problems. For the BFF-MA and BFF-AM problems, we use all three algorithms, i.e., FindBFF$_{\mathrm{M}}$, FindBFF$_{\mathrm{A}}$, and FindBFF$_{\mathrm{G}}$. In addition, for the BFF-MA problem, we use the *DCS* algorithm proposed by Jethava and Beerenwinkel (2015) for a problem similar to BFF-MA. The *DCS* algorithm is also an iterative algorithm that removes nodes, one at a time. At each step, *DCS* finds the subgraphs with the largest average density for each of the snapshots. Then, it identifies the subgraph with the smallest average density among them and removes the node that has the smallest degree in this subgraph.

Quality of the solution and comparison of the density function definitions: We start with an evaluation of the accuracy of our algorithms along with a comparison of the different aggregate densities. Since we do not have any ground truth information for the real data, we use first the synthetic datasets.

Synthetic datasets. We create 10 graph snapshots with $4,000$ nodes each using the forest fire model (Leskovec et al. (2007)). Then, in each one of the 10 snapshots we plant a dense random subgraph A with 100 nodes by inserting extra (different at each snapshot) edges with probability p_A. We consider subgraph A as our ground truth. We vary the edge probabilities from $p_A = 0.1$ to $p_A = 0.9$. In Fig. 2(a), we report the F measure for the four aggregate density definitions, when trying to recover A. Recall that F takes values in $[0, 1]$ and the larger the value the better the recall and precision of the solution with respect to the ground truth (in this case A). BFF-MM is the most sensitive measure, since it reports A as the densest subgraph even for the smallest edge probability. BFF-MA and BFF-AM achieve a perfect F value, for an edge probability larger than $p_A = 0.1$ and BFF-AA for an edge probability at least $p_A = 0.3$. For smaller values, these three density definitions locate supersets of A, due to averaging. All variations of the FindBFF algorithms produce the same results.

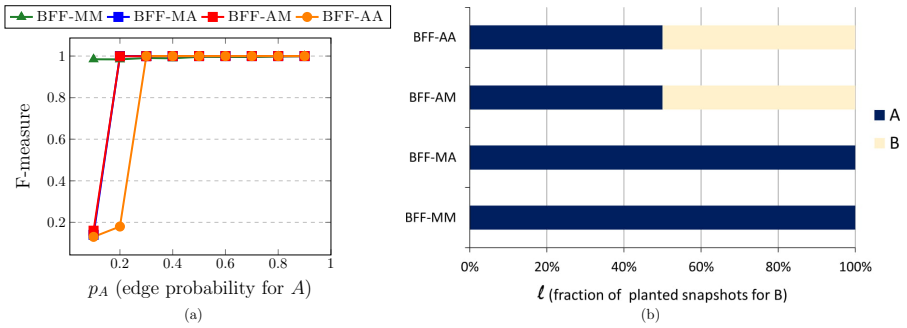


Fig. 2: Accuracy and density definitions: (a) F-measure for planted graph A, (b) reported dense subgraph ($p_A = 0.5$, $p_B = 0.9$).

We now study how the various density definitions behave when there is a second dense subgraph. In this case, we plant a subgraph A with edge probability $p_A = 0.5$ in all snapshots and a second dense subgraph B with the same number of nodes as A and edge probability $p_B = 0.9$ in a percentage ℓ of the snapshots, for different values of ℓ. Fig. 2(b) depicts which of two graphs, graph A (shown in blue), or graph B (shown in yellow), is output by the FindBFF algorithms for the different density definitions. BFF-MM and BFF-MA report A as the densest subgraph, since these measures ask for high density at each and every snapshot. However, BFF-AM and BFF-AA report B, when the denser subgraph B appears in a sufficient number (more than half) of the snapshots. All density definitions and algorithms, recover the exact set A, or B, at each case.

Table 2: Results of the algorithms for the BFF-MM and BFF-AA problems on the real datasets.

Datasets	BFF-MM			
	FindBFF$_{\text{M}}$		*Random*	
	f_{mm}	size	$\overline{f}_{mm}$	SD
$DBLP_{10}$	1.0	11	0.01	0.09
$Oregon_1$	14.0	33	0.84	0.37
$Oregon_2$	23.0	75	0.02	0.14
Caida	8.0	17	0.1	0.30
Twitter	0.0	-	0.0	0.0
AS	4.0	15	0.0	0.0

Datasets	BFF-AA			
	FindBFF$_{\text{A}}$		*Random*	
	f_{aa}	size	$\overline{f}_{aa}$	SD
$DBLP_{10}$	2.75	8	0.92	0.27
$Oregon_1$	25.73	59	4.43	0.72
$Oregon_2$	47.89	147	7.59	1.06
Caida	33.21	96	5.33	0.36
Twitter	1.38	5	0.0	0.0
AS	16.38	38	2.01	0.49

Table 3: Results of the algorithms for the BFF-MA problem on the real datasets.

Datasets	BFF-MA									
	FindBFF$_{\text{M}}$		FindBFF$_{\text{A}}$		FindBFF$_{\text{G}}$		DCS		*Random*	
	f_{ma}	size	f_{ma}	size	f_{ma}	size	f_{ma}	size	$\overline{f}_{ma}$	SD
$DBLP_{10}$	1.33	3	1.75	8	1.7	61	1.29	14	0.12	0.15
$Oregon_1$	23.7	80	23.86	70	24.05	80	24.05	77	4.75	0.80
$Oregon_2$	44.33	140	45.24	131	45.95	132	44.91	116	6.71	1.24
Caida	13.76	33	12.76	29	15.43	6	15.05	57	0.60	0.53
Twitter	0.04	836	0.29	7	0.62	13	0.05	720	0.0	0.0
AS	8.53	19	6.67	18	9.0	20	8.75	16	0.19	0.11

Real datasets. We also run all algorithms using the real datasets and present the results in Table 2, 3, and 4. We report the density and the size of the solution. In addition, to evaluate the quality of the recovered dense subgraphs, we performed the following randomization test. For each of the real datasets, we create a random subgraph with the same number of nodes as the recovered subgraph, by initiating a BFS traversal from a randomly selected node. In Tables 2, 3, and 4, we also report the density of these subgraphs (average and standard deviation (SD) over 100 tests). For the BFF-MA and BFF-AM problems, we use the size of the solution that has the highest density.

Table 4: Results of the algorithms for the BFF-AM problem on the real datasets.

Datasets	BFF-AM							
	FindBFF$_{\mathrm{M}}$		FindBFF$_{\mathrm{A}}$		FindBFF$_{\mathrm{G}}$		*Random*	
	f_{am}	size	f_{am}	size	f_{am}	size	$\overline{f}_{am}$	SD
$DBLP_{10}$	1.0	11	1.7	4	1.0	4	0.23	0.29
$Oregon_1$	14.22	33	15.0	35	2.0	20	0.53	0.33
$Oregon_2$	24.44	63	23.22	44	3.22	461	0.0	0.0
Caida	12.72	20	18.11	36	3.43	311	0.0	0.0
Twitter	0.0	-	1.0	3	1.0	3	0.0	-
AS	7.44	12	9.05	14	3.14	14	0.0	0.0

Table 5: Execution time (sec) of the algorithms for the BFF-MM and BFF-MA problem on the real datasets.

Datasets	BFF-MM
	FindBFF$_{\mathrm{M}}$
$DBLP_{10}$	0.08
$Oregon_1$	0.27
$Oregon_2$	0.36
Caida	2.24
Twitter	0.37
AS	3.49

Datasets	BFF-MA			
	FindBFF$_{\mathrm{M}}$	FindBFF$_{\mathrm{A}}$	FindBFF$_{\mathrm{G}}$	DCS
$DBLP_{10}$	0.05	0.03	2.04	0.34
$Oregon_1$	0.24	0.21	48	0.83
$Oregon_2$	0.29	0.47	51.58	1.03
Caida	2.51	2.30	2,519	11.22
Twitter	0.57	0.24	2.81	0.47
AS	2.82	2.16	738	17.37

Table 6: Execution time (sec) of the algorithms for the BFF-AM and BFF-AA problems on the real datasets.

Datasets	BFF-AM		
	FindBFF$_{\mathrm{M}}$	FindBFF$_{\mathrm{A}}$	FindBFF$_{\mathrm{G}}$
$DBLP_{10}$	0.05	0.08	1.58
$Oregon_1$	0.48	0.57	131
$Oregon_2$	0.58	0.65	117.58
Caida	6.31	5.97	1,652
Twitter	0.85	0.28	2.65
AS	9.29	10.43	470

Datasets	BFF-AA
	FindBFF$_{\mathrm{A}}$
$DBLP_{10}$	0.04
$Oregon_1$	0.28
$Oregon_2$	0.48
Caida	2.14
Twitter	0.52
AS	2.64

As expected, the density of the random "BFS" graph is orders of magnitude smaller than the density of the graph recovered by our algorithms. Note also, that the value of the aggregate density (independently of the problem variant) is larger for the more dense datasets. For BFF-MM problem we observe that the solutions usually have small cardinality compared to the solutions for the other problems, since the f_{mm} objective is rather strict (the solution for *Twitter* was empty). The solutions for BFF-MM problem in the autonomous-system datasets appear to have higher f_{mm} scores. This may be due to the fact that there are larger groups of nodes with lasting connections in these datasets, e.g., nodes that communicate intensely between each other during the observation period.

Comparison of FindBFF alternatives for BFF-MA and BFF-AM: As shown in Table 3, for the BFF-MA problem, FindBFF$_{\mathrm{G}}$ and FindBFF$_{\mathrm{A}}$ perform overall the best in all datasets producing subgraphs with large f_{ma} values. FindBFF$_{\mathrm{A}}$

performs slightly worse than FindBFF$_{\mathrm{G}}$ only in the *Caida* dataset. In the *Caida* dataset, due probably to the large number of snapshots, FindBFF$_{\mathrm{A}}$ – which is based on the average degree – returns a set with the smallest density. FindBFF$_{\mathrm{M}}$ and *DCS* have comparable performance, since they both remove nodes with small degrees in individual snapshots. They are both outperformed by FindBFF$_{\mathrm{A}}$ and FindBFF$_{\mathrm{G}}$.

For the BFF-AM problem in Table 4, we observe that FindBFF$_{\mathrm{A}}$ outperforms both FindBFF$_{\mathrm{M}}$ and FindBFF$_{\mathrm{G}}$. Our deeper analysis of the inferior performance of FindBFF$_{\mathrm{G}}$ for this problem revealed that FindBFF$_{\mathrm{G}}$ often gets trapped in local maxima after removing just a few nodes of the graph and it cannot find good solutions.

In Table 5 and Table 6, we report execution times. As expected, the response time of FindBFF$_{\mathrm{G}}$ algorithm is the slowest in all datasets, due to its quadratic complexity. For the BFF-MA problem, FindBFF$_{\mathrm{A}}$ is in general faster than DCS. The difference in execution times of FindBFF$_{\mathrm{M}}$ algorithms for the various problems are due to differences in the computation of the density functions.

Scalability: We also test the scalability of the algorithms in terms of both the size of the graphs and the number of snapshots using the synthetic datasets. For testing scalability with size, we create 10 graph snapshots with N nodes (for N = 20,000 up to 100,000). Then, in each one of the 10 snapshots we plant a dense random subgraph A with 100 nodes by inserting extra (different at each snapshot) edges with probability $p_A = 0.5$. We consider subgraph A as our ground truth. In Fig. 3(a), we report the average execution time (and variance) of the different algorithms for the BFF-MA problem. The corresponding algorithms have similar performance for the other BFF problems as well. For testing scalability with the number of graph snapshots, we create T snapshots of a graph with 50,000 nodes as before, for $T = 10$ up to 50 snapshots. We report the average execution time (and variance) in Fig. 3(b). All algorithms, except FindBFF$_{\mathrm{G}}$, scale well with both the size of the graph and the number of snapshots. In terms of accuracy, all algorithms in both cases achieve a perfect F measure.

Summary: In conclusion, our algorithms successfully discovered the planted dense subgraphs even when their density is small, with BFF-MM being the most sensitive measure. Minimum aggregation over densities (i.e., BFF-MM, BFF-MA) requires a dense subgraph to be present at all snapshots, whereas average aggregation over densities (i.e., BFF-AM, BFF-AA) asks that the nodes are sufficiently connected with each other on average. For the BFF-MA and BFF-AM problems, FindBFF$_{\mathrm{A}}$ returns in general denser subgraphs than the alternatives (including DCS). Both FindBFF$_{\mathrm{A}}$ and FindBFF$_{\mathrm{M}}$ scale well. They perform similarly for the different density functions with the differences in running time attributed to the complexity of calculating the respective functions.

5.2 O^{2}BFF evaluation

In this set of experiments, we evaluate the performance of the iterative and incremental FindO^{2}BFF algorithms.

Comparison of the algorithms in terms of solution quality: We start with an evaluation of the quality of the solution produced by the proposed FindO^{2}BFF algorithms.

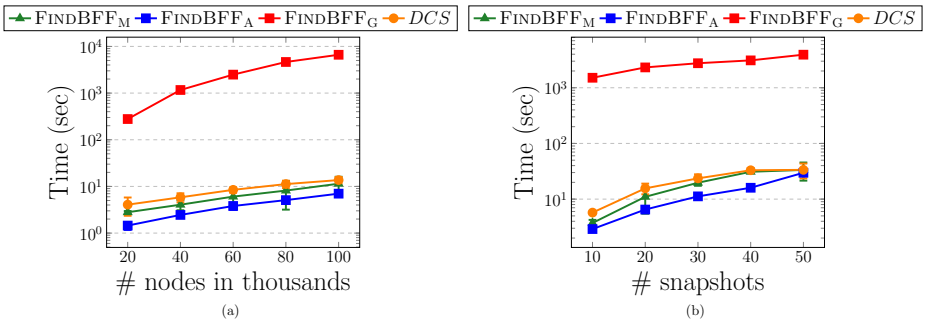


Fig. 3: Synthetic dataset ($p_A = 0.5$): execution time (log scale) of the different algorithms for the BFF-MA problem for varying number of (a) nodes, and (b) snapshots.

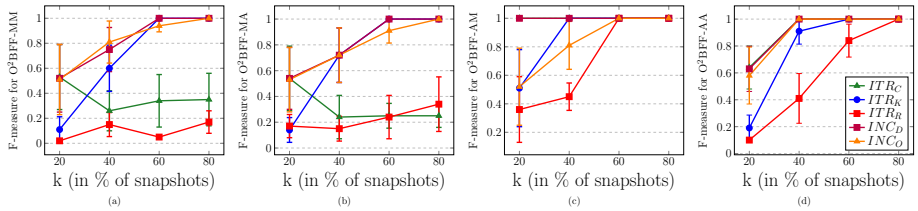


Fig. 4: Synthetic dataset ($p_A = 0.9$): F-measure for the: (a) O^{2}BFF-MM (b) O^{2}BFF-MA, (c) O^{2}BFF-AM, and (d) O^{2}BFF-AA problems.

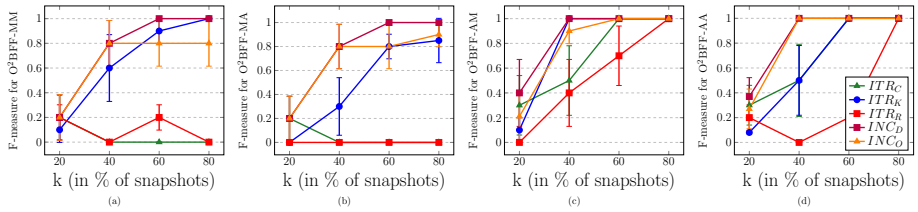


Fig. 5: Synthetic dataset ($p_A = 0.5$, $p_B = 0.9$): F-measure for the: (a) O^{2}BFF-MM (b) O^{2}BFF-MA, (c) O^{2}BFF-AM, and (d) O^{2}BFF-AA problems.

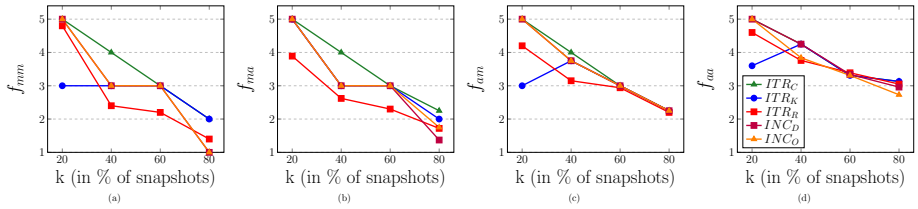


Fig. 6: $DBLP_{10}$ dataset: aggregate density functions f

Synthetic datasets. Similar to before, we plant a dense random graph A in k snapshots. We then run the FindO^{2}BFF algorithms with the same value of k. In Fig. 4,

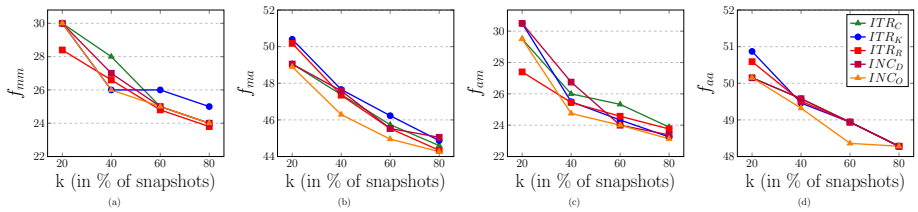


Fig. 7: *Oregon*$_2$ dataset: aggregate density functions f

we report the average F measure (and standard deviation) for the different values of k expressed as a percentage of the total number of snapshots. For the iterative FindO^{2}BFF algorithm, the *at-least-k* initialization (ITR$_K$) outperforms the other two, and it successfully locates A for all four density definitions, when A appears in a sufficient number of snapshots. Non-surprisingly, all initializations work equally well for average aggregation over time (i.e., O^{2}BFF-AM and O^{2}BFF-AA). For the incremental FindO^{2}BFF algorithm, *density* (INC$_D$) slightly outperforms *overlap* (INC$_O$). Overall, the incremental algorithms achieve highest F, when compared with the iterative ones.

We conduct a second experiment in which we plant a dense random graph A with edge probability $p_A = 0.5$ in all snapshots and a dense random graph B with the same number of nodes as A and edge probability $p_B = 0.9$ in k snapshots. In Fig. 5, we report the average F measure (and standard deviation) assuming that B is the correct output for the O^{2}BFF problem for different values of k expressed as a percentage of the total number of snapshots. Again, by comparing the different initializations for the iterative FindO^{2}BFF algorithm, we observe that among the iterative algorithms, ITR$_K$ successfully locates B for all four density definitions, when B appears in a sufficient number of snapshots. As in the previous experiment, all initializations work equally well for average aggregation over time. The incremental algorithms outperform the iterative ones with INC$_D$ being the champion, achieving high F values even when B appears in a few snapshots.

Real datasets. We also apply the FindO^{2}BFF algorithms on all real datasets for various values of k. In Figs. 6 and 7, we report the value of the aggregate density for *DBLP*$_{10}$ and *Oregon*$_2$ for different values of k, again expressed as a percentage of the total number of snapshots of the input graph history. Results are qualitatively similar for the other datasets. Overall, we observed that, in contradistinction to the experiments with real datasets, the *contiguous* initialization (ITR$_C$) of the iterative O^{2}BFF-AA algorithm emerges as the best algorithm in many cases, slightly outperforming INC$_D$. This is indicative of *temporal locality* of dense subgraphs in these datasets, i.e., in these datasets dense subgraphs are usually alive in a few contiguous snapshots. This is especially evident in datasets from collaboration networks such as the *DBLP* datasets. We also notice that the incremental algorithms find solutions with density very close to that of the iterative algorithms. Finally, we also observe that as k increases the aggregate density of the solutions decreases. This again is explained by the fact that often dense subgraphs are only "alive" in a few snapshots.

Convergence and running time: In terms of convergence, the iterative algorithms required 2-6 iterations to converge in all datasets. In Fig. 8, we report the

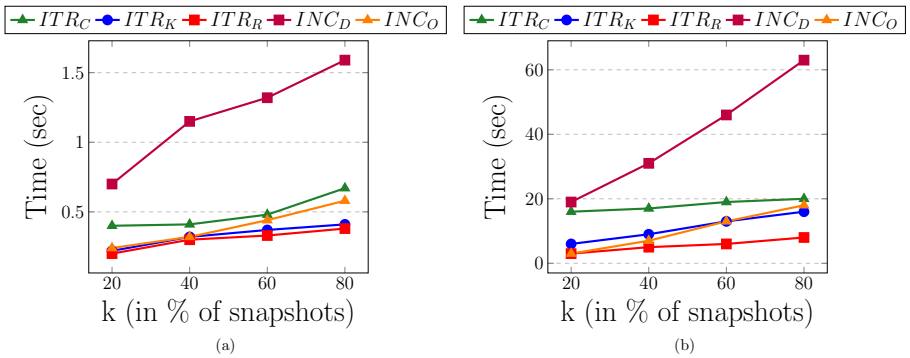


Fig. 8: Execution time of O^{2}BFF algorithms for the O^{2}BFF-MA problem for (a) the $DBLP_{10}$, and (b) the $Oregon_2$ datasets.

execution time of O^{2}BFF algorithms for the O^{2}BFF-MA problem for the $DBLP_{10}$, and $Oregon_2$ datasets. Results are qualitatively similar for the other datasets and O^{2}BFF problems. Both the iterative and incremental INC$_O$ algorithms scale well with k. Comparing between the incremental algorithms, INC$_O$ is up to 6x and 3.5x faster than INC$_D$ in the synthetic and the $Oregon_2$ datasets respectively due to the quadratic complexity of the latter.

Scalability: We also test the scalability of the algorithms in terms of both the size of the graphs and the number of snapshots using the synthetic datasets. For testing scalability with size, we create 10 graph snapshots with N nodes (for $N =$ 20,000 up to 100,000). Then, in each one of the 10 snapshots we plant at half of the snapshots a dense random subgraph A with 100 nodes each by inserting extra edges with probability $p_A = 0.9$. We consider subgraph A as our ground truth. We report the average execution time (and variance) of the different algorithms for the O^{2}BFF-MA problem with $k = 50\%$ in Fig. 9(a), when trying to recover A. For testing scalability with the number of graph snapshots, we create T snapshots of a graph with 50,000 nodes for $T = 10$ up to 50 snapshots, as described previously. We report the average execution time (and variance) in Fig. 9(b). In terms of scalability, INC$_O$ scales well with both the number of nodes and snapshots and clearly outperforms INC$_D$.

Summary: In conclusion, all algorithms successfully discovered the planted dense subgraphs that lasted a sufficient percentage (much less than half) of the snapshots with the incremental ones being more sensitive. The incremental algorithms outperform the iterative ones in most cases. Among the incremental algorithms, INC$_D$ is slightly better than INC$_O$. However, given the slow running time of INC$_D$, INC$_O$ offers an attractive alternative. Finally, in datasets consisting of dense subgraphs with temporal locality, ITR$_C$ is a good choice for detecting such graphs.

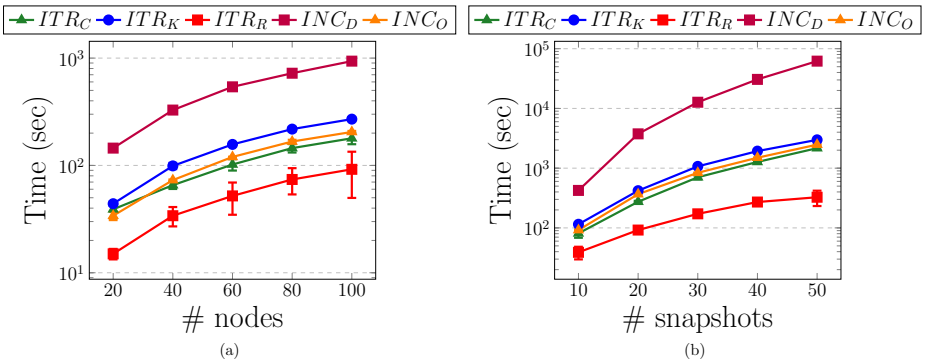


Fig. 9: Synthetic dataset ($p_A = 0.9$): execution time (log scale) of O^{2}BFF-MA problem for varying number of (a) nodes, and (b) snapshots.

5.3 Case studies

In this section, we report indicative results we obtained using the $DBLP_{10}$ and the *Twitter* datasets. These results identify lasting dense author collaborations and hashtag co-occurrences respectively.

Lasting dense co-authorships in $DBLP_{10}$: In Table 7, we report the set of nodes output as solutions to the different BFF problem variants, on the $DBLP_{10}$ dataset.

First, observe that three authors "Wei Fan", "Philip S. Yu", and "Jiawei Han" are part of *all* four solutions. These three authors have co-authored only two papers together in our dataset, but pairs of them have collaborated very frequently over the last decade. The solutions for BFF-AM and BFF-AA contain additional collaborators of these authors. For BFF-AA we obtain a solution of 8 authors. Although, this group has no paper in which they are all co-authors, subsets of the authors have collaborated with each other in many snapshots, resulting in high value of f_{aa}. The solutions for BFF-MM and BFF-MA contain the aforementioned three authors and some of their collaborators, but also some new names. These are authors that have scarce or no collaborations with the former group. Thus, in this case, the solutions consist of more than one dense subgroups of authors (grouped in parentheses), that are densely connected within themselves, but sparsely or not connected with others, while this is not the case for BFF-AM and BFF-AA.

Lasting dense hashtag appearances in *Twitter*: In Table 8, we report results of the O^{2}BFF problem on the *Twitter* dataset. Note that the results of the BFF problem on this dataset (as shown in Tables 2, 3, and 4) are very small graphs, since very few hashtags appear together in all 15 days of the dataset. As seen in Table 8, we were able to discover interesting dense subgraphs of hashtags appearing in k = 3, 6, and 9 of these days. These hashtags correspond to actual events (including f1 races, the tpp agreement and wikileaks) that were trending during that period.

Note also, that for large values of k, we do not get interesting results which is a fact consistent with the ephemeral nature of Twitter, where hashtags are short-lived. This is especially true for f_{mm} and f_{ma} that impose strict density constraints and as a result the solutions consist of disconnected edges.

Table 7: The BFF solutions for $DBLP_{10}$ (in parenthesis dense author subgroups)

BFF-MM
(Wei Fan, Philip S. Yu, Jiawei Han, Charu C. Aggarwal), (Lu Qin, Jeffrey Xu Yu, Xuemin Lin), (Guoliang Li, Jianhua Feng), (Craig Macdonald, Iadh Ounis)
BFF-MA
(Wei Fan, Jing Gao, Philip S. Yu, Jiawei Han, Charu C. Aggarwal), (Jeffrey Xu Yu, Xuemin Lin, Ying Zhang)
BFF-AM
(Wei Fan, Jing Gao, Philip S. Yu, Jiawei Han)
BFF-AA
(Wei Fan, Jing Gao, Philip S. Yu, Jiawei Han, Charu C. Aggarwal, Mohammad M. Masud, Latifur Khan, Bhavani M. Thuraisingham)

For each solution, we also report the selected snapshot dates. As expected there is time-contiguity in the selected dates, but our approach also captures the interest fluctuation over time. For example, for the wikileaks topic that is captured in the dense hashtag set {"wikileaks", "snowden", "nsa", "prism"}, the best snapshots are collections of contiguous intervals, rather than a single contiguous interval.

When comparing the results of the different variants of the O^{2}BFF problem, we see that the variants that consider average density over time (i.e., O^{2}BFF-AA and O^{2}BFF-AM) return much larger solutions than the variants that impose strict density requirement at each and every snapshot (i.e., O^{2}BFF-MM and O^{2}BFF-MA). For large k, the returned subgraphs refer to the "wikileaks" topic, while for small k, all variants, but O^{2}BFF-AM, return subgraphs that refer to the "f1" topic, indicating that "wikileaks" was loosely trending for a longer period, as opposed to "f1" for which we get dense subgraphs for smaller periods. O^{2}BFF-AM poses a requirement on the average minimum density and returns, for $k = 6$, a "wikileaks" subgraph consistent with the longer trending of this topic. For $k = 3$, it finds a large "tpp" subgraph whose average density may be smaller than the large "f1" subgraph found by O^{2}BFF-AA but all of its nodes are sufficiently connected with every other node in this "tpp" subgraph.

6 Related work

To the best of our knowledge, we are the first to systematically study all the variants of the BFF, and O^{2}BFF problems.

The research most related to ours is the recent work of Jethava and Beerenwinkel (2015) and Rozenshtein et al. (2014, 2017). To the best of our understanding, Jethava and Beerenwinkel (2015) consider one of the four variants of the BFF problem we studied here, namely, BFF-MA. In their paper, the authors conjecture that the problem is NP-hard and they propose a heuristic algorithm. Our work performs a rigorous and systematic study of the general BFF problem for multiple variants of the aggregate density function. We have also compared experimentally their DCS algorithm for the BFF-MA problem with our algorithms and shown that DCS is outperformed by the much faster FindBFF$_{\text{A}}$ algorithm. Additionally, we introduce and study the O^{2}BFF problem, which is not studied by Jethava and Beerenwinkel (2015). Rozenshtein et al. (2014) study a problem

Table 8: The hashtags and the chosen snapshot dates output as solutions to the O^{2}BFF problem on *Twitter*.

k = 3	O^{2}BFF-MM, O^{2}BFF-MA	O^{2}BFF-AM	O^{2}BFF-AA
	kimi, abudhabigp, f1, allowin	ozpol, nz, mexico, malaysia, signapore, vietnam, chile, peru, tpp, japan, canada	abudhabigp, fp1, abudhabi, guti, f1, pushpush, skyf1, hulk, allowin, bottas, kimi, fp3, fp2
Dates:	*Oct 31-Nov 2*	*Oct 27-28, Nov 7*	*Oct 31-Nov 2*
Density:	$f_{mm} = 3.0$, $f_{ma} = 3.25$	$f_{am} = 3.33$	$f_{aa} = 4.15$
k = 6	O^{2}BFF-MM, O^{2}BFF-MA	O^{2}BFF-AM	O^{2}BFF-AA
	abudhabigp, f1, skyf1	wikileaks, snowden, nsa, prism	abudhabigp, fp1, abudhabi, guti, f1, pushpush, skyf1, hulk, allowin, bottas, kimi, fp3, fp2
Dates:	*Oct 28-Nov 2*	*Oct 27-28, Nov 3,5,7*	*Oct 28, Oct 30-Nov 1, Nov 9*
Density:	$f_{mm} = 1.0$, $f_{ma} = 1.33$	$f_{am} = 2.0$	$f_{aa} = 2.35$
k = 9	O^{2}BFF-MM, O^{2}BFF-MA	O^{2}BFF-AM	O^{2}BFF-AA
	(No lasting graph found)	wikileaks, snowden, nsa, prism	assange, wikileaks, snowden, nsa, prism
Dates:		*Oct 27-31, Nov 3,5-7*	*Oct 27-29,31, Nov 3,5-7,10*
Density:		$f_{am} = 1.33$	$f_{aa} = 2.13$
k = 12	O^{2}BFF-MM, O^{2}BFF-MA	O^{2}BFF-AM	O^{2}BFF-AA
	(No lasting graph found)	wikileaks, snowden, nsa	assange, wikileaks, snowden, nsa, prism
Dates:		*Oct 27-Nov 1, Nov 3-7,10*	*Oct 27-31, Nov 2-7, 10*
Density:		$f_{am} = 1.33$	$f_{aa} = 1.76$

that can be considered as a special case of the O^{2}BFF problem. In particular, their goal is to identify a subset of nodes that are dense in the graph consisting of the union of edges appearing in the selected snapshots, which is a weak definition of aggregate density. Furthermore, they focus on finding collections of contiguous intervals, rather than arbitrary snapshots. They propose an algorithm similar to the iterative algorithm we consider, which we have shown to be outperformed by the incremental algorithms.

There is a huge literature on extracting "dense" subgraphs from a single graph snapshot. Most formulations for finding subgraphs that define near-cliques are often NP-hard and often hard to approximate due to their connection to the maximum-clique problem: Alvarez-Hamelin et al. (2005); Bourjolly et al. (2002); Makino and Uno (2004); McClosky and Hicks (2012); Tsourakakis et al. (2013). As a result, the problem of finding the subgraph with the maximum average or minimum degree has become particularly popular, due to its computational tractability. Specifically, the problem of finding a subgraph with the maximum average degree can be solved optimally in polynomial time (e.g., Charikar (2000); Goldberg (1984); Khuller and Saha (2009)), and there exists a practical greedy algorithm that gives a 2-approximation guarantee in time linear to the number of edges and nodes of the input graph (Charikar (2000)). The problem of identifying a subgraph with the maximum minimum degree, can be solved optimally in polynomial time (Sozio and Gionis (2010)), using again the greedy algorithm proposed by Charikar (2000). In our work, we use the average and minimum degree to quantify the density of the subgraph in a single graph snapshot, and we extend these definitions to sets of snapshots. The algorithmic techniques we use for the BFF problem are inspired by the techniques proposed by Charikar (2000), and by Sozio and Gionis (2010); however, adapting them to handle multiple snapshots is non-trivial.

Existing work also studies the problem of identifying a dense subgraph on dynamic time-evolving graphs: Epasto et al. (2015); Bahmani et al. (2012); Bhattacharya et al. (2015). These are graphs where new nodes and edges may appear over time and existing ones may disappear. The goal in this line of work is to devise a *streaming algorithm* that at any point in time it reports the densest subgraph for the current version of the graph. In our work, we are not interested in the dynamic version of the problem and thus the algorithmic challenges that our problem raises are orthogonal to those faced by the work on streaming algorithms.

Other related work focuses on detecting heavy, or dense, subgraphs in a special class of temporal weighted graphs with fixed nodes and edges, where only edge weights change over time and may take both positive and negative values (e.g., Bogdanov et al. (2011); Ma et al. (2017)). A filter-and-verify approach was proposed by Bogdanov et al. (2011), while a more scalable data-driven approach was recently introduced by Ma et al. (2017). The problem addressed in this work is different, since the set of edges is fixed, while we consider graphs with changing edge sets. Furthermore, density in the presence of edges with negative weights has different semantics.

Discovering evolving communities in graphs has also received a lot of attention (e.g., see Spiliopoulou (2011) and Fortunato (2009) for surveys). In this paper, we are interested in a more specific problem, that of identifying the densest subgraph over time, which in some sense can be viewed as a special type of a tightly-knit evolving community. Various approaches have been proposed for discovering communities in time-evolving graphs including incremental tensor analysis (e.g., Araujo et al. (2016)).

An interesting line of work casts the problem of finding dense subgraphs as a problem of frequent closed set discovery in ternary relations, or boolean tensors (e.g., Cerf et al. (2008); Nguyen et al. (2011, 2013)). In this setting an "itemset" is defined in the node space, and the support is defined over time. The goal is to find itemsets that appear frequently in time. This can be used to find dense subgraphs over multiple snapshots (similar to the O^2BFF problem), but it requires that the edges of the discovered subgraph appear in all snapshots, which is not necessarily the case in our setting.

Finally, another line of research focuses on processing queries e.g., reachability, path distance, graph matching, etc. over multiple graph snapshots: Semertzidis et al. (2015); Moffitt and Stoyanovich (2016); Semertzidis and Pitoura (2016, 2017, 2018); Khurana and Deshpande (2013); Ren et al. (2011). The main goal of this work is to devise effective storage, indexing and retrieving techniques so that queries over such sequences of graphs are answered efficiently. In this paper, we propose a novel problem that of finding dense subgraphs.

7 Summary

In this paper, we introduced and systematically studied the problem of identifying dense subgraphs in a collection of graph snapshots defining a graph history. We showed that for many definitions of aggregate density functions the problem of identifying a subset of nodes that are densely-connected in *all* snapshots (i.e., the BFF problem) can be solved in linear time. We also demonstrated that other versions of the BFF problem (i.e., BFF-MA and BFF-AM) cannot be solved with

the same algorithm. To identify dense subgraphs that occur in k, yet not all, the snapshots of a graph history we also defined the O^2BFF problem. For all variants of this problem we showed that they are NP-hard and we devised an iterative and an incremental algorithm for solving them. Our extensive experimental evaluation with datasets from diverse domains demonstrated the effectiveness and the efficiency of our algorithms.

References

J. Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Advances in Neural Information Processing Systems 18 [Neural Information Processing Systems, NIPS 2005, December 5-8, 2005, Vancouver, British Columbia, Canada]*, pages 41–50, 2005.

Miguel Araujo, Stephan Günnemann, Spiros Papadimitriou, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E. Papalexakis, and Danai Koutra. Discovery of "comet" communities in temporal and labeled graphs com^2. *Knowl. Inf. Syst.*, 46(3):657–677, 2016. doi: 10.1007/s10115-015-0847-2.

Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. Greedily finding a dense subgraph. volume 34, pages 203–221, 2000. doi: 10.1006/jagm.1999.1062.

Bahman Bahmani, Ravi Kumar, and Sergei Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5):454–465, 2012. doi: 10.14778/2140436.2140442.

Sayan Bhattacharya, Monika Henzinger, Danupon Nanongkai, and Charalampos E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 173–182, 2015. doi: 10.1145/2746539.2746592.

Petko Bogdanov, Misael Mongiovì, and Ambuj K. Singh. Mining heavy subgraphs in time-evolving networks. In *11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011*, pages 81–90, 2011. doi: 10.1109/ICDM.2011.101.

Jean-Marie Bourjolly, Gilbert Laporte, and Gilles Pesant. An exact algorithm for the maximum k-club problem in an undirected graph. *European Journal of Operational Research*, 138(1):21–28, 2002. doi: 10.1016/S0377-2217(01)00133-3.

Loïc Cerf, Jérémy Besson, Céline Robardet, and Jean-François Boulicaut. Data peeler: Contraint-based closed pattern mining in n-ary relations. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, pages 37–48, 2008. doi: 10.1137/1.9781611972788.4.

Moses Charikar. Greedy approximation algorithms for finding dense components in a graph. In *Approximation Algorithms for Combinatorial Optimization, Third International Workshop, APPROX 2000, Saarbrücken, Germany, September 5-8, 2000, Proceedings*, pages 84–95, 2000. doi: 10.1007/3-540-44436-X_10.

Alessandro Epasto, Silvio Lattanzi, and Mauro Sozio. Efficient densest subgraph computation in evolving graphs. In *Proceedings of the 24th International Conference on World Wide Web, WWW 2015, Florence, Italy, May 18-22, 2015*, pages 300–310, 2015. doi: 10.1145/2736277.2741638.

Santo Fortunato. Community detection in graphs. *CoRR*, abs/0906.0612, 2009. URL `http://arxiv.org/abs/0906.0612`.

Andrew V Goldberg. Finding a maximum density subgraph. Technical report, 1984.

Vinay Jethava and Niko Beerenwinkel. Finding dense subgraphs in relational graphs. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part II*, pages 641–654, 2015. doi: 10.1007/978-3-319-23525-7_39.

Samir Khuller and Barna Saha. On finding dense subgraphs. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, pages 597–608, 2009. doi: 10.1007/978-3-642-02927-1_50.

Udayan Khurana and Amol Deshpande. Efficient snapshot retrieval over historical graph data. In *29th IEEE International Conference on Data Engineering, ICDE 2013, Brisbane, Australia, April 8-12, 2013*, pages 997–1008, 2013. doi: 10.1109/ICDE.2013.6544892.

Jure Leskovec, Jon M. Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *TKDD*, 1(1):2, 2007. doi: 10.1145/1217299.1217301.

Shuai Ma, Renjun Hu, Luoshu Wang, Xuelian Lin, and Jinpeng Huai. Fast computation of dense temporal subgraphs. In *33rd IEEE International Conference on Data Engineering, ICDE 2017, San Diego, CA, USA, April 19-22, 2017*, pages 361–372, 2017. doi: 10.1109/ICDE.2017.95.

Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Algorithm Theory - SWAT 2004, 9th Scandinavian Workshop on Algorithm Theory, Humlebaek, Denmark, July 8-10, 2004, Proceedings*, pages 260–272, 2004. doi: 10.1007/978-3-540-27810-8_23.

Benjamin McClosky and Illya V. Hicks. Combinatorial algorithms for the maximum k-plex problem. *J. Comb. Optim.*, 23(1):29–49, 2012. doi: 10.1007/s10878-010-9338-2.

Vera Zaychik Moffitt and Julia Stoyanovich. Towards a distributed infrastructure for evolving graph analytics. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11-15, 2016, Companion Volume*, pages 843–848, 2016. doi: 10.1145/2872518.2889290.

Kim-Ngan Nguyen, Loïc Cerf, Marc Plantevit, and Jean-François Boulicaut. Multidimensional association rules in boolean tensors. In *Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, April 28-30, 2011, Mesa, Arizona, USA*, pages 570–581, 2011. doi: 10.1137/1.9781611972818.49.

Kim-Ngan Nguyen, Loïc Cerf, Marc Plantevit, and Jean-François Boulicaut. Discovering descriptive rules in relational dynamic graphs. *Intell. Data Anal.*, 17 (1):49–69, 2013. doi: 10.3233/IDA-120567.

Chenghui Ren, Eric Lo, Ben Kao, Xinjie Zhu, and Reynold Cheng. On querying historical evolving graph sequences. *PVLDB*, 4(11):726–737, 2011.

Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. Discovering dynamic communities in interaction networks. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, pages 678–693, 2014. doi: 10.1007/978-3-662-44851-9_43.

Polina Rozenshtein, Nikolaj Tatti, and Aristides Gionis. Finding dynamic dense subgraphs. *TKDD*, 11(3):27:1–27:30, 2017. doi: 10.1145/3046791.

Konstantinos Semertzidis and Evaggelia Pitoura. Durable graph pattern queries on historical graphs. In *32nd IEEE International Conference on Data Engineering, ICDE 2016, Helsinki, Finland, May 16-20, 2016*, pages 541–552, 2016. doi: 10.1109/ICDE.2016.7498269.

Konstantinos Semertzidis and Evaggelia Pitoura. Historical traversals in native graph databases. In *Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings*, pages 167–181, 2017. doi: 10.1007/978-3-319-66917-5_12.

Konstantinos Semertzidis and Evaggelia Pitoura. Top-k durable graph pattern queries on temporal graphs. volume 30, 2018. doi: 10.1109/TKDE.2018.2823754.

Konstantinos Semertzidis, Evaggelia Pitoura, and Kostas Lillis. Timereach: Historical reachability queries on evolving graphs. In *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015.*, pages 121–132, 2015. doi: 10.5441/002/edbt.2015.12.

Konstantinos Semertzidis, Evaggelia Pitoura, Evimaria Terzi, and Panayiotis Tsaparas. Best friends forever (BFF): finding lasting dense subgraphs. *CoRR*, abs/1612.05440, 2016.

Mauro Sozio and Aristides Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 939–948, 2010. doi: 10.1145/1835804.1835923.

Myra Spiliopoulou. Evolution in social networks: A survey. In *Social Network Data Analytics*, pages 149–175. 2011. doi: 10.1007/978-1-4419-8462-3_6.

Paraskevas Tsantarliotis and Evaggelia Pitoura. Topic detection using a critical term graph on news-related tweets. In *Proceedings of the Workshops of the EDBT/ICDT 2015 Joint Conference (EDBT/ICDT), Brussels, Belgium, March 27th, 2015.*, pages 177–182, 2015.

Charalampos E. Tsourakakis, Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Maria A. Tsiarli. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 104–112, 2013. doi: 10.1145/2487575.2487645.