Information Processing Letters 114 (2014) 317–321

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Enumerating maximal bicliques in bipartite graphs with favorable degree sequences

Peter Damaschke [*]

Department of Computer Science and Engineering, Chalmers University, 41296 Göteborg, Sweden

ARTICLE INFO

Article history:
Received 18 October 2013
Received in revised form 2 January 2014
Accepted 1 February 2014
Available online 4 February 2014
Communicated by R. Uehara

Keywords:
Analysis of algorithms
Biclique enumeration
Degree sequence
Power law
Hull operator

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.

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.

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,

[*] Tel.: +46 31 772 5405; fax: +46 31 772 3663.
E-mail address: ptr@chalmers.se.

http://dx.doi.org/10.1016/j.ipl.2014.02.001
0020-0190/© 2014 Elsevier B.V. All rights reserved.

the jth highest degree in X is about a $1/j^s$ fraction of the highest degree. Here s is any constant with $1 \leqslant s < 2$. 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 $j \leqslant p$ a *prefix maximum* in $[1..p]$ if no S_i, $i \leqslant p$, 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.