On Maximal Chain Subgraphs and Covers of Bipartite Graphs

Tiziana Calamoneri[1], Mattia Gastaldello[1,2], Arnaud Mary[2], Marie-France Sagot[2], and Blerina Sinaimeri[2]

[1] Sapienza University of Rome
via Salaria 113, 00198 Roma, Italy.
[2] INRIA and Université de Lyon
Université Lyon 1, LBBE, CNRS UMR558, France.

Abstract. In this paper, we address three related problems. One is the enumeration of all the maximal *not necessarily induced* chain subgraphs of a bipartite graph, for which we provide a polynomial delay algorithm. We give bounds on the number of maximal chain subgraphs for a bipartite graph and use them to establish the input-sensitive complexity of the enumeration problem. The second problem we treat is the one of finding the minimum number of chain subgraphs needed to cover all the edges a bipartite graph. For this we provide an exact exponential algorithm with a non trivial complexity. Finally, we approach the problem of enumerating all minimal chain subgraph covers of a bipartite graph and show that it can be solved in quasi-polynomial time.
Keywords: Chain Subgraph Cover Problem, Enumeration Algorithms, Exact exponential algorithms.

1 Introduction

Enumerating (listing) the subgraphs of a given graph plays an important role in analysing its structural properties. It thus is a central issue in many areas, notably in data mining and computational biology.

In this paper, we address the problem of enumerating without repetitions all maximal *not necessarily induced* chain subgraphs of a bipartite graph. These are graphs that do not contain a $2K_2$ as induced subgraph. From now on, we will refer to them as *chain subgraphs* for short when there is no ambiguity.

Bipartite graphs arise naturally in many applications, such as biology as will be mentioned later in the introduction, since they enable to model the relations between two different classes of objects. The problem of enumerating in bipartite graphs all subgraphs with certain properties has thus already been considered in the literature. These concern for instance maximal bicliques for which polynomial delay enumeration algorithms in bipartite [6, 11] as well as in general graphs [5, 11] were provided. In the case of maximal *induced* chain subgraphs, their enumeration can be done in output polynomial time as it can be reduced to the enumeration of a particular case of the minimal hitting set problem [7] (where the sets in the family are of cardinality 4). However, the existence of a polynomial delay algorithm for this problem remains open. To the best

of our knowledge, nothing is known so far about the problem of enumerating maximal *not necessarily induced* chain subgraphs in bipartite graphs.

In this paper, we propose a polynomial delay algorithm to enumerate all maximal chain subgraphs of a bipartite graph. We also provide an analysis of the time complexity of this algorithm in terms of input size. In order to do this, we prove some upper bounds on the maximum number of maximal chain subgraphs of a bipartite graph G with n nodes and m edges. This is also of intrinsic interest as combinatorial bounds on the maximum number of specific subgraphs in a graph are difficult to obtain and have received a lot of attention (see for *e.g.* [8, 12]).

We then address a second related problem called the *minimum chain subgraph cover* problem. This asks to determine, for a given graph G, the minimum number of chain subgraphs that cover all the edges of G. This has already been investigated in the literature as it is related to other well-known problems such as maximum induced matching (see *e.g.* [3, 4]). For bipartite graphs, the problem was shown to be NP-hard [14].

Calling m the number of edges in the graph, we provide an exact exponential algorithm which runs in time $O^*((2+\varepsilon)^m)$, for every $\varepsilon > 0$ by combining our results on the enumeration of maximal chain subgraphs with the inclusion-exclusion technique [1] (by O^* we denote standard big O notation but omitting polynomial factors). Notice that, since a chain subgraph cover is a family of subsets of edges, the existence of an algorithm whose complexity is close to 2^m is not obvious. Indeed, the basic search space would have size 2^{2^m}, which corresponds to all families of subsets of edges of a graph on m edges.

Finally, we approach the problem of enumerating all minimal covers by chain subgraphs. To this purpose, we provide a quasi-polynomial time algorithm to enumerate all *minimal* covers by maximal chain subgraphs of a bipartite graph. To do so, we prove that this can be polynomially reduced to the enumeration of the minimal set covers of a hypergraph.

Besides their theoretical interest, the problems of finding one minimum chain subgraph cover and of enumerating all such covers have also a direct application in biology. Nor *et al.* [13] showed that a minimum chain subgraph cover of such a bipartite graph provides a good model for identifying the minimum genetic architecture enabling to explain one type of manipulation, called *cytoplasmic incompatibility*, by bacteria of a genus called *Wolbachia* of their insect hosts. This phenomenon, results in the death of embryos produced in crosses between males carrying the infection and uninfected females. The observed cytoplasmic compatibility relationships, can be then represented by a bipartite graph with males and females in different classes. Moreover, as different minimum covers may correspond to solutions that differ in terms of their biological interpretation, the capacity to enumerate all such minimum chain covers becomes crucial.

The remainder of the paper is organised as follows. In Section 2, we give some definitions and preliminary results that will be used throughout the paper. Section 3 then provides a polynomial delay algorithm to enumerate all maximal chain subgraphs in a bipartite graph G with n nodes and m edges, and Section 4 presents an upper bound on their maximum number. We use the latter to further establish the input-sensitive complexity of the enumeration algorithm. In Section 5, we detail the exact algorithm for finding a minimum chain cover in bipartite graphs, and in Section 6 we exploit the

connection of this problem with the minimal set cover of a hypergraph to show that it is possible to enumerate in quasi-polynomial time all minimal covers by maximal chain subgraphs of a bipartite graph. Finally, we conclude with some open problems.

2 Preliminaries

Throughout the paper, we assume that the reader is familiar with the standard graph terminology, as contained for instance in [2]. We consider finite undirected graphs without loops or multiple edges. For each of the graph problems in this paper, we let n denote the number of nodes and m the number of edges of the input graph.

Given a bipartite graph $G = (U \cup W, E)$ and a node $u \in U$, we denote by $N_G(u)$ the set of nodes adjacent to u in G and by $E_G(u)$ the *set of edges incident to u in G*. Moreover, given $U' \subseteq U$ and $W' \subseteq W$, we denote by $G[U', W']$ the *subgraph of G induced by* $U' \cup W'$. A node $u \in U$ such that $N_G(u) = W$ is called a *universal node*.

A bipartite graph is a *chain graph* if it does not contain a $2K_2$ as an induced subgraph. Equivalently, a bipartite graph is a chain graph if and only if, for each two nodes v_1 and v_2 both in U (resp. in W), it holds that either $N_G(v_1) \subseteq N_G(v_2)$ or $N_G(v_2) \subseteq N_G(v_1)$. Given a chain subgraph $C = (X \cup Y, F)$ of G, we say that a permutation π of the nodes of X is a *neighbourhood ordering* of C if $N_C(u_{\pi(1)}) \subseteq N_C(u_{\pi(2)}) \subseteq \ldots \subseteq N_C(u_{\pi(|U|)})$. A permutation is just an ordered arrangement of a set of nodes. Observe that if $X \subset U$, the sets $N_C(u_{\pi(1)}), \ldots, N_C(u_{\pi(l)})$ for some integer $l \leq |U|$, may be empty. By the *largest neighbourhood of C*, we mean the neighbourhood of a node x in X for which the set $N_C(x) \subseteq Y$ has maximum cardinality. A set $Y' \subseteq Y$ is a *maximal neighborhood* of G, if there exists $x \in X$ such that $N_G(x) = Y'$ and there does not exist a node $x' \in X$ such that $N_G(x) \subset N_G(x')$. Two nodes x, x' such that $N_C(x) = N_C(x')$ are called *twins*.

In this paper, we always consider *non induced* chain subgraphs of a graph G. Hence, we identify a chain subgraph C of G by its set of edges $E(C) \subseteq E(G)$ and in that case its set of nodes will be constituted by all the nodes of G incident to at least one edge in C (sometimes abusing notation, we more simply write $C \subseteq G$ or $e \in C$). A *maximal chain subgraph* C of a given bipartite graph G is a connected chain subgraph such that no superset of $E(C)$ is a chain subgraph. We denote by $\mathcal{C}(G)$ the set of all maximal chain subgraphs in G.

A set of chain subgraphs $C_1, \ldots, C_k$ is a *cover* for G if $\cup_{1 \leq i \leq k} E(C_i) = E(G)$. Observe that, given any cover of G by chain subgraphs $C = \{C_1, \ldots C_k\}$, there exists another cover of same size $C' = \{C'_1, \ldots C'_k\}$ whose chain subgraphs are all maximal; more precisely, for each $i = 1, \ldots, k$, C'_i is a maximal chain subgraph of G and C'_i admits C_i as subgraph. In order to avoid redundancies, from now on, although not explicitly highlighted, we will restrict our attention to the covers by maximal chain subgraphs.

We denote by $\mathcal{S}(G)$ the set of all minimal chain covers of a bipartite graph G.

An enumeration algorithm is said to be *output polynomial* or *total polynomial* if the total running time is polynomial in the size of the input and the output. It is said to be *polynomial delay* if the time between the output of any one solution and the next one is bounded by a polynomial function of the input size [10].

3 Enumerating All Maximal Chain Subgraphs

In this section, we provide a polynomial delay algorithm for enumerating all the maximal chain subgraphs of a given bipartite graph. We start by proving the following result.

Proposition 1. *Let $C = (X \cup Y, F)$ be a chain subgraph of $G = (U \cup W, E)$, with $X \subseteq U$, $Y \subseteq W$ and $F \subseteq E$, and let $x \in X$ be a node with largest neighbourhood in C. Then C is a maximal chain subgraph of G if and only if:*

(i) *$N_C(x) = N_G(x)$ is a maximal neighbourhood of G, i.e. there does not exist a node $x' \in X$ such that $N_G(x) \subset N_G(x')$.*
(ii) *$C \setminus E_G(x)$ is a maximal chain subgraph of $G[U \setminus \{x\}, N_G(x)]$.*

Proof. ($\Rightarrow$) Let $C = (X \cup Y, F)$ be a maximal chain subgraph of $G = (U \cup W, E)$. To prove that *(i)* holds, suppose by contradiction that $N_C(x)$ is not a maximal neighbourhood of G, *i.e.* there exists $x' \in U$ with $N_C(x) \subset N_G(x')$ (possibly $x' = x$). Since $N_C(x)$ is the largest neighbourhood of C, for all $z \in X$, we have $N_C(z) \subseteq N_C(x) \subset N_G(x')$, so we can then add to C all the edges incident to x' and still obtain a chain subgraph thereby contradicting the maximality of C. To prove that *(ii)* holds, first observe that $N_G(x) = Y$ (otherwise we would violate (i) with $x' = x$). By contradiction, assume that $C \setminus E_G(x)$ is not maximal in $G[U \setminus \{x\}, N_G(x)]$. Then, there exists a chain subgraph C' such that $C \setminus E_G(x) \subset C' \subseteq G[U \setminus \{x\}, N_G(x)]$. By adding to each one of the previous graphs the edges in $E_G(x)$, we have that the strict inclusion is preserved because the added edges were not present in any one of the three graphs. Since C' with the addition of $E_G(x)$ is still a chain subgraph with $N_G(x)$ as its largest neighbourhood, we reach a contradiction with the hypothesis that C is maximal in G.

($\Leftarrow$) We show that if both *(i)* and *(ii)* hold, then the chain subgraph C of G is maximal. Suppose by contradiction that C is not maximal in G, and let C' be a chain subgraph of G such that $C \subset C'$. Let x be the node with the largest neighbourhood in C. It follows that $N_C(x) \subseteq N_{C'}(x)$. As *(i)* holds, we have that $N_G(x) = N_C(x) \subseteq N_{C'}(x) \subseteq N_G(x)$ from which we derive that $N_{C'}(x) = N_G(x)$, and that $C' \subseteq G[U, N_G(x)]$ since $N_{C'}(x)$ is a maximal neighbourhood of G, hence the largest neighbourhood of C' (and C by the hypothesis). This implies also that C and C' differ in some node different from x, *i.e.* $C \setminus E_G(x) \subset C' \setminus E_G(x) \subseteq G[U \setminus \{x\}, N_G(x)]$. Notice that $C' \setminus E_G(x)$ is still a chain subgraph because we simply removed node x and all its incident edges. We then get a contradiction with *(ii)*. □

Proposition 1 leads us to the design of Algorithm 1 which efficiently enumerates all maximal chain subgraphs of G. It exploits the fact that, in each maximal chain subgraph, a node u whose neighbourhood is largest is also maximal in G (part *(i)* of Proposition 1) and this holds recursively in the chain subgraph obtained by removing node u and restricting the graph to $N_C(u)$ (part *(ii)* of Proposition 1). To compute the maximal neighbourhood nodes, the algorithm uses a function, `computeCandidates`, that, given sets U and W, returns for each maximal neighbourhood $Y \subset W$, a unique node u, called *candidate*, for which $N_G(u) = Y$. This means that in case of twin nodes, the function `computeCandidates` extracts only one representative node according to some fixed order on the nodes (*e.g.* the node with the smallest label according to the lexicographical order). If the graph has no edges, the function returns the empty set.

Proposition 2 (Correctness). *Algorithm 1 correctly enumerates all the maximal chain subgraphs of the input graph G without repetitions.*

Proof. Let $G = (U \cup W, E)$ be a bipartite graph. We prove the correctness of Algorithm 1 by induction on $|U|$, *i.e.* we show that all the solutions are output, without repetitions.

When $|U| = 1$, let u be the only node in U. We have that $N_G(u)$ is the only neighbourhood in W, and line 3 returns $\{u\}$ as unique candidate. In line 9, the algorithm reduces the graph of interest. In line 10, the whole $E_G(u)$ is added to the current chain subgraph C. Then the function is recursively recalled, with $U' = \emptyset$ so the condition at line 4 is true and C is printed; it is in fact the only chain subgraph of G, and it is trivially maximal. No repetitions hold. Correctness then follows when $|U| = 1$.

Assume now that $|U| = k$ with $k > 1$. As inductive hypothesis, let the algorithm work correctly when $|U| = k - 1$.

For each candidate u, the algorithm recursively recalls the same function on a reduced graph and, by the inductive hypothesis, outputs all chain subgraphs of this reduced subgraph without repetitions. By Proposition 1, if we add to each one of these chain subgraphs the node u and all the edges incident to u in $G[U, W]$, we get a different maximal chain subgraph of G since each maximal chain subgraph has one and only one maximal neighborhood and the function `computeCandidates` returns only one representative node. Recall that in the case of twin nodes the algorithm will always consider the nodes in a precise order and so no repetition occurs. Moreover, iterating this process for all candidates guarantees that all maximal chain subgraphs are enumerated and no one is missed. □

Algorithm 1: Enumerate All Maximal Chain Subgraphs

Input: A bipartite graph $G = (U \cup W, E)$
Output: All maximal chain subgraphs of G

```
1  C ⟵ ∅ ;     /* C is the set of edges of the current chain subgraph */
2  enumerateMaximalChain(U, W, C)
3      Candidates ⟵ computeCandidates(U,W)
4      if Candidates == ∅ then
5          print(C);
6          return;
7      end
8      for u ∈ Candidates do
9          U' ⟵ U \ {u}; W' ⟵ W ∩ N_G(u);                    /* reduced graph */
10         F(u) ⟵ {edges of E_G(u) incident to some node in W'}
11         enumerateMaximalChain(U', W', C ∪ F(u));
12     end
```

Let $G = (U \cup W, E)$ be a bipartite graph, with $n = |U| + |W|$ and $m = |E|$. Before proving the time complexity of Algorithm 1, we observe that the running time of the function `ComputeCandidates` is $O(nm)$. Indeed, if we assume that the adjacency lists of the graph are ordered, for each node $u_i \in U$, it requires only time proportional to $i \cdot deg(u_i) \leq n \cdot deg(u_i)$ to check whether the neighbourhood of u_i either is included, or includes the neighbourhood of u_j, for each $j < i$.

Proposition 3 (Time Complexity and Polynomial Delay). *Let $G = (U \cup W, E)$ be a bipartite graph. The total running time of Algorithm 1 is $O(|\mathcal{C}(G)|n^2m)$. Moreover, the solutions are enumerated in polynomial time delay $O(n^2m)$.*

Proof. Represent the computation of Algorithm 1 as a tree of the recursion calls of `enumerateMaximalChain`, each node of which stores the current graph on which the recursion is called at line 11. Of course, the root stores G and on each leaf the condition $Candidates = \emptyset$ is true and a new solution is output. Observe that each leaf contains a feasible solution, and that no repetitions occur in view of Proposition 2, so the number of leaves is exactly $|\mathcal{C}(G)|$.

Since at each call the size of U is reduced by one, the tree height is necessarily bounded by $|U| = O(n)$; moreover, on each tree node, $O(nm)$ time is spent for running function `ComputeCandidates`.

It follows that, since the algorithm explores the tree in DFS fashion starting from the root, between two solutions the running time is at most $O(n^2m)$ and the total running time is $O(|\mathcal{C}(G)|n^2m)$. □

4 Upper bounds on the number of Maximal Chain Subgraphs

In this section, we give two upper bounds on the maximum number of maximal chain subgraphs of a bipartite graph G with n nodes and m edges. The first bound is given in terms of n while the second depends on m. These bounds are of independent interest, however we will use them in two directions. First, they will allow us to determine the (input-sensitive) complexity of Algorithm 1. Indeed, in Proposition 3, we proved that the total running time of Algorithm 1 is of the form $O(D(n) \times |\mathcal{C}(G)|)$, where $D(n)$ is the delay of the algorithm and $|\mathcal{C}(G)|$ is the number of maximal chain subgraphs of G. Thus, a bound on $|\mathcal{C}(G)|$ leads to a bound on the running time of Algorithm 1 depending on the size of the input. Second, the bound on $|\mathcal{C}(G)|$ in terms of edges allows us to compute the time complexity of an exact exponential algorithm for the minimum chain subgraph cover problem in Section 5.

4.1 Bound in terms of nodes

The following lemma claims that a given permutation is the neighbourhood ordering of at most one maximal chain subgraph.

Lemma 1. *Let C_1 and C_2 be two maximal chain subgraphs of $G = (U \cup W, E)$ and let π_1 (resp. π_2) be a neighbourhood ordering of C_1 (resp. C_2). Then, $\pi_1 = \pi_2 \implies C_1 = C_2$.*

Proof. The proof proceeds by induction on the number of nodes of U.

If $|U| = 1$ then G has only one maximal chain subgraph and the result trivially holds.

Assume now that $|U| > 1$. By Proposition 1, we have that $N_{C_1}(u_{\pi(|U|)}) = N_G(u_{\pi(|U|)}) = N_{C_2}(u_{\pi(|U|)})$. Using again Proposition 1, we obtain that $C'_1 := C_1[U \setminus \{u_{\pi(|U|)}\}, N_G(u_{\pi(|U|)})]$ and $C'_2 := C_2[U \setminus \{u_{\pi(|U|)}\}, N_G(u_{\pi(|U|)})]$ are maximal chain subgraphs of the graph defined as $G[U \setminus \{u_{\pi(|U|)}\}, N_G(u_{\pi(|U|)})]$. Applying the inductive hypothesis with the permutations restricted to the $|U| - 1$ elements, we have that $C'_1 = C'_2$. Finally, since $N_{C_1}(u_{\pi(|U|)}) = N_{C_2}(u_{\pi(|U|)})$, we conclude that $C_1 = C_2$. □

As a corollary, the maximum number of chain subgraphs of a graph $G = (U \cup W, E)$ is bounded by $|U|!$. Since the same reasoning can be applied on W, we have that $|\mathcal{C}(G)| \leq |W|!$ and hence:

$$|\mathcal{C}(G)| \leq \min(|U|, |W|)! \leq \frac{n}{2}!$$

This bound is tight as shown by the following family of graphs that reaches it.

Consider the *antimatching graph with n nodes* $A_n = (U \cup W, E)$ defined as the complement of an $n/2$ edge perfect matching, *i.e.*:

$$U := \{u_1, \ldots, u_{n/2}\}, \quad W := \{w_1, \ldots, w_{n/2}\},$$
$$E := \{(u_i, w_j) \in U \times W : i \neq j\}$$

It is not difficult to convince oneself that the maximal chain subgraphs of A_n are exactly $(n/2)!$ and that a different permutation corresponds to each of them. In particular, for each permutation π of the nodes of U, the corresponding maximal chain subgraph C_π of A_n can be defined by means of the set of neighbourhoods as follows:

$$N_{C_\pi}(u_i) := \{w_k \text{ s.t. } \pi^{-1}(k) < \pi^{-1}(i)\}.$$

The so-defined graph C_π is a chain subgraph since all the neighbourhoods form a chain of inclusions. Moreover, it is maximal since if we added to the neighbourhood of u_i any one of the missing edges (u_i, w_j) with $\pi^{-1}(j) \geq \pi^{-1}(i)$, we would introduce a $2K_2$ with the existing edge (u_j, w_i) as (u_j, w_j) and (u_i, w_i) are not in E.

4.2 Bound in terms of edges

Let $T(m)$ be the maximum number of maximal chain subgraphs over all bipartite graphs with m edges. We prove that $T(m) \leq 2^{\sqrt{m}\log(m)}$.

Lemma 2. *Let $G = (U \cup W, E)$ be a bipartite graph. Then $|\mathcal{C}(G)| \leq |U| \cdot T(m - |W|)$.*

Proof. In view of how the algorithm works and of Proposition 1, at the beginning, there at most $|U|$ candidates. For each candidate x, we can build as many chain subgraphs as there are in $G[U \setminus \{x\}, N_G(x)]$. We claim that this latter graph has at most $m - |W|$ edges. Indeed, in order to construct $G[U \setminus \{x\}, N_G(x)]$, we remove from G exactly $|E_G(x)|$ edges when deleting x from U, and $|W| - |N_G(x)|$ nodes (each one connected to at least a different edge as G is connected) when reducing W to $N_G(x)$. Observing that $|E_G(x)| = |N_G(x)|$, in total we remove $|W|$ edges. The proof follows from the fact that the number of chain subgraphs of $G[U \setminus \{x\}, N_G(x)]$ is bounded by $T(m - |W|)$. □

Theorem 1. *Let $G = (U \cup W, E)$ be a bipartite graph with n nodes and m edges; then $|\mathcal{C}(G)| \leq 2^{\sqrt{m}\log m}$, i.e. $T(m) \leq 2^{\sqrt{m}\log m}$.*

Proof. Assume w.l.o.g that $|U| \leq |W|$. The proof is by induction on m. Note that for $m = 1$ the theorem holds trivially.

Applying the inductive hypothesis and Lemma 2, we have:

$$|\mathcal{C}(G)| \leq |U|T(m - |W|) \leq \frac{n}{2} 2^{\left(\sqrt{m - \frac{1}{2}n}\log\left(m - \frac{1}{2}n\right)\right)}.$$

Since the function $\begin{array}{rcl} F : [\sqrt{m}, m-1] & \to & \mathbb{R} \\ x & \mapsto & x2^{\sqrt{m-x}\log(m-x)} \end{array}$ is decreasing (see Lemma 5 in Appendix), the maximum of $\frac{n}{2}2^{\sqrt{m-\frac{n}{2}}\log(m-\frac{n}{2})}$ is reached when $n/2$ is minimum. Note that trivially for a bipartite graph we have $n/2 > \sqrt{m}$. Hence,

$$|C(G)| \leq \sqrt{m}\,2^{\sqrt{m-\sqrt{m}}\log(m-\sqrt{m})}$$

Let $A := \sqrt{m} - \sqrt{m-\sqrt{m}}$ and $B := \frac{m-\sqrt{m}}{m}$. We then have:

$$\begin{aligned} |C(G)| &\leq \sqrt{m}\,2^{(\sqrt{m}-A)\log(mB)} \\ &= 2^{\sqrt{m}\log m} \times \sqrt{m}\,2^{\log B(\sqrt{m}-A)-A\log(m)} \end{aligned}$$

Let us show that $Z := \sqrt{m}\,2^{\log B(\sqrt{m}-A)-A\log m} \leq 1$ by showing that $\log Z \leq 0$:

$$\begin{aligned} \log Z &= \log\sqrt{m} + \log B(\sqrt{m}-A) - A\log(m) \\ &= \log\sqrt{m}(1-2A) + \log B(\sqrt{m}-A) \\ &\leq 0 \end{aligned}$$

considering that $B < 1$ and $1/2 < A \leq 1$ since:

$$A = \frac{1}{1+\sqrt{B}} = \frac{1}{1+\sqrt{1-\frac{1}{\sqrt{m}}}}$$

□

Corollary 1. *The (input-sensitive) complexity of Algorithm 1 is bounded by* $O^*(2^{\sqrt{m}log(m)})$.

5 Minimum Chain Subgraph Cover

In this section, we show how to find in polynomial space a minimum chain subgraph cover in time $O^*((2+\epsilon)^m)$, for every $\varepsilon > 0$. Since a chain subgraph cover is a family of subsets of edges, the existence of an algorithm whose complexity is close to 2^m is not obvious. Indeed the basic search space has size 2^{2^m}, as it corresponds to a family of subsets of edges. To obtain this result, we exploit Algorithm 1, the bound obtained in Theorem 1 and the inclusion/exclusion method [1, 8] that has already been successfully applied to exact exponential algorithms for many partitioning and covering problems.

We first express the problem as an inclusion-exclusion formula over the subsets of edges of G.

Proposition 4. *[1] Let $c_k(G)$ be the number of chain subgraph covers of size k of a graph G. We have that:*

$$c_k(G) = \sum_{A \subseteq E} (-1)^{|A|} a(A)^k$$

where $a(A)$ denotes the number of maximal chain subgraphs not intersecting A.

Exploring this result brings to the exact algorithm as described in the proof of the next theorem.

Theorem 2. *Given a bipartite graph G with m edges, for all $k \in \mathbb{N}^*$ and for all $\varepsilon > 0$, $c_k(G)$ can be computed in time $O^*((2+\epsilon)^m)$.*

Proof. Let $G = (U \cup W, E)$ be a bipartite graph, $k \in \mathbb{N}^*$ and $\varepsilon > 0$. Using the formula of Proposition 4, c_k can be computed in time $\sum_{i=0}^{m} \binom{m}{i} C(i)$, where $C(i)$ is the time complexity needed to compute $a(A)$, $|A| = i$.

Notice that to compute $a(A)$ for a given $A \subseteq E$, one can naively compute all maximal chain subgraphs of $G' = (U \cup W, E \setminus A)$ and, for each of them, check whether it is maximal in G. Using this fact, and Corollary 1, $C(i)$ can be determined in time $O(n^2 m 2^{\sqrt{m-i}\log(m-i)})$.

Thus we have that $c_k(G)$ can be computed in time $\sum_{i=0}^{m} \binom{m}{i} n^2 m 2^{\sqrt{m-i}\log(m-i)}$. Observe now that since $2^{\sqrt{m-i}\log(m-i)} = o((1+\varepsilon)^m)$, there exists a constant n_ε such that for all $m > n_\varepsilon$, $2^{\sqrt{m-i}\log(m-i)} < (1+\varepsilon)^m$.

Recalling that G is connected and thus $m \geq n$, we then have:

$$\begin{aligned}
\sum_{i=0}^{m} \binom{m}{i} n^2 m 2^{\sqrt{m-i}\log(m-i)} &= n^2 m \left(\sum_{i=0}^{m-n_\varepsilon-1} \binom{m}{i} 2^{\sqrt{m-i}\log(m-i)} + \sum_{i=m-n_\varepsilon}^{m} \binom{m}{i} 2^{\sqrt{m-i}\log(m-i)} \right) \\
&\leq n^2 m \left(\sum_{i=0}^{m-n_\varepsilon-1} \binom{m}{i} (1+\varepsilon)^{m-i} + n_\varepsilon m^{n_\varepsilon} 2^{\sqrt{n_\varepsilon}\log(n_\varepsilon)} \right) \\
&\leq n^2 m \left(\sum_{i=0}^{m} \binom{m}{i} (1+\varepsilon)^{m-i} + n_\varepsilon m^{n_\varepsilon} 2^{\sqrt{n_\varepsilon}\log(n_\varepsilon)} \right) \\
&\leq n^2 m (2+\varepsilon)^m + n^2 n_\varepsilon m^{1+n_\varepsilon} 2^{\sqrt{n_\varepsilon}\log(n_\varepsilon)} \\
&= O^*((2+\varepsilon)^m).
\end{aligned}$$

We conclude, by observing that the size of a minimum chain cover is given by the smallest value of k for which $c_k(G) \neq 0$. □

6 Enumeration of Minimal Chain Subgraph Covers

In this section, we prove that the enumeration of all minimal chain subgraph covers can be polynomially reduced to the enumeration of the minimal set covers of a hypergraph. This reduction implies that there is a quasi-polynomial time algorithm to enumerate all minimal chain subgraph covers. Indeed, the result in [9] implies that all the minimal set covers of a hypergraph can be enumerated in time $N^{\log N}$ where N is the sum of the input size (*i.e.* $n + m$) and of the output size (*i.e.* the number of minimal set covers).

Let $G = (U \cup W, E)$ be a bipartite graph, $C = C(G)$ the set of all its maximal chain subgraphs, and $\mathcal{S} = \mathcal{S}(G)$ the set of its minimal chain subgraph covers. Notice that the minimal chain subgraph covers of G are the minimal set covers of the hypergraph $\mathcal{H} := (V, \mathcal{E})$ where $V = E$ and $\mathcal{E} = C$. Unfortunately, the size of $\mathcal{H}$ might be exponential in the size of G plus the size of $\mathcal{S}$. Indeed not every maximal chain subgraph in C will necessarily be part of some minimal chain subgraph cover. In order to obtain a quasi-polynomial time algorithm to enumerate all minimal chain subgraph covers, we need to

enumerate only those maximal chain subgraphs that belong to a minimal chain subgraph cover.

Given an edge $e \in E$, let $\mathcal{C}_e$ be the set of all maximal chain subgraphs of G containing e and $\mathcal{M}_e$ the set of all edges $e' \in E$ inducing a $2K_2$ in G together with e.

We call an edge $e \in E$ *non-essential* if there exists another edge $e' \in E$ such that $\mathcal{C}_{e'} \subset \mathcal{C}_e$. An edge which is not non-essential is said to be *essential*. Note that for every non-essential edge e, there exists an essential edge e_1 such that $\mathcal{C}_{e_1} \subset \mathcal{C}_e$. Indeed, by applying iteratively the definition of a non-essential edge, we obtain a list of inclusions $\mathcal{C}_e \supset \mathcal{C}_{e_1} \supset \mathcal{C}_{e_2} \ldots$, where no $\mathcal{C}_{e_i}$ is repeated as the inclusions are strict. The last element of the list will correspond to an essential edge.

The following lemma claims that if a maximal chain subgraph C contains at least one essential edge, then it belongs to at least one minimal chain subgraph cover.

Lemma 3. *Let C be a maximal chain subgraph of a bipartite graph $G = (U \cup W, E)$. Then C belongs to a minimal chain subgraph cover of G if and only if C contains an essential edge.*

Proof. ($\Rightarrow$) Let C belong to a minimal chain subgraph cover M and assume that C contains no essential edge. Given $e \in C$, e therefore being non-essential, there exists an essential edge e' such that $\mathcal{C}_{e'} \subset \mathcal{C}_e$. Moreover, $e' \notin C$. As M is a cover, there exists $C' \in M$ such that $e' \in C'$. Thus, $C' \neq C$, $C' \in \mathcal{C}_{e'} \subset \mathcal{C}_e$, hence $e \in C'$. Since for every edge $e \in C$, there exists $C' \in M$ containing it, we have that $M \setminus \{C\}$ is a cover, contradicting the minimality of M.

($\Leftarrow$) Assume C contains an essential edge e. Let $\mathcal{C}' = \{D \in \mathcal{C}(G) : e \notin D\}$. Note that $\mathcal{C}' = \mathcal{C} \setminus \mathcal{C}_e$. We show that $\mathcal{C}' \cup \{C\}$ is a cover. Suppose on the contrary that there exists $e' \in E \setminus E(C)$ and e' is not covered by $\mathcal{C}'$ and thus $\mathcal{C}_{e'} \cap \mathcal{C}' = \emptyset$. This implies that $\mathcal{C}_{e'} \subseteq \mathcal{C} \setminus \mathcal{C}' = \mathcal{C}_e$ and as e is essential, we obtain $\mathcal{C}_{e'} = \mathcal{C}_e$ from which we deduce that $e' \in C$. Thus, $M = \mathcal{C}' \cup \{C\}$ is a cover and clearly it contains a minimal one. Finally, we conclude by observing that, since by construction C is the only chain subgraph of M that contains e, it belongs to any minimal cover contained in M. □

It follows that the set of maximal chain subgraphs that can contribute to a minimal chain cover is $\tilde{\mathcal{C}} = \cup \mathcal{C}_e$ where the index e of the union operation runs over all the essential edges of G. In the following, we show how to detect essential edges. This problem then consists in detecting all the couples e_1, e_2 such that $\mathcal{C}_{e_1} \subseteq \mathcal{C}_{e_2}$ before enumerating all useful maximal chain subgraphs. The next lemma is proved in Appendix.

Lemma 4. *Let C be a maximal chain subgraph of a bipartite graph $G = (U \cup W, E)$ and let $e \in E$ be such that for all $e' \in E(C)$, it holds that $e \notin \mathcal{M}_{e'}$. Then $e \in C$.*

Using this lemma we can now prove the following result.

Theorem 3. *Given a bipartite graph $G = (U \cup W, E)$, for any two edges $e, e' \in E$, $\mathcal{C}_e \subseteq \mathcal{C}_{e'}$ if and only if $\mathcal{M}_e \supseteq \mathcal{M}_{e'}$.*

Proof. ($\Rightarrow$) Given two edges $e, e' \in E$, suppose that $\mathcal{C}_e \subseteq \mathcal{C}_{e'}$, and assume on the contrary that there exists $f \in \mathcal{M}_{e'}$ and $f \notin \mathcal{M}_e$. Then there exists a maximal chain

C' containing e and f (as they do not form a $2K_2$ in G) but not e' ($f \in \mathcal{M}_{e'}$). Hence, $C' \in \mathcal{C}_e$ but $C' \notin \mathcal{C}_{e'}$, contradicting the assumption that $\mathcal{C}_e \subseteq \mathcal{C}_{e'}$.

($\Leftarrow$) Suppose now $\mathcal{M}_e \supseteq \mathcal{M}_{e'}$. Let $C \in \mathcal{C}_e$. By definition, none of the edges of $\mathcal{M}_e$ appears in C. Hence, e' does not form a $2K_2$ with any edge in C in the graph G (as $\mathcal{M}_e \supseteq \mathcal{M}_{e'}$). By Lemma 4 $e' \in C$. Thus, $\mathcal{C}_e \subseteq \mathcal{C}_{e'}$. □

Notice that, given an edge $e = (u, w) \in E$, $u \in U$ and $w \in W$, it is easy to determine the set $\mathcal{M}_e$. We just need to start from E and delete all edges that are incident either to u or to w, as well as all edges at distance 2 from e (that is all edges $e' = (u', w')$ such that either u' is adjacent to w or w' is adjacent to u). Checking whether $\mathcal{M}_e \supseteq \mathcal{M}_{e'}$ is also easy: it suffices to sort the edges in each set in lexicographic order, and then the inclusion of each pair can be checked in linear time in their size, that is in $O(m)$. It is thus possible to enumerate in polynomial delay only those maximal chain subgraphs that contain at least one essential edge by modifying Algorithm 1. Due to space limits, we do not detail the algorithm here and refer instead the reader to Algorithm 2 in the Appendix. Finally, we are now able to state the main result of this section.

Theorem 4. *Given a bipartite graph $G = (U \cup W, E)$, one can enumerate all its minimal chain subgraph covers, i.e. all the elements in $\mathcal{S}$, in time $O(|\mathcal{S}|^{\log(|\mathcal{S}|)+2})$.*

Proof. We first construct the hypergraph $\mathcal{H} = (V, \mathcal{E})$ where $V := E'$ is the set of essential edges of G and $\mathcal{E} := \mathcal{C}_{ess}$ is the set of maximal chain subgraphs of G that contain at least one essential edge. This takes time $O(n^2 m|\mathcal{C}_{ess}|)$. Applying then the algorithm given in [9], one can enumerate all minimal set covers of $\mathcal{H}$ (*i.e.* all minimal chain subgraph covers) in time $O((|\mathcal{H}| + |\mathcal{S}|)^{\log(|\mathcal{H}|+|\mathcal{S}|)}) = O((|\mathcal{C}_{ess}| + |\mathcal{S}|)^{\log(|\mathcal{C}_{ess}|+|\mathcal{S}|)})$. The total running time is thus $O(n^2 m|\mathcal{C}_{ess}| + (|\mathcal{C}_{ess}| + |\mathcal{S}|)^{\log(|\mathcal{C}_{ess}|+|\mathcal{S}|)})$. Notice now that since by Lemma 3, every maximal chain subgraph in $\mathcal{C}_{ess}$ belongs to at least one minimal chain subgraph cover, we have that $|\mathcal{C}_{ess}| \leq |\mathcal{S}|$. Finally, we obtain that the total running time is $O(n^2 m|\mathcal{S}| + (|\mathcal{S}| + |\mathcal{S}|)^{\log(|\mathcal{S}|+|\mathcal{S}|)}) = O(|\mathcal{S}|^{\log(|\mathcal{S}|)+2})$.

7 Conclusion

In this paper, we studied different problems related to maximal chain subgraphs and chain subgraph covers in bipartite graphs. This work raises many questions. First, it remains an open problem whether it is possible to enumerate the minimal chain covers of a graph in polynomial delay. Indeed, our problem is more constrained than an arbitrary instance of the set cover of a hypergraph. A future goal is to better exploit the connections between these two problems. Second, it would be interesting to determine the exact value of $T(m)$. We conjecture that a tighter bound may be $2^{\frac{1+\sqrt{1+4m}}{2}}$. Finally, it is worth exploring the different nature of the problems considered here in the case where we deal with an hereditary property (induced chain subgraphs) instead of a non-hereditary one (non necessarily induced chain subgraphs). In particular, it remains unknown whether enumerating maximal induced subgraphs can be done in polynomial delay.

Acknowledgments

T. Calamoneri is supported in part by the Italian Ministry of Education, University, and Research (MIUR) under PRIN 2012C4E3KT national research project "AMANDA - Algorithmics for MAssive and Networked DAta" and in part by Sapienza University of Rome project "Graph Algorithms for Phylogenetics: A Promising Approach". M. Gastaldello is supported by the Università Italo-Francese project "Algorithms and Models for the solution of difficult problems in biology". A. Mary is supported by the ANR project GraphEN "Enumération dans les graphes et les hypergraphes: algorithmes et complexité", ANR-15-CE40-0009.

References

1. Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, July 2009.
2. Béla Bollobás. *Modern graph theory*, volume 184 of *Graduate Texts in Mathematics*. Springer-Verlag, Berlin, 1998.
3. A. Brandstädt, E. M Eschen, and R. Sritharan. The induced matching and chain subgraph cover problems for convex bipartite graphs. *Theoretical computer science*, 381(1):260–265, 2007.
4. Yu Chang-Wu, Chen Gen-Huey, and Ma Tze-Heng. On the complexity of the k-chain subgraph cover problem. *Theoretical computer science*, 205(1):85–98, 1998.
5. Vânia M.F. Dias, Celina M.H. de Figueiredo, and Jayme L. Szwarcfiter. Generating bicliques of a graph in lexicographic order. *Theoretical Computer Science*, 337(1-3):240 – 248, 2005.
6. Vânia M.F. Dias, Celina M.H. de Figueiredo, and Jayme L. Szwarcfiter. On the generation of bicliques of a graph. *Discrete Applied Mathematics*, 155(14):1826 – 1832, 2007.
7. Thomas Eiter and Georg Gottlob. Identifying the minimal transversals of a hypergraph and related problems. *SIAM Journal on Computing*, 24(6):1278–1304, 1995.
8. Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 2010.
9. M. L Fredman and L. Khachiyan. On the complexity of dualization of monotone disjunctive normal forms. *Journal of Algorithms*, 21(3):618–628, 1996.
10. D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou. On generating all maximal independent sets. *Information Processing Letters*, 27(3):119–123, 1988.
11. Kazuhisa Makino and Takeaki Uno. *SWAT 2004, Lecture Notes in Computer Science*, chapter New Algorithms for Enumerating All Maximal Cliques, pages 260–272. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.
12. J. W. Moon and L. Moser. On cliques in graphs. *Israel Journal of Mathematics*, 3(1):23–28, 1965.
13. I. Nor, J. Engelstädter, O. Duron, M. Reuter, M-F. Sagot, and S. Charlat. On the genetic architecture of cytoplasmic incompatibility: inference from phenotypic data. *The American Naturalist*, 182(1):E15–E24, 2013.
14. Mihalis Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):351–358, 1982.

Appendix

Upper bounds on the number of Maximal Chain Subgraphs

We provide here the proof of the following result that was used in the proof of Theorem 1.

Lemma 5. *The function:*

$$\begin{array}{rccc} F : [\sqrt{m}, m-1] & \to & & \mathbb{R} \\ x & & \mapsto & x2^{\sqrt{m-x}\log(m-x)} \end{array}$$

is decreasing.

Proof. The derivative of $F(x)$ is given by:

$$\begin{aligned} & -x\left(\frac{\log(m-x)}{2\sqrt{m-x}} + \frac{1}{\sqrt{m-x}}\right)2^{(\sqrt{m-x}\log(m-x))} + 2^{(\sqrt{m-x}\log(m-x))} \\ = & -\frac{\left(x\log(m-x) + 2x - 2\sqrt{m-x}\right)2^{(\sqrt{m-x}\log(m-x))}}{2\sqrt{m-x}} \end{aligned}$$

Then the derivative is negative whenever $\left(x\log(m-x) + 2x - 2\sqrt{m-x}\right) \geq 0$.

Observe that $\log(m-x) \geq 0$ for $x \leq m-1$, while for $x \geq 0$ we have:

$$2x - 2\sqrt{m-x} \geq 0 \iff x \geq \frac{-1+\sqrt{1+4m}}{2} = -\frac{1}{2} + \sqrt{m+\frac{1}{4}}$$

and:

$$\sqrt{m} \geq -\frac{1}{2} + \sqrt{m+\frac{1}{4}}$$

□

Enumeration of Minimal Chain Subgraph Covers

We prove here Lemma 4. We start by showing the following fact.

Fact 1 *Let $C = (X \cup Y, F)$ be a maximal chain subgraph of a bipartite graph $G = (U \cup W, E)$, and let $z \in X$, $e = \{u, w\} \in E$ be such that for every $e' \in E_C(z)$, we have $e \notin \mathcal{M}_{e'}$. Then at least one of the following holds:*

(1) $w \in N_G(z)$.
(2) $u \in \cap_{y \in N_C(z)} N_G(y)$.

Proof. The proof follows straightforwardly by observing that for any $e' = \{z, y\} \in C$ then as $e \notin \mathcal{M}_{e'}$, either $\{z, w\} \in E(G)$ or $\{u, y\} \in E(G)$. □

Lemma 4. *Let C be a maximal chain subgraph of a bipartite graph $G = (U \cup W, E)$ and let $e \in E$ be such that for all $e' \in E(C)$, it holds that $e \notin \mathcal{M}_{e'}$. Then $e \in C$.*

Proof. Let $C = (X \cup Y, F)$ be a maximal chain subgraph of $G = (U \cup W, E)$ and w.l.o.g., let $N_C(u_1) \subseteq N_C(u_2) \subseteq \ldots \subseteq N_C(u_{|X|})$. Let $e = \{u, w\}$ in E be such that for all $e' \in E(C)$, it holds that $e \notin \mathcal{M}_{e'}$. Assume that $e \notin C$.

We will show that $w \in \cap_{x \in X} N_G(x)$ leads to a contradiction because we could add all the edges $E_G(w)$ to C and still obtain a chain subgraph (with $N_G(w)$ as the largest neighbourhood of C).

Observe that in the previous claim, we can re-write (2) in the form $N_C(z) \subseteq N_G(u)$.

Using the previous fact with $z = u_{|X|}$, we have that at least one from (1) and (2) must hold. Observe that (2) cannot hold as otherwise it implies that $u \in \cap_{y \in N_C(z)} N_G(y)$. We are then done by interchanging the roles of Y and X and observing by point (i) of Proposition 1 that $N_C(u_{|X|}) = N_G(u_{|X|}) = Y$. Thus, (1) must hold, *i.e.* $w \in N_G(u_{|X|})$.

If we now show that $w \in \cap_{k=j}^{|X|} N_G(u_k) \Rightarrow w \in N_G(u_{j-1})$, we are done since together with $w \in N_G(u_{|X|})$ this leads to:

$$w \in \bigcap_{k=1}^{|X|} N_G(u_k) = \bigcap_{x \in X} N_G(x)$$

To prove this, we apply again Fact 1 with $z = u_{j-1}$ and show that (2) cannot hold. Notice that from (2), we have $N_C(u_{j-1}) \subseteq N_G(u)$. Then using the maximality of C, we deduce that $u \in X$. Indeed, $N_C(u)$ has to contain at least $N_C(u_{j-1})$, and hence there exists $\tilde{k} \geq j - 1$ for which $u = u_{\tilde{k}}$. Then we could extend C to C' by adding e. Observe that C' has the following list of neighbourhoods:

$$\begin{aligned} N_{C'}(u_k) &:= N_C(u_k) && \text{for } k \neq \tilde{k} \\ N_{C'}(u_k) &:= N_C(u_k) \cup \{w\} && \text{for } k = \tilde{k} \end{aligned}$$

and is a chain subgraph since $N_C(u_{\tilde{k}}) \cup \{w\} \subseteq N_C(u_k)$ for all $k > \tilde{k} \geq j - 1$ by $w \in \cap_{k=j}^{|X|} N_G(u_k)$ and the maximality of C.

Description of Algorithm 2.

Algorithm 2: Enumerate All Maximal Chain Subgraphs with not an empty intersection with a given set of edges E'

Input: A bipartite graph $G = (U \cup W, E)$, a set of edges $E' := \{u_1w_1, ..., u_kw_k\}$
Output: All maximal chain subgraphs of G that intersect E'

```
1  C ⟵ ∅
2  enumerateMaximalChain(U, W, C)
3      Candidates ⟵ computeCandidates(U,W)
4      if Candidates == ∅ then
5          print(C);
6          return;
7      end
8      if {u_1, ..., u_k} ⊆ U then
9          Candidates ⟵ Candidates \ {u ∈ U : N_G(u) ∩ {w_1, ..., w_k} = ∅}
10     end
11     for u ∈ Candidates do
12         U' ⟵ U \ {u}; W' ⟵ W ∩ N_G(u);          /* reduced graph */
13         F(u) ⟵ { edges of E_G(u) incident to some node in W'}
14         enumerateMaximalChain(U', W', C ∪ F(u));
15     end
```