A Change-Sensitive Algorithm for Maintaining Maximal Bicliques in a Dynamic Bipartite Graph

Apurba Das, Srikanta Tirthapura, Senior Member, IEEE.

Abstract—We consider the maintenance of maximal bicliques from a dynamic bipartite graph that changes over time due to the addition or deletion of edges. When the set of edges in a graph changes, we are interested in knowing the change in the set of maximal bicliques (the “change”), rather than in knowing the set of maximal bicliques that remain unaffected. The challenge in an efficient algorithm is to enumerate the change without explicitly enumerating the set of all maximal bicliques. In this work, we present (1) near-tight bounds on the magnitude of change in the set of maximal bicliques of a graph, due to a change in the edge set (2) a “change-sensitive” algorithm for enumerating the change in the set of maximal bicliques, whose time complexity is proportional to the magnitude of change that actually occurred in the set of maximal bicliques in the graph. To our knowledge, these are the first algorithms for enumerating maximal bicliques in a dynamic graph, with such provable performance guarantees. Our algorithms are easy to implement, and experimental results show that their performance exceeds that of current baseline implementations by orders of magnitude.

1 INTRODUCTION

Graphs are ubiquitous in representing linked data in many domains such as in social network analysis, computational biology, and web search. Often, these networks are dynamic, where new connections are being added and old connections are being removed. The area of dynamic graph mining focuses on efficient methods for finding and maintaining significant patterns in a dynamic graph. In this work we focus on the maintenance of dense subgraphs within a dynamic graph.

Our work is motivated by many applications that require the maintenance of dense substructures from a dynamic graph. Angel et al. [6], propose an algorithm for identifying breaking news stories in real-time through dense subgraph mining of an evolving graph, defined on the co-occurrence of entities within messages in an online social network. [17] present methods for detecting communities among users in a microblogging platform through identifying dense structures in an evolving network representing connections among users. A sample of other applications of dense subgraph mining in networks include identification of communities in a social network [13, 23], identification of web communities [14, 28, 18], phylogenetic tree construction [11, 29, 34], communities in bipartite networks [19], genome analysis [26], and closed itemset mining [32, 20].

We consider the fundamental problem of maintaining maximal bicliques in a bipartite graph that is changing due to the addition or deletion of edges. Let $G = (L, R, E)$ be a simple undirected bipartite graph with its vertex set partitioned into $L$, $R$, and edge set $E \subseteq L \times R$. A biclique in $G$ is a bipartition $B = (X, Y)$, $X \subseteq L$, $Y \subseteq R$ such that each vertex in $X$ is connected to each vertex in $Y$. A biclique $B$ is called a maximal biclique if there is no other biclique $B'$ such that $B$ is a proper subgraph of $B'$. Let $BC(G)$ denote the set of all maximal bicliques in $G$.

Suppose that, starting from bipartite graph $G_1 = (L, R, E)$, the state of the graph changes to $G_2 = (L, R, E \cup H)$ due to the addition of a set of new edges $H$. Let $\Upsilon_{\text{new}}(G_1, G_2) = BC(G_2) \setminus BC(G_1)$ denote the set of new maximal bicliques that arise in $G_2$ that were not present in $G_1$ and $\Upsilon_{\text{del}}(G_1, G_2) = BC(G_1) \setminus BC(G_2)$ denote the set of maximal bicliques in $G_1$ that are no longer maximal in $G_2$ (henceforth called as subsumed bicliques).

See Fig. 1 for an example. Let $\Upsilon(G_1, G_2) = \Upsilon_{\text{new}}(G_1, G_2) \cup \Upsilon_{\text{del}}(G_1, G_2)$ denote the symmetric difference of $BC(G_1)$ and $BC(G_2)$. We ask the following questions:

1. How large can be the size of $\Upsilon(G_1, G_2)$? In particular, can a small change in the set of edges cause a large change in the set of maximal bicliques in the graph?

2. How can we compute $\Upsilon(G_1, G_2)$ efficiently? Can we quickly compute $\Upsilon(G_1, G_2)$ when $|\Upsilon(G_1, G_2)|$ is small? In short, can we design change-sensitive algorithms for enumerating elements of $\Upsilon(G_1, G_2)$, whose time complexity is proportional to the size of change, $|\Upsilon(G_1, G_2)|$?

1.1 Contributions

Magnitude of Change: Let $g(n)$ denote the maximum number of maximal bicliques possible in an $n$ vertex bipartite graph. A result due to Prisner [27] shows that $g(n) \leq 2n^2/2$, where equality occurs when $n$ is even. We show that the change in the number of maximal bicliques when a single edge is added to the graph can be as large as $3g(n-2) \approx 1.5 \times 2n^2/2$, which is exponential in the number of vertices in the graph. This shows that the addition of even a single edge to the graph can lead to a large change in the set of maximal bicliques in the graph. We further show that this bound is tight for the case of the addition of a single edge – the largest possible change in the set of maximal bicliques upon adding a single edge is $3g(n-2)$. For the case when more edges can be added to the graph,
from Fig. 1: Change in maximal bicliques when the graph changes
{\{a, y\}, \{c, x\}}. Note that each maximal biclique of \(G_1\) is
subsumed by a larger maximal biclique in \(G_2\), and there is
one new maximal biclique in \(G_2\).

it is easy to see that the maximum possible change is no
larger than \(2g(n)\).

Enumeration Algorithm: From our analysis, it is clear that
the magnitude of change in the set of maximal bicliques
in the graph can be as large as exponential in \(n\) in the
worst case. On the flip side, the magnitude of change can
be as small as 1 – for example, consider the case when a
newly arriving edge connects two isolated vertices in the
graph. Thus, there is a wide range of values the magnitude
of change can take. When the magnitude of change is
very large, an algorithm that enumerates the change must
inevitably pay a large cost, if only to enumerate the change.
On the other hand, when the magnitude of change is small,
it will ideally pay a smaller cost. This motivates our search
for a change-sensitive algorithm whose computational cost
for enumerating the change is proportional to the magni-
tude of the change in the set of maximal bicliques.

We present a change-sensitive algorithm, DynamicBC,
for enumerating the new maximal bicliques and subsumed
maximal bicliques, when a set of new edges \(H\) are added
to the bipartite graph \(G\). The algorithm DynamicBC has
two parts, NewBC, for enumerating new maximal bicliques,
and SubBC, for enumerating subsumed maximal bicliques.
When a batch of new edges \(H\) of size \(\rho\) is added to the
graph, the time complexity of NewBC for enumerating \(T^\text{new}\),
the set of new maximal bicliques, is \(O(\Delta^2\rho|T^\text{new}|)\) where
\(\Delta\) is the maximum degree of the graph after update. The
time complexity of SubBC for enumerating \(T^\text{del}\), the set
of subsumed bicliques, is \(O(2^\rho|T^\text{new}|)\). To the best of our
knowledge, these are the first provably change-sensitive
algorithms for maintaining maximal bicliques in a dynamic
graph.

Experimental Evaluation: We present an empirical eval-
uation of our algorithms on real bipartite graphs with mil-
lion of nodes. Our results shows that the performance of
our algorithms are orders of magnitude faster than current
approaches. For example, on the actor-movie-1 graph
with 640K vertices and 1.4M edges, our algorithm took
about 30 milliseconds for computing the change due to
the addition of a batch of 100 edges, while the baseline
algorithm took more than 30 minutes.

1.2 Related Work

Maximal Biclique enumeration (MBE) on a static graph:
There has been substantial prior work on enumerating
maximal bicliques from a static graph. Alexe et al. \[5\]
propose an algorithm for MBE from a static graph based on
the consensus method, whose time complexity is proportional
to the size of the output (number of maximal bicliques
in the graph) - termed as output-sensitive algorithm. Liu et
al. \[22\] propose an algorithm for MBE based on depth-
first-search (DFS). Damaschke \[7\] propose an algorithm for
biclique graphs with a skewed degree distribution. Gely et
al. \[13\] propose an algorithm for MBE through a reduction
to maximal clique enumeration (MCE). However, in their
work, the number of edges in the graph used for enumera-
tion increases significantly compared to the original graph.

Dense Structures from Dynamic Graphs: There have
been some prior works related to maintenance of dense
structures similar to maximal bicliques in dynamic graphs.

Kumar et al. \[18\] define \((i, j)\)-core which is a biclique with
\(i\) vertices in one partition and \(j\) vertices in another partition.
In their work, the authors propose a dynamic algorithm
for extracting non-overlapping maximal set of \((i, j)\)-cores
for interesting communities. \[30\], \[21\], \[16\] present methods
for maintaining \(k\)-cores and \(k\)-trusses in a dynamic graph,
and \[8\] present algorithms for maintaining maximal cliques
in a dynamic graph.

Roadmap: The remaining section are organized as fol-
lows. We present definitions and preliminaries in Section 2.
Then we describe our algorithms in Section 3 and results
on the size of change in the set of maximal bicliques in Section 4.

2 Preliminaries

Let \(V(G)\) denote the set of vertices of \(G\) and \(E(G)\) the set
of edges in \(G\). Let \(n\) and \(m\) denote the number of vertices
and number of edges in \(G\) respectively. Let \(\Gamma_G(u)\) denote
the set of vertices adjacent to vertex \(u\) in \(G\). If the graph \(G\)
is clear from the context, we use \(\Gamma(u)\) to mean \(\Gamma_G(u)\). For
an edge \(e = (u, v) \in E(G)\), let \(G - e\) denote the graph after
deleting \(e \in E(G)\) from \(G\) and \(G + e\) denote the graph after
adding \(e \notin E(G)\) to \(G\). For a set of edges \(H\), let \(G + H\)
\((G - H)\) denote the graph obtained after adding (deleting)
\(H\) to (from) \(E(G)\). Similarly, for a vertex \(v \notin V(G)\), let \(G + v\)
denote the graph after adding \(v\) to \(G\) and for a vertex \(v \in
V(G)\), let \(G - v\) denote the graph after deleting \(v\) and all its
adjacent edges from \(E(G)\). Let \(\Delta(G)\) denote the maximum

\[
\begin{array}{c|c|c}
\hline
\text{Maximal bicliques: } & \text{New maximal bicliques: } \& \text{Subsumed bicliques: } \\
\{a, b\}, \{x\}, \{b, c\}, \{y\} & \{a, b, c\}, \{x, y\} & \{a, b\}, \{x\}, \{b, c\}, \{y\} \\
\{b\}, \{x, y\} & & \{b, x\}, \{x, y\} \\
\hline
\end{array}
\]

Fig. 1: Change in maximal bicliques when the graph changes
from \(G_1\) to \(G_2\) due to the addition of edge set \(H =
\{\{a, y\}, \{c, x\}\}\). Note that each maximal biclique of \(G_1\) is
subsumed by a larger maximal biclique in \(G_2\), and there is
one new maximal biclique in \(G_2\).
degree of a vertex in $G$ and $\delta(G)$ the minimum degree of a vertex in $G$.

**Definition 1** (Change-Sensitive Algorithm). An algorithm for a dynamic graph stream is called change-sensitive if its time complexity of enumerating the change in the graph property is proportional to the magnitude of change.

**Results for a static graph.** In [27], Prisner presented the following result on the number of maximal bicliques in a bipartite graph with $n$ vertices.

**Theorem 1** (Theorem 2.1 [27]). Every bipartite graph with $n$ vertices contains at most $2^n \approx 1.41^n$ maximal bicliques, and the only extremal (maximal) bipartite graphs are the graphs $CP(k)$.

Here, $CP(k)$ denotes the cocktail-party graph which is a bipartite graph with $k$ vertices in each partition where $V(CP(k)) = \{a_1, a_2, \ldots, a_k, b_1, b_2, \ldots, b_k\}$ and $E(CP(k)) = \{(a_i, b_i) : i \neq p\}$ [27]. See Figure 2 for an example.

As a subroutine, we use an algorithm for enumerating maximal bicliques from a static undirected graph, whose runtime is proportional to the number of maximal bicliques. There are a few algorithms of this kind [5], [22], [35]. We use the following result due to Liu et al. [22] as it provides best possible time and space complexity.

**Theorem 2** (Liu et al., [22]). For a graph $G$ with $n$ vertices, $m$ edges, maximum degree $\Delta$, and number of maximal bicliques $\mu$, there is an algorithm $\text{MineLMBC}$ for enumerating maximal bicliques in $G$ with time complexity $O(n\Delta \mu)$ and space complexity $O(m + \Delta^2)$.

$\text{MineLMBC}$ is depth-first-search (DFS) based algorithm for enumerating maximal bicliques of a static graph $G = (V, E)$. It takes as input the graph $G$ and the size threshold $s$. The algorithm enumerates all maximal bicliques of $G$ with size of each partition at least $s$. Clearly, by setting $s = 1$, the algorithm enumerates all maximal bicliques of $G$.

### 3 CHANGE-SENSITIVE ALGORITHM FOR MAXIMAL BICLIQUES

In this section, we present a change-sensitive algorithm $\text{DynamicBC}$ for enumerating the change in the set of maximal bicliques. The algorithm has two parts: (1) Algorithm $\text{NewBC}$ for enumerating new maximal bicliques, and (2) Algorithm $\text{SubBC}$ for enumerating subsumed bicliques.

For graph $G$ and set of edges $H$, we use $\Upsilon_{\text{new}}$ to mean $\Upsilon_{\text{new}}(G, G + H)$, and $\Upsilon_{\text{del}}$ to mean $\Upsilon_{\text{del}}(G, G + H)$.

**Algorithm 1:** $\text{DynamicBC}(G, H, BC(G))$

- **Input:** $G$ - Input bipartite graph, $H$ - Edges being added to $G$, $BC(G)$
- **Output:** $\Upsilon$ : the union of set of new maximal bicliques and subsumed bicliques

1. $\Upsilon_{\text{new}} \leftarrow \text{NewBC}(G, H)$
2. $\Upsilon_{\text{del}} \leftarrow \text{SubBC}(G, H, BC(G), \Upsilon_{\text{new}})$
3. $\Upsilon \leftarrow \Upsilon_{\text{new}} \cup \Upsilon_{\text{del}}$

We first present Algorithm $\text{NewBC}$ for enumerating new cliques in Section 3.1 and Algorithm $\text{NewBC}$ for enumerating subsumed cliques in Section 3.2. The main result on the time complexity of $\text{DynamicBC}$ is summarized in the following theorem.

**Theorem 3.** $\text{DynamicBC}$ is a change-sensitive algorithm for enumerating the change in the set of maximal bicliques, with time complexity $O(\Delta^2 \rho \Upsilon_{\text{new}} + 2^\rho \Upsilon_{\text{new}})$ where $\Delta$ is the maximum degree of a vertex in $G + H$ and $\rho$ is the size of $H$.

**3.1 Enumerating New Maximal Bicliques**

![Fig. 3: The original graph $G$ has 4 maximal bicliques. When new edges in $H$ (in dotted line) are added to $G$, all maximal bicliques in $G$ remain maximal in $G + H$ and only one maximal biclique is newly formed ($\leq \{a_3, a_4\}, \{b_3, b_4\}$).](image3)

Let $G'$ denote the graph $G + H$. A baseline algorithm for enumerating new maximal bicliques in $G'$ is to (1) enumerate all maximal bicliques in $G$, (2) enumerate all maximal bicliques in $G'$ both using an output-sensitive algorithm such as $\text{MineLMBC}$, and then (3) compute $BC(G') \setminus BC(G)$. However, this is not change-sensitive, since we need to compute all maximal bicliques of $G'$ each time, but it is possible that most of the maximal bicliques in $G'$ are not new. For example, see Fig. 3. We next present an approach that overcomes this difficulty.

For each new edge $e \in H$, let $BC'(e)$ denote the set of maximal bicliques in $G'$ containing edge $e$.

**Lemma 1.** $\Upsilon_{\text{new}} = \bigcup_{e \in H} BC'(e)$.

**Proof.** Each biclique in $\Upsilon_{\text{new}}$ must contain at least one edge from $H$. To see this, consider a biclique $b \in \Upsilon_{\text{new}}$. If $b$ did not contain an edge from $H$, then $b$ is also a maximal biclique in $G$, and hence cannot belong to $\Upsilon_{\text{new}}$. Hence, $b \in BC'(e)$ for some edge $e \in H$, and $b \in \bigcup_{e \in H} BC'(e)$. This shows that $\Upsilon_{\text{new}} \subseteq \bigcup_{e \in H} B'(e)$.
Fig. 4: Construction of $G'_e$ from $G' = G + H$ when a set of new edges $H = \{e, h\}$ is added to $G$. $A = \Gamma_{G'}(v) = \{u, x\}$ and $B = \Gamma_{G'}(u) = \{v, y\}$.

Next consider a biclique $b \in \cup_{e \in H} BC'(e)$. It must be the case that $b \in BC'(h)$ for some $h \in H$. Thus $b$ is a maximal biclique in $G + H$, and $b$ contains edge $h \in H$ and $b$ cannot be a biclique in $G$. Thus $b \in \Gamma^{new}$. This shows that $\cup_{e \in H} BC'(e) \subseteq \Gamma^{new}$.

Next, for each edge $e = (u, v) \in H$, we present an efficient way to enumerate all bicliques in $BC'(e)$ through enumerating maximal bicliques in a specific subgraph $G'_e$ of $G'$, constructed as follows. Let $A = \Gamma_{G'}(u)$ and $B = \Gamma_{G'}(v)$. Then $G'_e = (A, B, E')$ is a subgraph of $G'$ induced by vertices in $A$ and $B$. See Fig. 4 for an example of the construction of $G'_e$.

Lemma 2. For each $e \in H$, $BC'(e) = BC(G'_e)$

Proof. First we show that $BC'(e) \subseteq BC(G'_e)$. Consider a biclique $b = (X, Y)$ in $BC'(e)$. Let $e = (u, v)$. Here $b$ contains both $u$ and $v$. Suppose that $u \in X$ and $v \in Y$. According to the construction $G'_e$ contains all the vertices adjacent to $u$ and all the vertices adjacent to $v$. And in $b$, all the vertices in $X$ are connected to all the vertices in $Y$. Hence, $b$ is a biclique in $G'_e$. Also, $b$ is a maximal biclique in $G'_e$, and $G'_e$ is an induced subgraph of $G'$ which contains all the vertices of $b$. Hence, $b$ is a maximal biclique in $G'_e$.

Next we show that $BC(G'_e) \subseteq BC'(e)$. Consider a biclique $b' = (X', Y')$ in $BC(G'_e)$. Clearly, $b'$ contains $e$ as it contains both $u$ and $v$ and $b'$ is a maximal biclique in $G'_e$. Hence, $b'$ also a biclique in $G'$ that contains $e$. Now we prove that $b'$ is also maximal in $G'$. Suppose not, that there is a vertex $w \in V(G')$ such that $b'$ can be extended with $w$. Then, as per the construction of $G'_e$, $w \in V(G'_e)$ since $w$ must be adjacent to either $u$ or $v$. Then, $b'$ is not maximal in $G'_e$. This is a contradiction. Hence, $b'$ is also maximal in $G'$. Therefore, $b' \in BC'(e)$. \hfill \square

Based on the above observation, we present our change-sensitive algorithm $\texttt{NewBC}$ (Algorithm 2). We use an output-sensitive algorithm for a static graph $\texttt{MineLMBC}$ for enumerating maximal bicliques from $G'_e$. Note that typically, $G'_e$ is much smaller than $G'$ since it is localized to edge $e$, and hence enumerating all maximal bicliques from $G'_e$ should be relatively inexpensive.

Theorem 4. $\texttt{NewBC}$ enumerates the set of all new bicliques arising from the addition of $H$ in time $O(\Delta^2 \rho |\Gamma^{new}|)$ where $\Delta$ is the maximum degree of a vertex in $G'$ and $\rho$ is the size of $H$. The space complexity is $O(|E(G')| + \Delta^2)$.

Proof. First we consider correctness of the algorithm. From Lemma 1 and Lemma 2, we know that $\Gamma^{new}$ is enumerated by enumerating $BC(G'_e)$ for every $e \in H$. Our algorithm does this exactly, and use the $\texttt{MineLMBC}$ algorithm for enumerating $BC(G'_e)$.

For the runtime, consider that the algorithm iterates over each edge $e \in H$. In each iteration, it constructs a graph $G'_e$ and runs $\texttt{MineLMBC}(G'_e)$. Note that the number of vertices in $G'_e$ is no more than $2\Delta$, since it is the size of the union of the edge neighborhoods of $\rho$ edges in $G'$. The set of maximal bicliques generated in each iteration is a subset of $\Gamma^{new}$, therefore the number of maximal bicliques generated from each iteration is no more than $|\Gamma^{new}|$. From Theorem 2 we have that the runtime of each iteration is $O(\Delta^2 |\Gamma^{new}|)$. Since there are $\rho$ edges in $H$, the result on runtime follows. For the space complexity, we note that the algorithm does not store the set of new bicliques in memory at any point. The space required to construct $G'_e$ is linear in the size of $G'$. From Theorem 2 the total space requirement is $O(|E(G')| + \Delta^2)$. \hfill \square

### 3.2 Enumerating Subsumed Maximal Bicliques

We now present a change-sensitive algorithm for enumerating $BC(G) \setminus BC(G')$ where $G' = G + H$. Suppose a new maximal biclique $b$ of $G'$ subsumed a maximal biclique $b'$ of $G$. Note that $b'$ is also a maximal biclique in $b - H$. So, one idea is to enumerate all maximal bicliques in $b - H$ and then check which among them is maximal in $G$. However, checking maximality of a biclique is costly operation since we need to consider neighborhood of every vertex in the biclique. Another idea is to store the bicliques of the graph explicitly and see which among the generated bicliques are contained in the set of maximal bicliques of $G$. This is not desirable either since large amount of memory is required to store the set of all maximal bicliques of $G$.

A more efficient approach is to store the signatures of the maximal bicliques instead of storing the bicliques themselves. Then, we enumerate all maximal bicliques in $b - H$ and for each biclique generated, we compare the signature of the generated biclique with the signatures of the bicliques stored. An algorithm following this idea is presented in Algorithm 3. In this algorithm we reduce the cost of main memory by storing the signatures. We use a standard hash function (such as 64 bit murmur hash) for

1. https://sites.google.com/site/murmurhash/
computing signatures of maximal bicliques. For computing the signature, first we represent a biclique in canonical form (vertices in first partition represented in lexicographic order followed by vertices in another partition represented in lexicographic order). Then we convert the string into bytes, and apply hash function on the computed bytes. The hash function returns signature as output. By storing the signatures instead of maximal bicliques, we are able to check whether a maximal biclique from $b - H$ is contained in the set of maximal bicliques of $G$ by comparing their hash values. Thus we pay much less cost in terms of memory by storing the signatures of bicliques.

Now we prove that Algorithm \ref{alg:subBC} indeed enumerates all maximal bicliques of $b - H$.

**Lemma 3.** In Algorithm \ref{alg:subBC} for each $b \in \Upsilon^{new}$, $S$ after Line 14 contains all maximal bicliques in $b - H$.

**Proof.** First observe that, removing $H$ from $b$ is equivalent to removing those edges in $H$ which are present in $b$. Hence, computing maximal bicliques in $b - H$ reduces to computing maximal bicliques in $b - H_1$ where $H_1$ is the set of all edges in $H$ which are present in $b$.

We use induction on the number of edges $k$ in $H_1$. Consider the base case, when $k = 1$. $H_1$ contains a single edge $e_1 = \{u, v\}$. Clearly, $b - H_1$ has two maximal bicliques $b \setminus \{u\}$ and $b \setminus \{v\}$. Suppose, that the set $H_1$ is of size $k$. Our inductive hypothesis is that all maximal bicliques in $b - H_1$ are enumerated. Consider $H'_1 = \{e_1, e_2, \ldots, e_k, e_{k+1}\}$ with $k + 1$ edges. Now each maximal biclique $b' \in b - H_1$ either remains maximal within $b - H'_1$ (if at least one endpoint of $e_{k+1}$ is not in $b'$) or generates two maximal bicliques in $b - H'_1$ (if both endpoints of $e_{k+1}$ are in $b'$). Thus, for each $b \in \Upsilon^{new}$, $S$ after Line 14 contains all maximal bicliques within $b - H$.

Now we show that the algorithm described above is a change-sensitive algorithm for enumerating all elements of $\Upsilon^{del}$ when the number of edges $\rho$ in $H$ is constant.

**Theorem 5.** Algorithm \ref{alg:subBC} enumerates all bicliques in $\Upsilon^{del} = \mathcal{BC}(G) - \mathcal{BC}(G + H)$ using time $O(2^\rho |\Upsilon^{new}|)$ where $\rho$ is the number of edges in $H$. The space complexity of the algorithm is $O(|E(G')| + |V(G')| + \Delta^2 + |\mathcal{BC}(G)|)$.

**Proof.** We first show that every biclique $b'$ enumerated by the algorithm is indeed a biclique in $\Upsilon^{del}$. Note that $b'$ is a maximal biclique in $G$, due to explicitly checking the condition. Further, $b'$ is not a maximal biclique in $G + H$, since it is a proper subgraph of $b$, a maximal biclique in $G + H$. Next, we show that all bicliques in $\Lambda^{del}$ are enumerated. Consider any subsumed biclique $b' \in \Lambda^{del}$. It must be contained within $b \setminus H$, where $b$ is a maximal biclique within $\Lambda^{new}$. Moreover, $b'$ will be a maximal biclique within $b \setminus H$, and will be enumerated by the algorithm according to Lemma \ref{lemma:subBC}.

For the time complexity we show that for any $b \in \Upsilon^{new}$, the maximum number of maximal bicliques in $b - H$ is $2^\rho$ using induction on $\rho$. Suppose $\rho = 1$ so that $H$ contains a single edge, say $e_1 = \{u, v\}$. Then, $b - H$ has two maximal bicliques, $b \setminus \{u\}$ and $b \setminus \{v\}$, proving the base case. Suppose that for any set $H$ of size $k$, it was true that $b - H$ has no more than $2^k$ maximal bicliques. Consider a set $H'' = \{e_1, e_2, \ldots, e_{k+1}\}$ with $k + 1$ edges. Let $H'' = \{e_1, e_2, \ldots, e_k\}$. Subgraph $b - H''$ is obtained from $b - H'$ by deleting a single edge $e_{k+1}$. By induction, we have that $b - H''$ has no more than $2^k$ maximal bicliques. Each maximal biclique $b' \in b - H'$ either remains a maximal biclique within $b - H''$ (if at least one endpoint of $e_{k+1}$ is not in $b'$) or leads to two maximal bicliques in $b - H''$ (if endpoints of $e_{k+1}$ are in different partition of $b'$). Hence, the number of maximal bicliques in $b - H''$ is no more than $2^{k+1}$, completing the inductive step.

Following this, for each biclique $b \in \Upsilon^{new}$, we need to check for maximality for no more than $2^\rho$ bicliques in $G$. This checking can be performed by checking whether each such generated biclique in contained in the set $\mathcal{BC}(G)$ and for each biclique, this can be done in constant time.

For the space bound, we first note that in Algorithm \ref{alg:subBC} enumerating maximal bicliques within $b - H$ consumes space $O(|E(G')| + \Delta^2)$, and checking for maximality can be done in space linear in size of $G$. However, for storing the maximal bicliques in $G$ takes $O(|\mathcal{BC}(G)|)$ space. Hence, for these operations, the overall space-cost for each $b \in \Upsilon^{new}$ is $O(|E(G')| + |V(G')| + \Delta^2 + |\mathcal{BC}(G)|)$. The only remaining space cost is the size of $\Upsilon^{new}$, which can be large. Note that, the algorithm only itrates through $\Upsilon^{new}$ in a single pass. If elements of $\Upsilon^{new}$ are provided as a stream from the output of an algorithm such as NewBC, then they do not need to be stored within a container, so that the memory cost of receiving $\Upsilon^{new}$ is reduced to the cost of storing a single maximal biclique within $\Upsilon^{new}$ at a time.
3.3 Decremental and Fully Dynamic Cases

We now consider the maintenance of maximal bicliques in the decremental case, when edges are deleted from the graph. This case can be handled using a reduction to the incremental case. We show that the maintenance of maximal bicliques due to deletion of a set of edges \( H \) from a bipartite graph \( G \) is equivalent to the maintenance of maximal bicliques due to addition of \( H \) to the bipartite graph \( G - H \).

**Lemma 6.** \( \forall n \in \mathbb{N} \), the maximum size of \( |BC(G, G + e)| \) is \( 3g(n - 2) \) when \( e = (u, v) \) is inserted to \( G \).

**Proof.** We prove this theorem in the following two lemmas. In Lemma 5 we prove that the size of \( \Upsilon(G, G + e) \) can be as large as \( 3g(n - 2) \) in Lemma 9 we prove that the size of \( \Upsilon(G, G + e) \) is at most \( 3g(n - 2) \).
does not contain \(e\). Then \(b\) must be present in \(G\) but is not maximal in \(G\), and there must be another vertex \(w \in L\) (or \(R\)) that can be added to \(b\) while remaining a biclique. Clearly, \(w\) can be added to biclique \(b\) in \(G + e\) also, so that \(b\) is not maximal in \(G + e\), contradicting our assumption.

**Lemma 8.** If \(e = (u, v)\) is added to \(G\), each biclique \(b \in BC(G) - BC(G + e)\) contains either \(u\) or \(v\).

**Proof.** Proof by contradiction. Suppose there is maximal biclique \(b = (b_1, b_2) \in BC(G) - BC(G + e)\) that contain neither \(u\) nor \(v\). Then, \(b\) must be maximal biclique in \(G\). Since \(b\) is not maximal biclique in \(G + e\), \(b\) is contained in another maximal biclique \(b' = (b'_1, b'_2)\) in \(G + e\). From Lemma 7, \(b'\) must contain edge \(e = (u, v)\), and hence, both vertices \(u\) and \(v\). Since \(b'\) is a biclique, every vertex in \(b'_2\) is connected to \(v\) in \(G'\). Hence, every vertex in \(b_2\) is connected to \(u\) even in \(G\). Therefore, \(b' \cup \{u\}\) is a biclique in \(G\), and \(b\) is not maximal in \(G\), contradicting our assumption.

**Observation 1.** For a bipartite graph \(G = (L, R, E)\) and a vertex \(u \in V(G)\), the number of maximal bicliques that contains \(v\) is at most \(g(n - 1)\).

**Proof.** Suppose, \(u \in L\). Then each maximal biclique \(b\) in \(G\) that contains \(u\), corresponds to a unique maximal biclique in \(G - \{u\}\). Such maximal bicliques can be derived from \(b\) by deleting \(u\) from \(b\). As the maximum number of maximal bicliques in \(G - \{u\}\) is \(g(n - 1)\), maximum number of maximal bicliques in \(G\) can be no more than \(g(n - 1)\).

**Observation 2.** The number of maximal bicliques containing a specific edge \((u, v)\) is at most \(g(n - 2)\).

**Proof.** Consider an edge \((u, v) \in E(G)\). Let vertex set \(V' = \Gamma_G(u) \cup \Gamma_G(v) - \{u, v\}\), and let \(G'\) be the subgraph of \(G\) induced by \(V'\). Each maximal biclique \(b\) in \(G\) that contains \((u, v)\) corresponds to a unique maximal biclique in \(G'\) by simply deleting vertices \(u\) and \(v\) from \(b\). Also, each maximal biclique \(b'\) in \(G'\) corresponds to a unique maximal biclique in \(G\) that contains \((u, v)\) by adding vertices \(u\) and \(v\) to \(b'\). Thus, there is a bijection between the maximal bicliques in \(G'\) and the set of maximal bicliques in \(G\) that contains edge \((u, v)\). The number of maximal bicliques in \(G'\) can be at most \(g(n - 2)\) since \(G'\) has no more than \((n - 2)\) vertices, completing the proof.

**Lemma 9.** For a bipartite graph \(G = (L, R, E)\) on \(n\) vertices and edge \(e = (u, v) \in E(G)\), the size of \(\Upsilon(G, G + e)\) can be no larger than \(3g(n - 2)\).

**Proof.** Proof by contradiction. Suppose there exists a bipartite graph \(G = (L, R, E)\) and edge \(e \notin E(G)\) such that \(|\Upsilon(G, G + e)| \geq 3g(n - 2)\). Then either \(|BC(G + e) - BC(G)| \geq g(n - 2)\) or \(|BC(G) - BC(G + e)| \geq 2g(n - 2)\).

**Case 1:** \(|BC(G + e) - BC(G)| \geq g(n - 2)\): This means that total number of new maximal bicliques formed due to addition of edge \(e\) is larger than \(g(n - 2)\). From Lemma 7, each new maximal biclique formed due to addition of \(e\) must contain \(e\). From Observation 2, the total number of maximal bicliques in an \(n\) vertex bipartite graph containing a specific edge can be at most \(g(n - 2)\). Thus, the number of new maximal bicliques after adding edge \(e\) is at most \(g(n - 2)\), contradicting our assumption.

**Case 2:** \(|BC(G) - BC(G + e)| \geq 2g(n - 2)\): Using Lemma 8, each maximal biclique \(b \in BC(G) - BC(G + e)\) must contain either \(u\) or \(v\), but not both. Suppose that \(b\) contains \(u\) but not \(v\). Then, \(b\) must be a maximal biclique in \(G - v\). Using Observation 1, we see that the number of maximal bicliques in \(G - v\) that contains a specific vertex \(u\) is no more than \(g(n - 2)\). In a similar way, the number of possible maximal bicliques that contain \(v\) is at most \(g(n - 2)\). Therefore, the total number of maximal bicliques in \(BC(G) - BC(G + e)\) is at most \(2g(n - 2)\), contradicting our assumption.

Combining Lemma 7, Theorem 6, and using the fact that \(3g(n - 2) = 1.5g(n)\) for even \(n\), we obtain the following when \(n\) is even:

**Theorem 7.** \(1.5g(n) \leq \lambda(n) \leq 2g(n)\)

## 5 Experimental Evaluation

In this section, we present results of an experimental evaluation of our algorithms.

### 5.1 Data

We consider the following real-world bipartite graphs in our experiments. A summary of the datasets is presented in Table 1 in the actor-movie graph, vertices consist of actors in one bipartition and movies in another bipartition. There is an edge between an actor and a movie if the actor played in that movie. In the dblp-author graph, vertices consist of authors in one partition and the publications in another partition. Edges connect authors to their publications. In the opinions-rating graph, vertices consist of users in one partition and products in another partition. There is an edge between a user and a product if the user rated the product. Also, the edges have timestamps of their creation. In the flickr-membership graph, vertices consists of users and groups. There is an edge between a user and a group if that user is a member of that group.

We converted the above graphs into dynamic graphs by creating edge streams as follows: For actor-movie, dblp-author, and flickr-membership we created initial graphs by retaining each edge in the original graph with probability 0.1 and deleting the rest. Then the deleted edges are added back as an edge stream, until the original graph is reached. We named the initial graphs as actor-movie-1, dblp-author-1, and flickr-membership-1. For the opinions-rating graph, we created the initial graph by retaining initial 10\% edges of the original graph according to their timestamps, and considered rest of the edges for creating the edge stream in timestamp ordering. We named the initial graph as opinions-rating-init. In Table 1, the number of edges of the initial graph is in the column Edges(initial) and the number of edges when we end the experiment is in column Edges(final).

### 5.2 Experimental Setup and Implementation Details

We implemented our algorithms using Java on a 64-bit Intel(R) Xeon(R) CPU clocked at 3.10 Ghz and 8G DDR3 RAM with 6G heap memory space. Unless otherwise specified, we considered batches of size 100.
| Dataset                  | Nodes  | Edges(initial) | Edges(final) | Edges(original graph) | Avg. deg.(original graph) |
|-------------------------|--------|----------------|--------------|-----------------------|---------------------------|
| actor-movie-1           | 639286 | 146917         | 1470404      | 1470404               | 6                         |
| dblp-author-1           | 685176 | 864372         | 8649016      | 8649016               | 3                         |
| epinions-rating-init    | 996744 | 1366832        | 1631832      | 13668320              | 31                        |
| flickr-membership-1     | 895589 | 855179         | 1355179      | 8545307               | 35                        |

**TABLE 2: Comparison with Baseline: computation time for adding a single batch of size 100**

| Initial-graph            | DynamicBC | BaselineBC |
|--------------------------|-----------|------------|
| actor-movie-1            | 30 ms.    | > 30 min.  |
| dblp-author-1            | 20 ms.    | > 20 hours |
| epinions-rating-init     | 3.5 sec.  | > 10 hours |
| flickr-membership-1      | 0.5 sec.  | 1 hour     |

**Fig. 6: Computation time (in sec.) for enumerating the change in maximal bicliques, per batch of edges.**

**Metrics:** We evaluate our algorithms using the following metrics: (1) computation time for new maximal bicliques and subsumed bicliques when a set of edges are added, (2) change-sensitiveness, that is, the total computation time as a function of the size of change. We measure the size of change as the sum of the total number of edges in the new maximal bicliques and the subsumed bicliques, and (3) space cost, that is the memory used by the algorithm for storing the graph, and other data structures used by the algorithm, and (4) cumulative computation time for different batch sizes, that is the cumulative computation time from the initial graph to the final graph while using different batch size.

**5.3 Discussion of Results**

**Comparison with Baseline.** We compared the performance of our algorithm, DynamicBC, with a baseline algorithm for maintaining maximal bicliques, we have implemented
algorithm that we call BaselineBC. The baseline algorithm computes $\mathcal{Y}(G,G+H)$ by (1) Enumerating $\mathcal{BC}(G)$, (2) Enumerating $\mathcal{BC}(G+H)$, and (3) computing the difference of the two. We use MineMBBC [22] for enumerating bicliques from a static graph. Table 5 shows a comparison of the runtimes of DynamicBC and BaselineBC. From the table, it is clear that DynamicBC is faster than BaselineBC by two to three orders of magnitude. For instance, for adding a single batch of size 100 to actor-movie-1, BaselineBC takes more than 30 min., whereas DynamicBC takes around 30 ms.

Computational Time per Batch of Edges: Let an “iteration” denote the addition of a single batch of edges. Fig. 6 shows the computation time per iteration versus iteration number. From the plots, we observe that the computation time increases as the iteration increases. This trend is consistent with predictions. Note that as computation progresses, the number of edges in the graph increases, and note that the computation time is proportional to the size of graph as well as size of change (Theorem 3). In Fig. 6(c) we see that computation time decreases suddenly and then again increases. This may seem anomalous, but is explained by noting that in these cases, the magnitude of change decreases in those iterations, and then increases thereafter.

In Fig. 7 we show the breakdown of the computation time of DynamicBC into time taken for enumerating new cliques (NewBC) and for enumerating subsumed cliques (SubBC). Observe that the computation time increases for both new maximal bicliques and subsumed bicliques as more batches are added. This is because the graph becomes denser when more batches are added and the time taken to compute the change increases, consistent with Theorem 3.

Change-Sensitivity: Fig. 8 shows the computation time as a function of the size of change. We observe that the computation time of DynamicBC is roughly proportional to the size of change. The computation time of both NewBC and SubBC increases as number of new maximal bicliques and subsumed bicliques increases. Clearly, this observation supports our theoretical analysis. In some plots (Fig. 8(c) and (d)) we see a rapid increase in the computation time with the size of change. This is because, when the graph grows, memory consumption increases considerably and this affects the computation time of the algorithm.

Space Cost: Fig. 9 shows the space cost of DynamicBC for different graphs. As SubBC needs to maintain the maximal
bicliques in memory for computing subsumed bicliques, we report the space consumption in two cases: (1) when we store the maximal bicliques in memory, (2) when we store the signatures of bicliques in memory instead of storing the bicliques. Signatures consume less memory than the actual bicliques as the signatures have fixed size (64 bits in our case using the murmur hash function) for different sizes of bicliques. Therefore, memory consumption by the algorithm that uses signatures should be smaller than the algorithm that does not use signatures. The trend is also clear in the plots. The difference in memory consumption is not prominent during the initial iterations because, sizes of maximal bicliques are much smaller during initial iterations and therefore memory consumption is mainly due to the graph that we maintain in memory. We are not showing the space cost without hash for the third input graph because the algorithm could not execute on the third input graph without hashing, due to running out of memory.

Computation Time for Different Batch Size: Table 3 shows the cumulative computation time for different graphs when we use different batch size. We observe that the total computation time increases when increasing the batch size. The reason for this trend is that the computation time for subsumed cliques increases with increasing batch size, while the computation time for the new maximal bicliques remains almost same across different batch sizes. Note that, the time complexity for SubBC has (in the worst case) an exponential dependence on the batch size. Therefore, the computation time for subsumed cliques tends to increase with an increase in the batch size. However, with a very small batch size (such as 1 or 10), the change in the maximal bicliques is very small, and the overhead can be large.

Maintaining Large Maximal Bicliques: We also consider maintaining large maximal bicliques with predefined size threshold $s$, where it is required that each bipartition of the biclique has size at least $s$. For large subsumed bicliques, we provide $s$ in addition to other inputs to SubBC as well. Table 4 shows the cumulative computation time by varying the threshold size $s$ from 1 to 6. Clearly, $s = 1$ means that we maintain all maximal bicliques. As expected, the cumulative computation time decreases significantly in most of the cases as the size threshold $s$ increases.

6 Conclusion
In this work, we presented a change-sensitive algorithm for enumerating changes in the set of maximal bicliques in
dynamic graph. The performance of this algorithm is proportional to the magnitude of change in the set of maximal bicliques – when the change is small, the algorithm runs faster, and when the change is large, it takes a proportionally longer time. We present near-tight bounds on the maximum possible change in the set of maximal bicliques, due to a change in the set of edges in the graph. Our experimental evaluation shows that the algorithm is efficient in practice, and scales to graphs with millions of edges. This work leads to natural open questions (1) Can we design more efficient algorithms for enumerating the change, especially for enumerating subsumed cliques? (2) Can we parallelize the algorithm for enumerating the change in maximal bicliques?

**REFERENCES**

[1] Actor movies network dataset – KONECT. [http://konect.uni-koblenz.de/networks/actor-movie](http://konect.uni-koblenz.de/networks/actor-movie) Oct. 2016.
A. Java, X. Song, T. Finin, and B. L. Tseng. Why we twitter: An analysis of a microblogging community. In WebKDD/SNA-KDD, pages 118–138, 2007.

X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu. Querying k-truss community in large and dynamic graphs. In SIGMOD, pages 131–142, 2014.

A. Java, X. Song, T. Finin, and B. L. Tseng. Why we twitter: An analysis of a microblogging community. In WebKDD/SNA-KDD, pages 118–138, 2007.

R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. Computer networks, 31(11):1481–1493, 1999.

S. Lehmann, M. Schwartz, and L. K. Hansen. Biclique communities. Physical Review E, 78(1):016108, 2008.

J. Li, H. Li, D. Soh, and L. Wong. A correspondence between maximal complete bipartite subgraphs and closed patterns. In European Conference on Principles of Data Mining and Knowledge Discovery, pages 146–156, Springer, 2005.

R. Li, J. X. Yu, and R. Mao. Efficient core maintenance in large dynamic graphs. TKDE, 26(10):2453–2465, 2014.

G. Liu, K. Sim, and J. Li. Efficient mining of large maximal bicliques. In Data warehousing and knowledge discovery, pages 437–448. Springer, 2006.

D. Lo, D. Surian, K. Zhang, and E.-P. Lim. Mining direct antagonistic communities in explicit trust networks. In CIKM, pages 1013–1018, 2011.

K. Makino and T. Uno. New algorithms for enumerating all maximal cliques. In SWAT, pages 260–272, 2004.

A. P. Mukherjee and S. Tirthapura. Enumerating maximal bicliques from a large graph using mapreduce. In IEEE BigData Congres, pages 707–716, 2014.

N. Nagarajan and C. Kingsford. Uncovering genomic reassembly events among influenza strains by enumerating maximal bicliques. In Bioinformatics and Biomedicine, 2008. BIBM’08. IEEE International Conference on, pages 223–230. IEEE, 2008.

E. Prisner. Bicliques in graphs i: Bounds on their number. Combinatorica, 20(1):109–117, 2000.

J. E. Rome and R. M. Haralick. Towards a formal concept analysis approach to exploring communities on the world wide web. In Formal Concept Analysis, volume 3403 of LNCS, pages 33–48, 2005.