Enumerating Maximal Bicliques in Bipartite Graphs with Favorable Degree Sequences

Peter Damaschke*
Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@chalmers.se

Abstract

We propose an output-sensitive algorithm for the enumeration of all maximal bicliques in a bipartite graph, tailored to the case when the degree distribution in one partite set is very skewed. We accomplish a worst-case bound better than previously known general bounds if, e.g., the degree sequence follows a power law.

Keywords: analysis of algorithms, biclique enumeration, degree sequence, power law, hull operator

1 Introduction

A bipartite graph $H = (X, Y, E)$ with edges set E has its edges only between two vertex sets X and Y, but not inside these sets. A bipartite graph is complete, or a biclique, if E consists of all $|X| \cdot |Y|$ possible edges. A biclique within another bipartite graph is called a maximal biclique if it is not contained in a larger biclique. Enumeration of the maximal bicliques of a given biparite graph has important applications in data analysis, which have been reported at many places. As the number β of maximal bicliques can be exponential in the graph size, one important type of enumeration algorithms is output-sensitive algorithms whose time bounds are polynomial in the graph size and in the output size β. A stronger demand is that one may always want to output a new item, i.e., maximal biclique, after some polynomial delay (in the size of the graph only). However, in the present paper we are only concerned with the total running time. If not said otherwise, let n and m denote the number of vertices and edges, respectively, of the input graph, and let Δ be the maximal vertex degree.

Let us review known total running times from the literature about the problem. For general (not only bipartite) graphs, several incomparable time bounds are derived in [1], in particular, $O(n^2\beta^2)$ or alternatively $O(n^3\beta)$. The latter bound also appears in [2] and is later refined to $O(nm\beta)$ in [3]. This bound, in turn, also comes out in [6] from a different angle. A weighted and thresholded version of the problem is considered in [9] where an $O(n^2\beta)$ time bound is reported. A unifying view is presented in [4], however no better time bounds in our direction follow there. Finally, one of the results in [8] is an $O(\Delta^2\beta)$ time bound for the case of bipartite graphs. The extended abstract [5] deals with the bipartite case, too, but gives no explicit time bound.

*Tel. 0046-31-772-5405. Fax 0046-31-772-3663.

In the present paper we take advantage of skewed degree distributions in the bipartite graph $H = (X, Y, E)$, resulting in time bounds that can beat the aforementioned $O(\Delta^2\beta)$ under some circumstances. More technically, consider the following special case of a sorted degree sequence in one partite set X, which goes as $1/j^s$, that is, the jth highest degree in X is about a $1/j^s$ fraction of the highest degree. Here s is any constant with $1 \leq s < 2$. Then we achieve $O(\Delta k^{2-s}\beta)$ time, where $k = |X|$. If, furthermore, $k^{2-s} < \Delta$, then this is faster than $O(\Delta^2\beta)$. (We remark that the algorithm in [8] also outputs a new biclique after a delay of $O(\Delta^2)$ time, whereas we do not aim for a polynomial delay, and apparently it would be hard to achieve combined with our result.)

This type of time bounds is our main theoretical contribution. It seems relevant because just such non-uniform degree distributions appear in practice. We are mainly interested in applications where the vertices in Y represent many short texts, the vertices in X represent the words in these text snippets, and $xy \in E$ is an edge, if word x occurs (at least once) in text y. The texts can, e.g., be tweets, short news about events, comments in a forum, or reviews of hotels, restaurants, of products or artistic works. Combinations of words that occur frequently indicate topics and can serve as a basis for, e.g., clustering, opinion mining, or automatic summarization.

Now, the point is that rather few words appear very frequently (and these are not only stop words but also characteristic terms from the discussed domain, or evaluating phrases), whereas others are more occasional. There is empirical evidence [7] that word frequencies in random texts follow Zipf's law, i.e., they are proportional to $1/j^s$ with s close to 1. In text corpora focused on one theme one can expect more "hyper-Zipfian" distributions with $s > 1$, since now only a limited set of words is very frequent. Also $k^{2-s} < \Delta$ is easily fulfilled, as Δ is high (frequent words in many texts), whereas the number k of different words comes with an exponent below 1. In a preprocessing phase we can even omit rare words that are of no interest, and thus reduce k right from the beginning.

We remark that the degree sequence does not have to obey exactly some power law, and the algorithm itself does not depend on that. We only discussed this function for its mathematical simplicity, in order to get some "crisp" specific worst-case bound. Rather, the more general, somewhat informal conclusion is that we can enumerate the maximal bicliques faster than what earlier time bounds indicate, whenever the degree sequence is "more skewed" than a Zipf's law sequence with $s = 1$.

Our algorithm, while taking the degree sequence into account, still follows natural ideas and should also be easy to implement. Despite earlier works we present the algorithm from scratch because, of course, the details are important for the analysis. We also add some more tricks that do not further help the worst-case bound but are beneficial for certain instances.

2 Prefix Maxima in a Sequence of Sets

The enumeration algorithm in Section 3 will have to deal with a certain sequence of sets of vertices and, very roughly speaking, recognize which of them are already subsets of other sets early in the sequence. (See Definition 1 below for the precise statement.) This will be needed to avoid returning non-maximal bicliques. In this section we solve this task separately by a routine called PrefMax, such that we can later use this routine and focus on the main algorithm.

As a notational remark, $\subset$ denotes the proper subset relation, whereas $\subseteq$ also allows for equality of sets. Two sets not in $\subseteq$ relation are called incomparable.

Definition 1 *Let $S_1, \ldots, S_k$ be a sequence of finite sets which are not necessarily different, that is, $S_i = S_j$ for $i \neq j$ is allowed. We call S_j with $j \leq p$ a* prefix maximum *in $[1..p]$ if no S_i, $i \leq p$, satisfies $S_j \subset S_i$, and no S_i, $i < j$, satisfies $S_i = S_j$. We define $p(j)$ to be the largest p such that S_j is a prefix maximum in $[1..p]$. If S_j is not a prefix maximum in any prefix, we define $p(j) := 0$.*

The following algorithm PrefMax computes all $p(j)$ by scanning the sequence $S_1, \ldots, S_k$. The algorithm temporarily marks and unmarks some sets (actually, their indices), and the sets being still marked in the end are the prefix maxima, see Lemma 2.

PrefMax
for $j = 1, \ldots, k$ do
begin
    mark j;
    for all marked $i < j$ do
        case '$S_i \subset S_j$': unmark i
        case '$S_j \subseteq S_i$': $p(i) := j$; unmark j;
        case 'S_i, S_j incomparable': $p(i) := j$;
    if j marked then $p(j) := j$ else $p(j) := 0$;
end

Lemma 2 *After the jth iteration of the outer loop in PrefMax, the marked indices are exactly those of the prefix maxima in $[1..j]$.*

This is easy to see by induction on j, from the definition of prefix maxima and the rules of the procedure. Now the correctness of PrefMax follows from Lemma 2, as the $p(j)$ are updated accordingly.

Next we extend the use of PrefMax a bit. Suppose that, for some fixed element s, all occurrences of s are removed from the S_j. (We may also remove several fixed elements at once.) While existing inclusion relations $S_i \subseteq S_j$ are obviously preserved by that, it may happen that some incomparable S_i, S_j become comparable by removals. We wish to recompute the $p(j)$ without running algorithm PrefMax from scratch. In fact, we do apply PrefMax, but in general with fewer comparisons: Note that, since existing inclusions are preserved by element removals, the $p(j)$ can never grow. But some p_j may get smaller, because more sets S_i may now satisfy $S_j \subseteq S_i$. In particular, we know immediately that every $p(j) = 0$ remains zero. Thus we can skip indices j with $p(j) = 0$ in the outer loop of PrefMax. Moreover, since such j got unmarked, no indices i with $p(i) = 0$ need to be considered in the inner loop either. That is, we can ignore such indices altogether and run PrefMax on those indices, in the given order, where the $p(j)$ are still positive.

3 The Maximal Bicliques Generated Through a Hull Operator

Let $H = (X, Y, E)$ be our given bipartite graph. As usual, let $N(v)$ denote the set of all neighbors of a vertex v.

Definition 3 *For $A \subseteq X$ we define $\sigma(A)$ to be the set of all vertices in Y which are adjacent to all vertices of A. Equivalently, $\sigma(A) = \bigcap_{v \in A} N(v)$. We define $\sigma(B)$ similarly for $B \subseteq Y$, and we let $\phi(A) := \sigma(\sigma(A))$.*

The following lemmas are straightforward to prove.

Lemma 4 *ϕ is a hull operator, that is, ϕ is extensive, increasing, and idempotent. In detail: $A \subseteq \phi(A)$, $A \subseteq A'$ implies $\phi(A) \subseteq \phi(A')$, and $\phi(\phi(A)) = \phi(A)$.*

Lemma 5 *Consider any maximal biclique (A, B) of H, where $A \subseteq X$ and $B \subseteq Y$. Then we have $\sigma(A) = B$ and $\phi(A) = A$. Conversely, for any subset $A \subseteq X$ with $\phi(A) = A$, we have that (A, B) is a maximal biclique, where $B := \sigma(A)$.*

From now on we suppose without loss of generality that no vertex of X is adjacent to all vertices of Y. Namely, if the set $U \subseteq X$ of such vertices is non-empty, then $\sigma(U) = Y$ and $\phi(U) = U$, hence (U, Y) is a maximal biclique, furthermore, $U \subseteq A$ holds for every maximal biclique (A, B). Thus we can instantly remove U without changing the problem. By convention, $\sigma(\emptyset) = Y$ and $(\emptyset, Y)$ is now a maximal biclique.

Consider any maximal biclique (A, B). Let g_1 be the leftmost vertex of A. For $i > 1$ we define further elements $g_i \in A$ inductively: Since ϕ is a hull operator and $\phi(A) = A$, we have $\phi(\{g_1, \ldots, g_i\}) \subseteq A$. Let g_{i+1} be the leftmost vertex of $A \setminus \phi(\{g_1, \ldots, g_i\})$. We stop as soon as $\phi(\{g_1, \ldots, g_j\}) = A$. At this moment we call $G = \{g_1, \ldots, g_j\}$ the *generator* set of A, and the g_i are called generator vertices. All other vertices in the aforementioned hulls $\phi(\{g_1, \ldots, g_i\})$ are called *forced* vertices, as they must belong to every maximal biclique that contains the generator vertices. Note that $(\emptyset, Y)$ has the generator set $\emptyset$.

Based on the inductive definition of generators, we give a routine which is the building block of our enumeration algorithm.

Extension(G)
Given a set $G \subset X$, check all $g \in X$ to the right of G:
If no vertices of $\phi(G \cup \{g\}) \setminus \phi(G)$ are to the left of g, then return $G \cup \{g\}$.

We are ready to describe our main algorithm MaxBiclique which enumerates the generator sets of all maximal bicliques; recall that they also yield the maximal bicliques. The algorithm maintains a family F of sets $G \subseteq X$ which are processed in arbitrary order. Initially F contains only $G = \emptyset$. (That is, $F = \{\emptyset\}$, which should not be confused with $F = \emptyset$.)

MaxBicliques
$F := \{\emptyset\}$;
repeat
    Pick any set $G \in F$.
    Output G and remove G from F.
    Apply Extension(G).
    Put all returned sets $G \cup \{g\}$ in F.
until $F = \emptyset$

Theorem 6 *MaxBicliques outputs exactly the generator sets G of all maximal bicliques, and each of them exactly once.*

Proof. We show that every set G returned by the algorithm is in fact the generator set of some maximal biclique. This is proved by induction on $|G|$. For $G = \emptyset$ this is vacuously true. For the induction step, consider any $G = G' \cup \{g\}$, where G' is the set that generated G, that is, G is among the results of Extension(G'). By the induction hypothesis, G' is already the

generator set of some maximal biclique. Define $A := \phi(G)$. Due to the Extension procedure, g is to the right of G', and no vertex of $\phi(G' \cup \{g\}) \setminus \phi(G') = A \setminus \phi(G')$ is to the left of g. In other words, g is the leftmost vertex of $A \setminus \phi(G')$. Next define $B := \sigma(A)$. Since ϕ is idempotent by Lemma 4, we also have $\phi(A) = \phi(\phi(G)) = \phi(G) = A$, hence (A, B) is a maximal biclique by Lemma 5, and $G' \cup \{g\} = G$ is its generator set, according to the inductive definition of generator sets.

Conversely, the algorithm does not miss any generator set G of a maximal biclique. This can be seen by induction on $|G|$, too. $G = \emptyset$ is considered due to initialization of F. For the induction step, consider any $G \neq \emptyset$ and let g denote the rightmost vertex of G, and $G = G' \cup \{g\}$. Applying the inductive definition of generator sets again, we see that G' is the generator set of some maximal biclique. Moreover, g satisfies the precondition of Extension(G'). By the induction hypothesis, G' is produced by the algorithm, hence G is produced as well. The last assertion is trivial; since the sets G grow strictly from left to right, no set can be produced in two different ways. $\diamond$

It remains to discuss the implementation of Extension(G). Remember that $A = \phi(G)$ and $B = \sigma(A)$. Thus, for any vertex $x \in X$ we have $B \cap N(x) \subset B$ if and only if $x \notin A$. Another elementary observation is that g forces x, that is, $x \in \phi(G \cup \{g\}) \setminus \phi(G)$, if and only if $B \cap N(g) \subseteq B \cap N(x)$. Thus, a candidate g for the next generator vertex satisfies the precondition of Extension(G) if and only if $B \cap N(g)$ is a prefix maximum in the sequence of sets $B \cap N(x)$, where the x are all vertices in $X \setminus A$ to the left of g and including $x = g$. Now it is obvious that we can check the precondition of Extension(G) collectively for all g to the right of G by running PrefMax, where the sets S_j are the $B \cap N(x)$, for all vertices $x \in X \setminus A$ given in the order induced by our fixed order of X.

4 Fine-Tuning and Analysis

We presume a uniform-cost model where dictionary operations need $O(1)$ time. (We deal with subsets of vertices that can be stored as a table of size n, or as a hashtable, along with information about the cardinality.) In particular, a test whether $S \subseteq T$ can be done in $O(|S|)$ time, and the intersection $S \cap T$ of two sets can be computed in $O(\min\{|S|, |T|\})$ time. (In a logarithmic cost model our analysis works similarly, just with logarithmic factors attached.) Since we aim at output-sensitive time bounds, we assign every computation to some of the produced maximal bicliques, more precisely, to its generator set G, and we let G "pay" for the computation.

In order to make our enumeration algorithm efficient, we take some more measures. First of all, we now use a particular order of X, by decreasing degrees: We index the vertices $x_i \in X$ such that $m_1 \geq \ldots \geq m_k$, where m_j is the degree of x_j. (The algorithm developed in the previous section works for any order of X.) For $z, x \in X$ we say that "we compare z and x", as a shorthand for the comparison of $B \cap N(z)$ and $B \cap N(x)$. Let L and R be the set of vertices in $X \setminus A$ to the left and to the right, respectively, of the right end of G. Recall that we only have to figure out which vertices $g \in R$ yield new generator sets $G \cup \{g\}$, and for these g, the $B \cap N(g)$ are prefix maxima.

Thus, instead of running PrefMax on the entire sequence $X \setminus A$, we first compare every $z \in L$ with every $x \in R$, and let G pay for these comparisons. Then we remove from R every vertex x with $B \cap N(x) \subseteq B \cap N(z)$ for some $z \in L$, because such $B \cap N(x)$ cannot be prefix maxima. Finally we apply PrefMax to the remaining (ordered) set R only. Due to the way

PrefMax works, we compare $x \in R$ with $z \in R$ to the left of x, only if $B \cap N(z)$ is currently a prefix maximum, and hence $G \cup \{z\}$ is already a new generator set. This allows us to let the new $G \cup \{z\}$ pay for the comparison.

Next we analyze the invoices which any generator set G has received. G pays for all comparisons between the sets from its L and R during Extension(G). Since we have sorted X by decreasing degrees, and always the smaller set determines the complexity of a set comparison, the total payment for this part is bounded by $O(t \sum_{j=t+1}^{k} m_j)$, where index t marks the right end of L. As t depends on G, we maximize this product over all indices t to obtain a general upper bound. Before that, G has also paid for the comparisons of its rightmost member z with vertices x further to the right, during the Extension routine that produced G (see above). By the same argument, these payments are bounded by $O(m)$. This finally shows:

Theorem 7 *The β maximal bicliques of a bipartite graph, with m edges and degree sequence $m_1 \geq \ldots \geq m_k$ on one side, can be enumerated in $O((m + \max_t t \sum_{j=t+1}^{k} m_j)\beta)$ time.*

The time bound might look somewhat incomprehensible, however it becomes simple for some prominent degree sequences. For notational convenience we omit ceiling brackets at fractional numbers that actually should be integers, since the O-notation suppresses lower-order terms anyhow. Also note that we can (with a bit of care) replace sums with integrals in an O-expression.

If the degree distribution on X is $m_j = m_1/j$ as in Zipf's law, the product in our time bound becomes $t \int_t^k m_1 x^{-1} dx = m_1 t(\ln k - \ln t)$. In order to maximize this term, we calculate t for which the derivative is zero. This yields $\ln k - \ln t - 1 = 0$, hence $t = k/e$. This results in the time bound $O((m + m_1 k)\beta)$.

But now consider a degree sequence that decreases a little faster: let $m_j = m_1/j^s$ for some constant s, $1 < s < 2$. Then the same product evaluates to $t \int_t^k m_1 x^{-s} dx = \frac{m_1}{s-1} t(t^{1-s} - k^{1-s})$. Again we set the derivative to zero and obtain $t^{1-s} - k^{1-s} + t(1-s)t^{-s} = (2-s)t^{1-s} - k^{1-s} = 0$, hence $t = (2-s)^{s-1}k = \Theta(k)$, resulting in the time bound $O((m + m_1 k^{2-s})\beta)$. We notice that it also holds for $s = 1$. Finally observe that $m = O(m_1 \log k)$ if $s = 1$, and $m = O(m_1)$ if $s > 1$. For $s \geq 2$ we still have $t \int_t^k m_1 x^{-s} dx = \frac{m_1}{s-1} t(t^{1-s} - k^{1-s}) = \frac{m_1}{s-1}(t^{2-s} - tk^{1-s})$, which is now monotone decreasing in t, hence maximized if $t = 1$, such that it becomes $O(m_1) = O(m)$. Altogether this shows:

Corollary 8 *Let $s \geq 1$ be any constant. Then the β maximal bicliques of a bipartite graph, with m edges and degree sequence $m_1 \geq \ldots \geq m_k$ on one side, where $m_j = \Theta(m_1/j^s)$, can be enumerated in $O(m_1 k^{2-s}\beta)$ time if $s < 2$, and in $O(m\beta)$ time if $s \geq 2$.*

In particular, since $m_1 \leq \Delta$, we get the $O(\Delta k^{2-s}\beta)$ bound announced in the Introduction. Trivially, the space is also bounded by the given time bounds. It might be interesting to explore whether the space for intermediate computations can be reduced. (However we have to output and store the bicliques anyway.) It appears to be practically more relevant to further trim the calculations as proposed below.

After running Extension(G), in every branch where a vertex g is added to G, the set B is diminished to $B \cap N(g)$, that is, some vertices are removed from B. Thus we can use the remark after Lemma 2: By memoizing the $p(j)$ values and handing them over to Extension($G \cup \{g\}$), we can save unnecessary computations there. We have not even taken advantage of this fact in our analysis, but clearly it is beneficial not to repeat computations

whose results are already there. Trivially, we can also rightaway skip those vertices of X that are no longer neighbors of B, and work on the remaining induced subgraph of H. Removing these unnecessary vertices and edges costs $O(m)$ time, and we have anyhow charged every G by at least that amount in our analysis. This observation is particularly helpful in later phases with small B, provided that the vertices in Y have small degrees. The latter is the case in our intended applications where the vertices in Y represent short texts.

Acknowledgment

This work has been supported by the Swedish Foundation for Strategic Research (SSF) through grant IIS11-0089 for a data mining project entitled "Data-driven secure business intelligence".

References

[1] G. Alexe, S. Alexe, Y. Crama, S. Foldes, P.L. Hammer, B. Simeone. Consensus algorithms for the generation of all maximal bicliques. Discr. Appl. Math. 145 (2004) 11–21

[2] V.M.F. Dias, C.M.H. de Figueiredo, J.L. Szwarcfiter. Generating bicliques of a graph in lexicographic order. Theor. Comp. Sci. 337 (2005) 240–248

[3] V.M.F. Dias, C.M.H. de Figueiredo, J.L. Szwarcfiter. On the generation of bicliques of a graph. Discr. Appl. Math. 155 (2007) 1826–1832

[4] A. Gely, L. Nourine, B. Sadi. Enumeration aspects of maximal cliques and bicliques. Discr. Appl. Math. 157 (2009) 1447–1459

[5] E. Kayaaslan. On enumerating all maximal bicliques of bipartite graphs. In: U. Faigle, R. Schrader, D. Herrmann (eds.) CTW 2010, 105–108

[6] J. Li, G. Liu, H. Li, L. Wong. Maximal biclique subgraphs and closed pattern pairs of the adjacency matrix: A one-to-one correspondence and mining algorithms. IEEE Trans. Knowl. Data Eng. 19 (2007) 1625–1637

[7] W. Li. Random texts exhibit Zipf's-law-like word frequency distribution. IEEE Trans. Info. Th. 38 (1992) 1842–1845

[8] K. Makino, T. Uno. New algorithms for enumerating all maximal cliques. In: T. Hagerup, J. Katajainen (eds.) SWAT 2004, LNCS 3111, 260–272

[9] N. Nagarajan, C. Kingsford. Uncovering genomic reassortments among influenza strains by enumerating maximal bicliques. In: X. Chen, X. Hu, S. Kim (eds.) BIBM 2008, IEEE Comp. Soc., 223–230