Discrete Mathematics and Theoretical Computer Science DMTCS vol. **15:1**, 2013, 55–74

Hereditary biclique-Helly graphs: recognition and maximal biclique enumeration

Martiniano Eguía and Francisco J. Soulignac[†]

Universidad de Buenos Aires, Facultad de Ciencias Exactas y Naturales, Departamento de Computación, Argentina

received 10th March 2011, revised 6th April 2012, accepted 11th January 2013.

A biclique is a set of vertices that induce a complete bipartite graph. A graph G is biclique-Helly when its family of maximal bicliques satisfies the Helly property. If every induced subgraph of G is also biclique-Helly, then G is hereditary biclique-Helly. A graph is C_4-dominated when every cycle of length 4 contains a vertex that is dominated by the vertex of the cycle that is not adjacent to it. In this paper we show that the class of hereditary biclique-Helly graphs is formed precisely by those C_4-dominated graphs that contain no triangles and no induced cycles of length either 5 or 6. Using this characterization, we develop an algorithm for recognizing hereditary biclique-Helly graphs in $O(n^2 + \alpha m)$ time and $O(n+m)$ space. (Here n, m, and $\alpha = O(m^{1/2})$ are the number of vertices and edges, and the arboricity of the graph, respectively.) As a subprocedure, we show how to recognize those C_4-dominated graphs that contain no triangles in $O(\alpha m)$ time and $O(n + m)$ space. Finally, we show how to enumerate all the maximal bicliques of a C_4-dominated graph with no triangles in $O(n^2 + \alpha m)$ time and $O(\alpha m)$ space.

Keywords: hereditary biclique-Helly graphs, maximal bicliques, triangle-free graphs, domination problems.

1 Introduction

A family of sets satisfies the Helly property, or simply is a Helly family, if for every subfamily $\mathcal{F}$ of pairwise intersecting sets there is an element common to all the sets in $\mathcal{F}$. The Helly property plays a central role in the study of clique graphs [3, 13, 18]. In the last years, several concepts that are widely studied for cliques have been translated in terms of bicliques [6, 7, 8, 10, 9, 11, 14, 19]. In the process, the classes of biclique-Helly and hereditary biclique-Helly graphs were first studied.

A graph is biclique-Helly when its family of maximal bicliques is Helly. A biclique-Helly graph can be obtained from any bipartite graph by inserting a vertex adjacent to all the vertices in one bipartition. Thus, induced subgraphs of a biclique-Helly graph need not be biclique-Helly. Hereditary biclique-Helly graphs are defined as those graphs whose induced subgraphs are all biclique-Helly. Hereditary biclique-Helly graphs were studied by Groshaus and Szwarcfiter, who characterized them by a family of six forbidden induced subgraphs with at most 8 vertices [10]. This characterization implies a polynomial recognition

[†]Partially supported by UBACyT Grant 20020100300048, PICT ANPCyT Grant 1970, and CONICET PIP Grant 11220100100310

algorithm for hereditary biclique-Helly graphs, though the most efficient algorithm to this date takes $O(n^3m^2)$ time (cf. [3]).

The problem of enumerating all the maximal bicliques of a graph is widely studied both for general graphs and for bipartite graphs (e.g. [2, 5, 12, 15]). Since a biclique-Helly graph can be obtained from any bipartite graph by inserting a vertex, any algorithm that enumerates the maximal bicliques of a biclique-Helly graph can also be used to enumerate all the maximal bicliques of a general bipartite graph. Since bipartite graphs can can have an exponential number of bicliques (see [17]), enumerating the maximal bicliques of a biclique-Helly graph can require an exponential amount of time.

In this paper we focus on two problems for hereditary biclique-Helly graphs: their recognition and the enumeration of maximal bicliques. For the recognition problem, we rephrase the characterization by Groshaus and Szwarcfiter in more algorithmic terms and, using this new characterization, we develop an $O(\alpha m + n^2)$ time and $O(n + m)$ space recognition algorithm, where $\alpha < \sqrt{m}$ is the arboricity of the graph. As one of the steps in our algorithm, we require the recognition of a larger class, formed by all the triangle-free graphs whose C_4's have at least one vertex dominated by other vertex of the cycle. We call this class, the class of C_4-dominated graphs with no triangles. We develop a recognition algorithm for this class that takes $O(\alpha m)$ time and $O(n+m)$ space. For the enumeration of maximal bicliques, we develop an $O(\alpha m + o)$ time and space algorithm that outputs all the bicliques of any triangle-free C_4-dominated graph, where $o < n^2$ is the size of the output. Furthermore, we prove that every maximal biclique of a graph in this class is formed by those vertices that either are adjacent or dominate v, for some vertex v. As a result, hereditary biclique-Helly graphs can have at most n maximal bicliques.

The article is organized as follows. In the next section we introduce the notation and terminology employed. In Section 3 we develop a simple $O(nm)$ time and $O(n^2)$ space algorithm for the recognition of biclique-Helly graphs. Following, in Sections 4 and 5, we present an improved implementation of this simple algorithm, so that it runs in $O(\alpha m + n^2)$ time and $O(n + m)$ space. The algorithm for enumerating the maximal bicliques of C_4-dominated graphs with no triangles is described in Section 6. Finally, in Section 7, we give some remarks and leave some open problems.

2 Preliminaries

In this paper we work with simple graphs. Let G be a graph with vertex set $V(G)$ and edge set $E(G)$, and call $n = |V(G)|$ and $m = |E(G)|$. Write vw to denote the edge of G formed by vertices $v, w \in V(G)$. For $v \in V(G)$, represent by $N_G(v)$ the set of vertices adjacent to v. The set $N_G(v)$ is called the *neighborhood* of v, and $d_G(v) = |N_G(v)|$ is the *degree of* v. Similarly, for $v, w \in V(G)$, define the *common neighborhood* of v and w as $N_G(vw) = N_G(v) \cap N_G(w)$. Say that v *dominates* w, or equivalently that w is *dominated* by v, when $N(w) \subseteq N(v)$. Note that v dominates w only if v is not adjacent to w. The set of vertices that dominate v is represented by $Dom_G(v)$. Say that v is *dom-comparable* to w when v either dominates or is dominated by w. When there is no ambiguity, we may omit the subscripts from N, Dom, and d.

Say that a total ordering $<$ of $V(G)$ is a *degree ordering* when $v < w$ only if $d(v) \leq d(w)$. A *degree ordered* graph is a pair $(G, <)$, where G is a graph and $<$ is a degree ordering of G. For the sake of simplicity, we say that G is a *degree ordered graph* to indicate that there is a degree ordering $<$ such that $(G, <)$ is a degree ordered graph. For a vertex v of a degree ordered graph G, define $\mathrm{MAX}_G(v) = \{w \in V(G) \mid w > v\}$ and $\mathrm{MIN}_G(v) = \{w \in V(G) \mid w < v\}$. For $W \subseteq V(G)$, we also use $\max_G W$ and $\min_G W$ to refer to the maximum and minimum elements of W, according to $<$. As before, we omit the

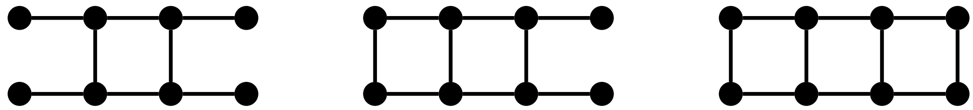

Fig. 1: The ladder graphs.

subscript from MAX, MIN, $\max$, and $\min$ when there is no ambiguity. For two vertices $v > w$, define the *least common neighborhood* as $L(v, w) = N(vw) \cap \mathrm{MIN}(v)$; note that, by definition, $L(w, v)$ is undefined for $v > w$.

For $W \subseteq V(G)$, denote by $G[W]$ the subgraph of G induced by W. An *independent set* is a set $W \subseteq V(G)$ formed by pairwise non-adjacent vertices. Graph G is *bipartite* when $V(G)$ can be partitioned into two independent sets W_1 and W_2. In this case, the unordered pair $\{W_1, W_2\}$ is called a *bipartition* of G. Furthermore, if vw is an edge of G for every $v \in W_1$ and $w \in W_2$, then G is a *complete* bipartite graph. A *biclique* H of G is a complete bipartite induced subgraph of G with at least one edge; we also use the term *biclique* to refer to both $V(H)$ and the unique bipartition of H.

Let $\mathcal{F}$ be a family of sets. Say that $\mathcal{F}$ is *pairwise intersecting* when $S \cap T \neq \emptyset$, for every $S, T \in \mathcal{F}$, while $\mathcal{F}$ is *globally intersecting* when $\bigcap \mathcal{F} \neq \emptyset$. Family $\mathcal{F}$ is *Helly* when all its pairwise intersecting subfamilies are globally intersecting. A graph G is *biclique-Helly* when its family of maximal bicliques is Helly, and it is *hereditary biclique-Helly* when all its induced subgraphs are biclique-Helly.

Denote by C_n the cycle graph with n vertices; C_3 is also called a *triangle*. For a graph H, say that G is H-free when no induced subgraph of G is isomorphic to H. Similarly, for a family $\mathcal{H}$ of graphs, say that G is $\mathcal{H}$-free when G is H-free for every $H \in \mathcal{H}$. The *arboricity* $\alpha(G)$ of G is the minimum number of edge-disjoint spanning forests into which G can be decomposed. Chiba and Nishizeki proved that $\alpha(G) \leq m^{1/2}$ [1].

In this paper we also work with simple digraphs. Let D be a graph with vertex set $V(D)$ and edge set $E(D)$. Write $v \to_D w$ to indicate that the ordered pair (v, w) is an edge of D. When (v, w) is not an edge of D, we write $v \not\to_D w$. For $v \in V(D)$, define $N^+_D(v) = \{w \in V(D) \mid v \to w\}$ and $N^-_D(v) = \{w \in V(D) \mid w \to v\}$. Sets $N^+_D(v)$ and $N^-_D(v)$ are respectively the *out-neighborhood* and *in-neighborhood* of v, while the members of $N^+_D(v)$ and $N^-_D(v)$ are the *out-neighbors* and *in-neighbors* of v, respectively. The *out-degree* and *in-degree* of v are the values $d^+_D(v) = |N^+_D(v)|$ and $d^-_D(v) = |N^-_D(v)|$, respectively. When there is no ambiguity, we may omit the subscripts from $\to$, $\not\to$, N^+, N^-, d^+, and d^-.

Groshaus and Szwarcfiter proved the following characterization of hereditary biclique-Helly graphs, by means of minimal forbidden induced subgraphs.

Theorem 2.1 ([10]) *A graph is hereditary biclique-Helly if and only if it is does not contain any triangles, C_5's, C_6's, nor ladders as induced subgraphs (see Figure 1).*

As a consequence of this theorem, the authors obtain an $O(n^3 m^2)$ time algorithm for the recognition of hereditary biclique-Helly graphs (cf. [3]).

3 Simple recognition of hereditary biclique-Helly graphs

In this section we rephrase Theorem 2.1 in such a way that an efficient recognition algorithm can be obtained. To describe our algorithm, we require the following definitions for a graph G. Say that a cycle of G is *dominated* if it contains a pair of dom-comparable vertices. When every C_4 of G is dominated, we say that G is C_4-*dominated*. Theorem 2.1 is rephrased as follows.

Theorem 3.1 *A graph is hereditary biclique-Helly if and only if it is C_4-dominated and $\{$triangle, C_5, $C_6\}$-free.*

Proof: By Theorem 2.1, graphs that contain triangles, C_5's, or C_6's as induced subgraphs are not hereditary biclique-Helly. Suppose now that G is a $\{$triangle, C_5, $C_6\}$-free graph that contains a cycle v_1, v_2, v_3, v_4 in which neither v_1 and v_3 nor v_2 and v_4 are dom-comparable. For the sake of notation, call $v_{i+4} = v_i$ for every $i \in \mathbb{Z}$. Then, for $i \in \mathbb{Z}$, there is a vertex $w_i \in N(v_i) \setminus N(v_{i+2})$. (Again, call $w_{j+4} = w_j$ for every $j \in \mathbb{Z}$.) Since G is triangle-free, w_i is adjacent to neither v_{i-1} nor v_{i+1}. Hence, $w_i \neq w_j$ for every $1 \leq i \leq j \neq 4$.

Clearly, v_1, v_2, v_3, v_4 is an induced cycle because G is triangle-free. Now, consider all the possible edges between the vertices in $\{w_1, w_2, w_3, w_4\}$. First observe that w_i is not adjacent to w_{i+2}; otherwise $w_i v_i v_{i+1} v_{i+2} w_{i+2}$ would induce a C_5 in G. Similarly, w_i is adjacent to neither w_{i-1} nor w_{i+1}; otherwise $w_{i-1} v_{i-1} v_{i-2} v_{i+1} w_{i+1} w_i$ would induce a C_6 in G. Therefore, the subgraph of G induced by $\{v_i w_i\}_{1 \leq i \leq 4}$ is isomorphic to a ladder and, by Theorem 2.1, G is not hereditary biclique-Helly .

For the converse, just observe that the cycle of a ladder formed by the vertices of degree at least 3 is not dominated. Then, the result follows from Theorem 2.1. □

Theorem 3.1 yields an algorithm for the recognition of hereditary biclique-Helly graphs, summarized in Algorithm 1. For Step 1, the algorithm in [1] is called so as to find a triangle in $O(m\alpha(G))$ time and $O(n+m)$ space, when one exists. In the rest of this section we discuss a simple $O(nm)$ time and $O(m^2)$ space implementation for Steps 2 to 4. In the following sections we improve the implementation of this algorithm so as to run $O(n^2 + \alpha(G)m)$ time and linear space.

Algorithm 1 Recognition of hereditary biclique-Helly graphs.

Input: a graph G.

Output: if G is not hereditary biclique-Helly, then either a triangle, a non-dominated C_4, an induced C_5, or an induced C_6; otherwise, a message.

1. If G contains a triangle T, then output T and halt.
2. If G contains a non-dominated C_4 called C, then output C and halt.
3. If G contains an induced C_5 called C, then output C and halt.
4. If G contains an induced C_6 called C, then output C and halt.
5. Output "G is HBH".

3.1 An $O(nm)$ time implementation of Step 2

The main tools for this step are the squares families. Fix a degree ordered graph G with no triangles for the rest of this section.

For a vertex v, the *squares family* of v is the family $\mathcal{S}(v)$ that contains one triple $S = (v, w, L(v, w))$ for each $w < v$ such that $L(v, w) \neq \emptyset$. Refer to v, w and $L(v, w)$ as the *high vertex*, *low vertex*, and *common* *neighborhood* of S, respectively. When $|L(v,w)| > 1$, the triple S encodes all the C_4's that contain v and w, where v is the maximum vertex of the C_4. Indeed, v, a, w, b is a C_4 of G and $v > \max\{a, b, w\}$ if and only if $a, b \in L(v, w)$. In this case, we say that S *represents* the cycle v, a, w, b, for every $a, b \in L(v, w)$. The *squares family* of G is $\mathcal{S}(G) = \bigcup_{v \in V(G)} \mathcal{S}(v)$. When G is understood, we will simply write $\mathcal{S}$ to mean $\mathcal{S}(G)$. Observe that every C_4 of G is represented by exactly one triple of $\mathcal{S}$. Thus, the squares family of G encodes all the C_4's of G in $O(m\alpha(G))$ space, though G could have $O(n^2)$ C_4's [1]. For the sake of notation, write $v(S)$, $w(S)$, and $L(S)$ to respectively mean the high vertex, the low vertex, and the common neighborhood of S, for every $S \in \mathcal{S}$. Also, we sometimes write (v, w) instead of $(v, w, L(v, w))$; we may write, for instance, that $(v, w) \in \mathcal{S}$ to indicate that $(v, w, L(v, w)) \in \mathcal{S}$.

Say that $S \in \mathcal{S}$ is *dominated* when all the C_4's represented by S are dominated. (If $|L(S)| = 1$, then S is vacuously dominated.) By definition, G is C_4-dominated if and only if S is dominated, for every $S \in \mathcal{S}$. Say also that S is *safe* if $v(S)$ dominates $w(S)$, and that it is *unsafe* otherwise. Observe that if S is safe, then it is also dominated. Otherwise, S is dominated if and only if a dominates b, for every $a, b \in L(S)$ such that $a > b$. These observations yield an algorithm to find a non-dominated C_4 of G, when one such C_4 exists, summarized as Algorithm 2.

Algorithm 2 Non-dominated C_4 of a triangle-free graph G.

Input: a degree ordered graph G with no triangles.
Output: if existing, a non-dominated C_4 of G; otherwise, a message.

1. Let $v_1 > \ldots > v_n$ be the vertices of G.
2. Compute the matrix $D \in \{0,1\}^{n \times n}$ such that $d_{i,j} = 1$ if and only if v_i dominates v_j, for $1 \leq i < j \leq n$. Write $D(v_i, v_j) = d_{i,j}$.
3. For $i = 1$ to n, do:
4. Compute $UNSAFE := \{(v_i, w, L(v_i, w)) \mid L(v_i, w) \neq \emptyset$ and $D(v_i, w) = 0\}$.
5. For each $S \in UNSAFE$ do:
6. Let $a_1 > \ldots > a_{|L(S)|}$ be the vertices of $L(S)$
7. If $D(a_j, a_{j+1}) = 0$ for $j \in \{1, \ldots, |L(S)| - 1\}$, then output $v(S), a_j, w(S), a_{j+1}$ and halt.
8. Output "G is C_4-dominated".

In Step 4, Algorithm 2 finds each unsafe $S \in \mathcal{S}(v_i)$. Following, the inner cycle checks that every such unsafe triple S is dominated. For this, it first computes a degree ordering $a_1 > \ldots > a_{|L(S)|}$ of $L(S)$, and then checks that a_j dominates a_{j+1}, for every $1 \leq j < |L(S)|$. If this check is fulfilled, then, since domination is a transitive relation, we obtain that a dominates b for every $a, b \in L(S)$ such that $a > b$. Thus, Algorithm 2 is correct. Algorithm 2 can be implemented so as to run in $O(nm)$ time and $O(n^2)$ space. A better implementation is given in Section 4.

3.2 An $O(nm)$ time implementation of Steps 3 and 4

For this paragraph, suppose that G is a degree ordered graph that is C_4-dominated and contains no triangles. The next lemma shows how to find an induced C_6 in G that contains any given vertex $v \in V(G)$, if existing. We omit its proof because a stronger version appears in Section 5.

Lemma 3.2 *Let G be a degree ordered graph that is C_4-dominated and contains no triangles, and $v_0 \in V(G)$. Then, there is an induced C_6 in G that contains v_0 if and only if there is a cycle $v_0, \ldots, v_5$ with the following properties:*

(i) $v_3 \notin N(v_0)$,

(ii) $v_1 = \min N(v_0v_2)$ *and* $v_5 = \min N(v_0v_4)$,

(iii) v_0 *is dominated by neither* v_2 *nor* v_4*, and*

(iv) v_1 *and* v_5 *are not dom-comparable.*

The above lemma yields Algorithm 3, that finds an induced C_6 in $O(nm)$ time and $O(n^2)$ space, if existing. A similar algorithm can be used to find if there is an induced C_5. Just observe that there is an induced C_5 containing a vertex v if and only if there are two adjacent vertices w and z at distance 2 from v. In Section 5 we devise better algorithms for finding if there is an induced C_5 or C_6.

Algorithm 3 Induced C_6 in a C_4-dominated graph G with no triangles.

Input: a degree ordered graph G that is C_4-dominated and contains no triangles.
Output: if existing, an induced C_6 of G; otherwise, a message.

1. Let $v_1 > \ldots > v_n$ be the vertices of G.

2. Compute the matrix $D \in \{0,1\}^{n \times n}$ such that $d_{i,j} = 1$ if and only if v_i dominates v_j, for $1 \leq i < j \leq n$. Write $D(v_i, v_j) = d_{i,j}$.

3. For each $w_0 \in V(G)$, do:

4. Compute $N_2 := \{w_2 \in V(G) \mid N(w_0w_2) \neq \emptyset$ and $D(w_2, w_0) = 0\}$

5. For each $w_2 \in N_2$, set $p(w_2) := \min N(w_0w_2)$.

6. If there is a vertex $w_3 \notin N(w_0)$ that is adjacent to $w_2, w_4 \in N_2$ and $D(p(w_2), p(w_4)) + D(p(w_4), p(w_2)) = 0$, then output $w_0, p(w_2), w_2, w_3, w_4, p(w_4)$ and halt.

7. Output "G contains no induced C_6's".

4 Faster recognition of C_4-dominated graphs with no triangles

In this section we develop an improved implementation of Algorithm 2 whose running time and space consumption are $O(m\alpha(G))$ and $O(n+m)$, respectively. The idea is the same, for each $S \in \mathcal{S}$ we first check if S is safe. If not, then we check the dominations between the vertices in $L(S)$. The major difference is that the domination matrix D is no longer employed; instead, while checking the safeness, we compute a digraph that encodes some dominations of interest of G. We begin with the description of this digraph. As before, we assume that G is a degree ordered graph with no triangles.

Fix $S \in \mathcal{S}$. Define the binary relation $\to_S$ on $L(S)$ as follows: for $a, b \in L(S)$, $a \to_S b$ if and only if $a < b$ and there is no $c \in L(S)$ such that $a < c < b$. In other words, $\to_S$ defines the subordering of $<$ induced by the members in $L(S)$. We require a generalization of $\mathcal{S}$ from vertices to sets. For $V \subseteq V(G)$, define $\mathcal{S}(V) = \bigcup_{v \in V} \mathcal{S}(v)$. Note that $\mathcal{S}(\emptyset) = \emptyset$ and $\mathcal{S}(V(G)) = \mathcal{S}(G)$. Now we are ready to define the domination digraph.

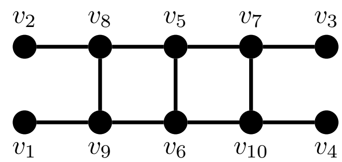

Fig. 2: An ordered graph G where $v_i < v_j$ if and only if $i < j$

For $V \subseteq V(G)$, the *unsafe domination digraph* of V is the digraph $U(V)$ with vertex set $V(G)$ such that, for $a, b \in V(G)$, $a \to_D b$ if and only if $a \to_S b$ for some unsafe $S \in \mathcal{S}(V)$. In other words, for each unsafe $S \in \mathcal{S}(V)$, there is a path that goes through the vertices of $L(S)$ in the order given by $<$. The *unsafe domination digraph* of G is the digraph $U(G) = U(V(G))$.

For an example, consider the ordered graph G depicted in Figure 2, where $v_i < v_j$ if and only if $i < j$, and let $S = (v_{10}, v_5)$ and $T = (v_9, v_5)$. By definition, $v_6 \to_S v_8$ and $v_6 \to_T v_7$ because $L(v_{10}, v_5) = \{v_7, v_6\}$ and $L(v_9, v_5) = \{v_8, v_6\}$. Hence, $v_6 \to v_7$ is an edge of $U(\{v_{10}\})$, while $v_6 \to v_8$ is an edge of $U(\{v_{10}, v_9\})$. These two edges are the only edges of $U(G) = U(V(G))$, because $L(v_i, v_j)$ is either empty or a singleton for any $i \leq 8$.

The following lemma shows that, when every $S \in \mathcal{S}(V)$ is dominated, $U(V)$ is actually a directed forest that encodes some dominations.

Lemma 4.1 *Let G be a degree ordered graph that is C_4-dominated and contains no triangles, and $V \subseteq V(G)$. If S dominated for every $S \in \mathcal{S}(V)$, then the following conditions hold for any $v \in V(G)$:*

(i) $d^+_{U(V)}(v) \leq 1$, *and*

(ii) v *is dominated in G by all its out-neighbors.*

Proof: (i). Suppose that $v \in V(G)$ has at least two out-neighbors $a > b$ in $U(V)$. Then, by definition, there are two unsafe triples in $\mathcal{S}(V)$, say S and T, such that $v \in L(S) \cap L(T)$, $v \to_S a$, and $v \to_T b$. Now, since $a > b$ and $v \to_S a$, it follows that $b < v(S)$ and $b \notin L(S)$. Consequently, b is not adjacent to either $v(S)$ or $w(S)$, thus v is not dominated by b in G. Also, since $v \to_T b$, it follows that $v < b$, i.e. $d_G(b) \geq d_G(v)$, hence b is neither dominated by v in G. But then, $T \in \mathcal{S}(V)$ is not dominated.

(ii). If $d^+_{U(V)}(v) = 0$, then (ii) is vacuously true. Otherwise, let w be an out-neighbor of v. Since $v \to_{U(V)} w$, we obtain that $v \to_S w$, for some unsafe $S \in \mathcal{S}(V)$. Since S is dominated and unsafe, it follows that w dominates v in G. $\square$

Recall that the idea of the new implementation is to build $U(G)$ so as to test the dominations inside $L(S)$, for every unsafe $S \in \mathcal{S}$. In turn, to build $U(G)$, we need to know the safeness status of some triples in $\mathcal{S}$, for which we require the dominations of these triples. It turns out that $U(G)$ can be iteratively computed while the partial results are used to check the safeness of those triples of interest. The next lemma shows how this construction is done.

Lemma 4.2 *Let G be a degree ordered graph that is C_4-dominated and contains no triangles, and $S \in \mathcal{S}$. Call $v = v(S)$, $w = w(S)$, and $L = L(S)$. If every triple of $\mathcal{S}(\mathrm{MAX}(v))$ is dominated, then the following are equivalent statements.*

(i) *S is safe.*

(ii) $|L| = |N(w) \cap \mathrm{MIN}(v)|$ *and if* $w \in L(T)$ *for some unsafe* $T \in \mathcal{S}(\mathrm{MAX}(v))$*, then there is a path from* w *to* v *in* $U(\mathrm{MAX}(v))$*.*

Proof: $(i) \Longrightarrow (ii)$. Suppose first that S is safe, i.e., v dominates w. By definition, $L = N(vw) \cap \mathrm{MIN}(v)$, thus $|L| = |N(w) \cap \mathrm{MIN}(v)|$. Now, suppose that there is some unsafe $T \in \mathcal{S}(\mathrm{MAX}(v))$ such that $w \in L(T)$. In this case, since v dominates w, it follows that v is adjacent to both $v(T)$ and $w(T)$. Hence $v \in L(T)$, because $v < v(T)$ as $T \in \mathcal{S}(\mathrm{MAX}(v))$. Therefore, there is a path from w to v in $U(\mathrm{MAX}(v))$ because $v > w$.

$(ii) \Longrightarrow (i)$. In this case, we argue by contradiction. Suppose that (ii) is true and yet S is unsafe, i.e., $N(w) \setminus N(v)$ contains some vertex z. Since $L = N(vw) \cap \mathrm{MIN}(v) \subseteq N(v)$ and $|L| = |N(w) \cap \mathrm{MIN}(v)|$, we obtain that $L = N(w) \cap \mathrm{MIN}(v)$, thus $z > v$. Fix $a \in L$. Since w is adjacent to both z and a, it follows that $w \in L(z, a)$, thus $(z, a) \in \mathcal{S}(z)$. Also, since $v \in N(a) \setminus N(z)$, it follows that (z, a) is unsafe, so there is a path from w to v in $U(\mathrm{MAX}(v))$. Therefore, by Lemma 4.1, v dominates w, a contradiction. $\square$

Lemma 4.2 yields Algorithm 4. Its input is the graph G, and its output is $U(G)$ if G is C_4-dominated, or a non-dominating cycle otherwise. Thus, Algorithm 4 is just a replacement of Algorithm 2. We discuss its correctness and complexity in the next paragraphs.

Algorithm 4 Non-dominated C_4 of a triangle-free graph G.

Input: a degree ordered graph G with no triangles.
Output: if G is C_4-dominated, then $U(G)$; otherwise, a non-dominated C_4 of G.

1. Let $v_1 > \ldots > v_n$ be the vertices of G.

2. Set $UNSAFE := \emptyset$, $U := (V(G), \emptyset)$ and $d^<(v) := |N(v)|$, for every $v \in V(G)$.

3. For $i := 1, \ldots, n$, do:

4. Set $REACH := \{w \in V(G) \mid d^<(w) > 0$ and there is a path from w to v_i in $U\}$.

5. For each $S \in \{(v_i, w, L) \in \mathcal{S} \mid$ either $|L| \neq d^<(w)$ or $(w \in UNSAFE$ and $w \notin REACH)\}$.

{*This loop checks that every unsafe S is dominated.*}

6. For each $a, b \in L(S)$ such that $a \to_S b$ do:

7. If $N_U^+(a) \not\subseteq \{b\}$, then output $v_i, a, w(S), b$ and halt.

8. If $N_U^+(a) = \emptyset$ and a is not dominated by b, then output $v_i, a, w(S), b$ and halt.

9. Add $a \to b$ to U.

10. Set $UNSAFE := UNSAFE \cup L(S)$.

11. Set $d^<(w) := d^<(w) - 1$, for every $w \in N(v_i)$.

12. Output U.

4.1 Correctness of Algorithm 4

The Loop 3–11 examines the vertices in the order defined by $<$, beginning from the greatest. Fix $i \in \{1, \ldots, n\}$, and call $MAX = \text{MAX}(v_i)$, and $MIN = \text{MIN}(v_i)$. Observe that, by Step 1, $MAX = \{v_1, \ldots, v_{i-1}\}$, and $MIN = \{v_{i+1}, \ldots, v_n\}$. Immediately before Loop 3–11 is executed for v_i, every $S \in \mathcal{S}(MAX)$ is dominated, and the state of the variables is as follows:

(i) $UNSAFE = \{v \in V(G) \mid v \in L(S)$ for some unsafe $S \in \mathcal{S}(MAX)\}$.

(ii) $U = U(MAX)$.

(iii) $d^<(v) = |N(v) \cap (MIN \cup \{v_i\})|$, for every $v \in V(G)$.

Step 4 finds all those $w \in V(G)$ such that there is a path from w to v_i in $U(MAX)$, and $d^<(w) > 0$. Observe that if $d^<(w) = 0$, then, by (iii), $L(v_i, w) = \emptyset$, thus $(v_i, w) \notin \mathcal{S}$. Then, by Lemma 4.2, Loop 5–10 iterates every unsafe triple $S \in \mathcal{S}$ whose high vertex is v_i. This loop is the responsible for testing whether S is dominated or not, and it has the following three alternatives.

Alternative 1: the algorithm halts at Step 7 while examining $a \to_S b$. For this to happen, $N_U^+(a)$ must contain a vertex different than b, say $\mu(a)$. Thus, by (ii), both b and $\mu(a)$ are out-neighbors of a in $U(MAX \cup \{v_i\})$. Therefore, G is not C_4-dominated by Lemma 4.1.

Alternative 2: the algorithm halts at Step 8 while examining $a \to_S b$. This alternative occurs when a is not dominated by b, thus S is not dominated. Therefore, G is not C_4-dominated.

Alternative 3: the algorithm does not halt inside Loop 5–10. In this last alternative, for every $a \to_S b$, either $N_U^+(a) = \{b\}$ or a is dominated by b. Whichever the case, by (ii), a is dominated by b and $a \to b$ is an edge of $U(MAX \cup \{v\})$. In particular, S is dominated.

Therefore, if Loop 5–10 halts if and only if $\mathcal{S}(MAX \cup \{v_i\})$ contains a non-dominated triple. Furthermore, if Loop 5–10 does not halt, then by Step 9, (ii) is satisfied before the execution of the outer loop for v_{i+1}. Finally, by Steps 10 and 11, (i) and (iii) also hold immediately before the execution of Loop 3–11 for v_{i+1}. (Observe that Step 7 is superfluous, and it can be removed from the algorithm without affecting its correctness. However, its inclusion drops the time complexity required by the algorithm. In some sense, it tells us that the domination of a by b was already tested.)

Algorithm 4 gives its output in one of three steps. Suppose that the algorithm halts at Step 7, as in Alternative 1. This happens because there is an unsafe triple T, already processed by the algorithm, such that $a \to_T \mu(a)$. We claim that $b < \mu(a)$; otherwise, as in Lemma 4.1, T would not be dominated, contradicting the fact that Algorithm 4 stops immediately after it process a non-dominated triple. Then, as in Lemma 4.1, a is not dominated by b. Similarly, if Algorithm 4 halts at Step 8, as in Alternative 2, then a is not dominated by b. Therefore, in both of these alternatives, $v_i, a, w(S), b$ is a non-dominated C_4. Finally, if Algorithm 4 does not halt inside Loop 5–10, as in Alternative 3, then $U = U(G)$, by (ii). Summing up, Algorithm 4 is correct.

4.2 Implementation and complexity of Algorithm 4

The implementation of Algorithm 4 is rather straightforward. Recall that, by Lemma 4.1, every vertex of U has at most one out-neighbor. We record such out-neighbors in a vector with n positions, where the i-th position is b if and only if $v_i \to_U b$. If $N^+(v_i) = \emptyset$, then the i-th position of the array is some undefined value $\perp$. Similarly, $d^<$ and $UNSAFE$ are also stored in vectors with n positions; the i-th position respectively indicates the values of $d^<(v_i)$ and $v_i \in UNSAFE$. We assume that a can be inserted into $N_U^-(b)$ in $O(1)$ time at Step 9, and that a can be also removed from $N_U^-(b)$ in $O(1)$ time when $d^<(a)$ drops from 1 to 0 at Step 11. This can be achieved by storing $N_U^-(b)$, for every $b \in V(G)$, and a pointer from a to its position in $N_U^-(b)$, for every $a \in N_U^-(b)$.

The time complexity of the algorithm with the above implementation is as follows. Fix $i \in \{1, \dots, n\}$. Before entering the Loop 3–11 for v_i, $\mathcal{S}(v_i)$ is obtained by executing one iteration of method $C4$ developed by Chiba and Nishizeki in [1]. Such iteration takes G and a vertex $v_i \in V(G)$ as input, and it outputs $\mathcal{S}(v_i)$. (In fact, method $C4$ discards those $S \in \mathcal{S}(v_i)$ for which $|L(S)| = 1$. However, the algorithm can be easily modified so as to output these triples as well.) Furthermore, for each $S \in \mathcal{S}(v_i)$, the obtained list $L(S)$ is ordered in such a way that $a \in L(S)$ appears before $b \in L(S)$ if and only if $a > b$, i.e., $b \to_S a$ if and only if a immediately precedes b. As proved in [1], this iteration of $C4$ costs $O(\sum_{S \in \mathcal{S}(v_i)} |L(S)|)$ time. For Step 4, just use a tree traversal algorithm, taking advantage that $N_U^-(b)$ is stored for every $b \in V(G)$. Such a traversal takes $O(1)$ time per vertex traversed. Observe that w is traversed at Step 4 if and only if $L(v_i, w) \neq \emptyset$, thus Step 4 takes $O(|\mathcal{S}(v_i)|)$ time. Finally, Loop 5–10 takes $O(|L(S)|)$ time to examine each $S \in \mathcal{S}(v_i)$. Thus, this loop also requires $O(\sum_{S \in \mathcal{S}(v_i)} |L(S)|)$. Finally, all the steps outside the Loop 3–11 cost $O(n)$ time. Therefore, as proven in [1], the time complexity of Algorithm 4 is

$$O\left(n + \sum_{v \in V(G)} \sum_{S \in \mathcal{S}(v)} |L(S)|\right) = O(n + m\alpha(G)).$$

On the other hand, only

$$O\left(n + \max_{v \in V(G)} \left\{ \sum_{S \in \mathcal{S}(v)} |L(S)| \right\}\right) = O(n + m)$$

bits of additional space are used by the algorithm, thus the space complexity is $O(n + m)$.

5 Faster recognition of hereditary biclique-Helly graphs

In this section we improve Algorithm 3, and the corresponding algorithm that finds an induced C_5, so that both run in $O(m\alpha(G) + n^2)$ time and $O(n + m)$ space. Again, the idea is to evaluate if a given vertex v belongs to an induced C_5 or an induced C_6, by looking at those vertices at distance at most 3 from v. In this section, however, we take advantage of the dominations implied by the squares family of G. The improved algorithm that finds an induced C_5 is rather similar to the improvement of Algorithm 3 for finding an induced C_6. So, we only describe in detail the improvement of Algorithm 3. The main tool in this section is a new forest that extends the unsafe domination digraph. For the rest of this section, suppose that G is a degree ordered graph that is C_4-dominated and has no triangles.

The *squares domination digraph* of G is the digraph $S(G)$ that is obtained from $U(G)$ by inserting an edge $w \to v$ for every safe $(v, w) \in \mathcal{S}$. Fix $v \in V(G)$. Define $\sigma(v) = \min N^+_{S(G)}(v)$, when $d^+_{S(G)}(v) > 0$. We write $\sigma(v) = \perp$ to indicate that $d^+_{S(G)}(v) = 0$. For the sake of simplicity, we assume $\perp > \max V(G)$.

The following lemma is analogous to Lemma 3.2. It shows how to find an induced C_6, when one such cycle exists. Observe the role that σ plays by marking that v_2 and v_4 are not dom-comparable (condition (ii)), and that v_1, v_3, and v_5 are neither dom-comparable (condition (iii)). Indeed, $\sigma(v_i)$ was chosen as the minimum that dominates v_i in either $U(G)$ or in a safe triple. So, if $v < \sigma(w)$, and v and w share a C_4, then v cannot dominate w.

Lemma 5.1 *Let G be a degree ordered graph that is C_4-dominated and contains no triangles. Then, G contains an induced C_6 if and only if it contains two paths v_0, v_1, v_2, v_3 and v_0, v_5, v_4, v_3 such that*

(i) $v_2 \neq v_4$, $v_3 \notin N(v_0)$ *and* $v_0 > \max\{v_1, v_2, v_4, v_5\}$,

(ii) $\sigma(v_2) > v_0$ *and* $\sigma(v_4) > v_0$, *and*

(iii) $\sigma(v_1) = \sigma(v_3) = \sigma(v_5)$.

Furthermore, if G contains the paths v_0, v_1, v_2, v_3 and v_0, v_5, v_4, v_3 satisfying (i)–(iii), then $v_0, \ldots, v_5$ induce a cycle in G.

Proof: Suppose first that G contains an induced C_6. For each vertex v, define $r(v)$ as the number of vertices reached from v in $S(G)$. For each $C \subseteq V(G)$, define $R(C) = \sum_{v \in C} r(v)$.

From among all the induced C_6's of G, chose the cycle $v_0, \ldots, v_5$ such that:

(1) there is no induced C_6 of G containing a vertex in $\mathrm{MAX}(v_0)$, and

(2) $R(\{v_0, \ldots, v_5\})$ is minimum among those cycles containing v_0.

We claim that paths v_0, v_1, v_2, v_3 and v_0, v_5, v_4, v_3 satisfy (i)–(iii). Clearly, $v_2 \neq v_4$, $v_3 \notin N(v_0)$, and $v_0 > \max\{v_1, \ldots, v_5\}$, thus (i) follows.

Suppose, contrary to statement (ii), that $\sigma(v_2) \leq v_0$. Notice that v_2 is not dominated by v_0, so $\sigma(v_2) < v_0$. Then, $\sigma(v_2) \neq \perp$, hence $\sigma(v_2)$ dominates v_2. By (2), $v_0, v_1, \sigma(v_2), v_3, v_4, v_5$ do not induce a cycle in G, because $r(v_2) > r(\sigma(v_2))$. Thus, as G is triangle-free, we obtain that $\sigma(v_2)$ is adjacent to v_1, v_3, and v_5, hence, $v_0, v_1, \sigma(v_2), v_5$ is a C_4. Now, since G is C_4-dominated and v_1 and v_5 are not dom-comparable, it follows that $\sigma(v_2)$ dominates v_0. However, this contradicts the fact that $\sigma(v_2) < v_0$. Similar arguments prove that $\sigma(v_4) > v_0$, so (ii) follows.

Finally, consider statement (iii). Clearly, (iii) is true when $\sigma(v_1) = \sigma(v_3) = \sigma(v_5) = \perp$. Suppose, then, that $\sigma(v_i) \neq \perp$, for some $i \in \{1, 3, 5\}$. In this case, $\sigma(v_i)$ dominates v_i, by definition of σ. Again, by (2), we obtain that $\sigma(v_i)$ must be adjacent to v_0, v_2, and v_4, or otherwise v_i could be replaced by $\sigma(v_i)$ so as to obtain an induced C_6 with lower value of R. Then, $\{v_0, v_1, v_2, \sigma(v_i)\}$, $\{v_0, v_5, v_4, \sigma(v_i)\}$, and $\{v_2, v_3, v_4, \sigma(v_i)\}$ all induce C_4's. Since G is C_4-dominated and v_0, v_2, v_4 are pairwise not dom-comparable, it follows that $\sigma(v_i)$ dominates v_1, v_3, and v_5. Thus, none of $\sigma(v_1)$, $\sigma(v_3)$, and $\sigma(v_5)$ is equal to $\perp$. Repeating the above arguments for $i = 1$, 3, and 5, we obtain that $\sigma(v_i)$ dominates v_j for every $i, j \in \{1, 3, 5\}$. Then (iii) follows, because $\sigma(v) = \min N^+_{S(G)}(v)$, for every $v \in V(G)$.

For the converse, suppose that there are two paths v_0, v_1, v_2, v_3, and v_0, v_5, v_4, v_3 that satisfy (i)–(iii). Observe that it is enough to prove that $v_1 \notin N(v_4)$ and $v_5 \notin N(v_2)$. Indeed, in this case $v_1 \neq v_5$, thus $v_0, \ldots, v_5$ is a cycle of G and, as G is triangle-free, this cycle is induced.

Suppose, to obtain a contradiction, that $v_1 \in N(v_4)$. Then, v_1, v_2, v_3, v_4 is a C_4 in G. This cycle is represented by some $S \in \mathcal{S}$. Consider the following cases.

Case 1: $v(S) \in \{v_2, v_4\}$. Observe that $w(S) \in \{v_2, v_4\}$, so $\sigma(w(S)) > v_0 > v(S)$ by (i) and (ii). Then, $v(s)$ does not dominate $w(S)$, thus S is an unsafe triple of $\mathcal{S}$. Consequently, since $v_1, v_3 \in L(S)$ and $v_0 \in N(v_1) \setminus N(v_3)$, it follows that v_1 dominates v_3, thus there is a path from v_3 to v_1 in $U(G)$. Then, $\sigma(v_1) > v_1 \geq \sigma(v_3)$, a contradiction.

Case 2: $v(S) \notin \{v_2, v_4\}$. In this case, both v_2, v_4 belong to $L(S)$, thus, as $\sigma(v_2) > v_0 > v_4$ and $\sigma(v_4) > v_0 > v_2$, we obtain that S must be safe. Consequently, since $v_0 \in N(v_1) \setminus N(v_3)$, it follows that $S = (v_1, v_3)$. This is again impossible, because $\sigma(v_1) > v_1 \geq \sigma(v_3)$.

Replacing v_1 by v_5 above, we obtain that $v_5 \notin N(v_2)$ as well. Thus, $v_0, \ldots, v_5$ is an induced cycle of G. □

Lemma 5.1 implies Algorithm 5. Its input is the graph G, and its output is a message if G contains no induced C_6, or an induced C_6 otherwise. Thus, Algorithm 5 is just a replacement of Algorithm 3. We discuss the algorithm, its correctness, and its complexity in the next paragraphs.

Algorithm 5 Induced C_6 of a triangle-free C_4-dominated graph.

Input: a degree ordered graph G that is C_4-dominated and has no triangles.
Output: if existing, an induced C_6 of G; otherwise, a message.

1. Let $v_1 > \ldots > v_n$ be the vertices of G.

2. Compute $\sigma(v)$, for every $v \in V(G)$.

3. For each $vw \in E(G)$, set $X(v, w) := \{z \in N(v) \mid \sigma(z) = w\}$ and $X(w, v) := \{z \in N(w) \mid \sigma(z) = v\}$.

4. For $i := 1$ to n, do:

5. Set $N_1 := \{w_1 \in N(v_i) \mid w_1 < v_i\}$.

6. Set $N_2 := \bigcup_{w_1 \in N_1} \{w_2 \in N(w_1) \mid \sigma(w_2) > v_i > w_2\}$.

7. For each $w_2 \in N_2$, set $N_3(w_2) := \bigcup \{X(w_2, \sigma(w_1)) \setminus N_1 \mid w_1 \in N_1 \cap N(w_2)\}$.

8. For $w_2, w_4 \in N_2$ ($w_2 \neq w_4$), if $N_3(w_2) \cap N_3(w_4)$ contains a vertex, say w_3, then:

9. Output $v_i, w_1, w_2, w_3, w_4, w_5$, for some $w_1, w_5 \in N_1$ such that $w_1 \neq w_5$ and $\sigma(w_1) = \sigma(w_5) = \sigma(w_3)$, and halt.

10. Output "G contains no induced C_6's."

5.1 Correctness of Algorithm 5

Algorithm 5 either halts at Step 9 or it runs until its termination. In this paragraph, we prove that Algorithm 5 halts at Step 9 if and only if G contains an induced C_6. Furthermore, when Algorithm 5 halts at Step 9, its output is an induced C_6 of G.

Suppose first that Algorithm 5 halts at Step 9, when the i-th iteration of Loop 4–9 is being executed. Examine the state of the variables immediately before Step 9 is executed. By Steps 3, 5 and 7, $w_3 \in N(w_2 w_4) \setminus N(v_i)$, and there are two vertices $w_1 \in N(v_i w_2)$ and $w_5 \in N(v_i w_4)$ such that $\sigma(w_1) = \sigma(w_3) = \sigma(w_5)$. By Steps 5 and 6, $v_i > \max\{w_1, w_2, w_4, w_5\}$, so, by Lemma 5.1, $v_i, w_1, \ldots, w_5$ induce a C_6 in G.

For the converse, suppose that G contains an induced C_6. By Lemma 5.1, there must be two paths w_0, w_1, w_2, w_3 and w_0, w_5, w_4, w_3 satisfying conditions (i)–(iii) of the lemma. Vertex w_0 gets some name v_i in Algorithm 5, for some $1 \leq i \leq n$. If Algorithm 5.1 halts before the i-th iteration of Loop 4–9, then there is nothing to prove. So, suppose that the i-th iteration of Loop 4–9 is executed, and consider its effects. By Step 5 and Lemma 5.1 (i), $w_1, w_5 \in N_1$. By Step 6 and Lemma 5.1 (i)–(ii), $w_2, w_4 \in N_2$. By Step 3 and Lemma 5.1 (iii), w_3 belongs to both $X(w_2, \sigma(w_1))$ and $X(w_4, \sigma(w_5))$, thus, by Step 7 and Lemma 5.1 (i), $w_3 \in N_3(w_2) \cap N_3(w_4)$. Therefore, the condition of Step 8 is satisfied, and Algorithm 5 halts at Step 9.

Finally, observe that Step 9 always finds the vertices $w_1 \neq w_5$ such that $\sigma(w_1) = \sigma(w_5) = \sigma(w_3)$. Indeed, such w_1 and w_5 exist as argued before. And, by Lemma 5.1, $v_i, w_1, \ldots, w_5$ induce C_6 in G. Summing up, Algorithm 5 is correct.

5.2 Implementation and complexity of Algorithm 5

The data structures involved in Algorithm 5 are a little harder than those in Algorithm 4. The input graph G is implemented with adjacency lists, i.e., the list $N(v)$ is stored for each $v \in V(G)$. While $N(v)$ is being iterated, say the next vertex is w, $O(1)$ time access to the position that v occupies in $N(w)$ is required. This can be achieved by keeping a pointer to this position paired with w in $N(v)$. The value of $\sigma(v)$ is stored together with the vertex v. Given a vertex v, $O(1)$ time access to the set $children(v) = \{w \in V(G) \mid \sigma(w) = v\}$ is also required. So, a list with the elements of $children(v)$ is stored together with v. A list with the elements of $X(v, w)$ is stored for every $vw \in E(G)$. The list $X(v, w)$ has to be obtained in $O(1)$ time while w is being examined in a traversal of $N(v)$. Thus, $X(v, w)$ is also paired with w in $N(v)$. Furthermore, for each $z \in X(v, w)$, while traversing $N(z)$ with v as the next vertex, the list $X(v, w)$ has to be accessed in $O(1)$ time. So, a reference $x(z, v)$ pointing to $X(v, w)$ is paired together with v in the list $N(z)$. Recall that $\{X(v, w)\}_{w \in N(v)}$ is a partition of $N(v)$; indeed, z belongs only to $X(v, \sigma(z))$. Thus there is a unique pointer $x(z, v)$ associated with v in $N(z)$. Finally, sets N_1, N_2 are implemented as n-position vectors so that membership can be tested in $O(1)$ time.

Consider the time complexity of Algorithm 5. To compute σ at Step 2 we first set $\sigma(v)$ as the out-neighbor of v in $U(G)$ by calling Algorithm 4; next, we traverse the family $\mathcal{S}$ given by method $C4$ in [1] so as to update the values of σ accordingly. All these steps take $O(m\alpha(G))$ time. Next, by traversing $V(G)$, we compute $children(v)$ in $O(n)$ time. For computing the sets $X(v, w)$ at Step 3, for a given $v \in V(G)$, we use a three step procedure. First, we traverse $N(v)$ and set $X(v, w) = \emptyset$, for each $w \in N(v)$. Second, we sort $N(v)$ according to the value of σ, i.e., w appears before z in $N(v)$ only if $\sigma(w) \geq \sigma(z)$. Third, we traverse $N(v)$ once again and, for each $w \in N(v)$, we insert w into $X(v, \sigma(w))$. These steps take $O(d(v))$ time each; in particular, the third step can be done as in a merging procedure, because $N(v)$ is sorted according to the values of σ. Therefore, Step 3 takes $O(n + m)$ time. Before executing Step 4, we should update the values of the pointers $x(z, v)$. For this, we traverse each $N(v)$ once again as in the merging step and, when z is being traversed so as to be inserted in $X(v, w)$, we access the position of v inside $N(z)$ in $O(1)$ time, and set $x(z, v)$ to point to $X(v, w)$. Therefore, the update of the x pointers takes $O(n + m)$ time as well. To evaluate the time required by Loop 4–9, consider a vertex $v_i \in V(G)$. Step 5 requires only a traversal of $N(v_i)$, so it takes $O(d(v_i))$. For Step 6, it is enough to traverse $N(w)$, for each $w \in N_1$, while σ is accessed in $O(1)$ time. Therefore, Step 5 takes $O(\sum_{w \in N(v_i)} d(w))$ time. Steps 7 and 8 are implemented together, as follows. For Step 7, we traverse each $w_2 \in N(w_1) \cap N_2$, for every $w_1 \in N_1$, and mark every vertex $w_3 \in X(w_2, \sigma(w_1)) \setminus N_1$ with the value (w_1, w_2). The condition at Step 8 is true for v_i if and only if some vertex w_3 is marked twice. Recall that testing membership in N_1 and N_2 takes $O(1)$ time, while $X(w_2, \sigma(w_1))$ can be obtained in $O(1)$ time while examining w_2 in a traversal of $N(w_1)$ with the pointer $x(w_1, w_2)$ (notice that $w_1 \in X(w_2, \sigma(w_1))$ by definition, so $x(w_1, w_2)$ was previously recorded). As each vertex outside N_1 is marked at most twice, the time required by Steps 7 and 8 is $O(n + \sum_{w_1 \in N_1} d(w_1))$. Finally, if the condition at Step 8 is true, then we obtain a vertex w_3 that has two marks, say (w_1, w_2) and (w_5, w_4). These marks indicate that $w_3 \in X(w_2, \sigma(w_1)) \cap X(w_4, \sigma(w_5))$, this $v_i, w_1, \ldots, w_5$ is a valid output for Step 9. Clearly, this step takes $O(1)$ time. Summing up, by [1], Algorithm 5 has time complexity

$$O\left(\alpha(G)m + \sum_{i=1}^{n} \left(\sum\{d(w) + n \mid w \in N(v_i) \cap \mathrm{MIN}(v_i)\}\right)\right) = O(n^2 + \alpha(G)m).$$

As for the space complexity, recall again that w belongs only to $X(v, \sigma(w))$, for every edge vw. Therefore, as each call to method $C4$ takes $O(n + m)$ space, Algorithm 5 requires $O(n + m)$ bits.

5.3 Finding an induced C_5 efficiently

An induced C_5 can be found in $O(n^2 + m\alpha(G))$ time and linear space, if existing, with a procedure similar to Algorithm 5. We omit the implementation details, that follow from the next lemma.

Lemma 5.2 *Let G be a degree ordered graph that is C_4-dominated and contains no triangles. Then G contains an induced C_5 if and only if it contains a cycle $v_0, \ldots, v_4$ such that* $\min\{\sigma(v_2), \sigma(v_3)\} > v_0 > \max\{v_2, v_3\}$.

The main theorem of this paper is the following corollary.

Theorem 5.3 *Hereditary biclique-Helly graphs can be recognized in $O(n^2 + \alpha(G)m)$ time and $O(n+m)$ space.*

6 Maximal bicliques of C_4-dominated graphs with no triangles

In this section we focus on the problem of enumerating all the maximal bicliques of a C_4-dominated graph with no triangles. At the end of this section, we also devote a paragraph to discuss some biclique problems on this class of graphs.

Observe that every pair of twin vertices of any graph G belong to the same maximal bicliques. Thus, every maximal set of twin vertices S can be identified into one vertex $v(S)$ so as to obtain a twin-free graph H. Then, for every maximal biclique B of H, we can replace each $v(S) \in B$ with the set S so as to obtain a maximal biclique of G. So, in this section we assume that G is a degree ordered graph that is C_4-dominated and has no triangles nor twins.

For each $v \in V(G)$, define $B(v) = \{N(v), Dom(v) \cup \{v\}\}$. Observe that, since G is triangle-free, $B(v)$ is a biclique of G. The following theorem shows that $\{B(v) \mid v \in V(G)\}$ is precisely the family of maximal bicliques of G.

Theorem 6.1 *Let G be a degree ordered graph that is C_4-dominated and has no triangles nor twins, and B be a biclique of G. Then, B is a maximal biclique of G if and only if $B = B(v)$ for some $v \in V(G)$.*

Proof: Suppose first that B is a maximal biclique of G, and let $\{B_1, B_2\}$ be its bipartition. Fix $v_1 = \min B_1$ and $v_2 = \min B_2$. To obtain a contradiction, suppose that neither $B_1 \subseteq Dom(v_1)$ nor $B_2 \subseteq$

$Dom(v_2)$. Then, there are two vertices $w_1 \in B_1$ and $w_2 \in B_2$ that do not dominate v_1 and v_2, respectively. On the other hand, since $v_1 < w_1$ and $v_2 < w_2$, we obtain that v_1 and v_2 do not dominate w_1 and w_2, respectively. But this implies that v_1, v_2, w_1, w_2 is a non-dominated C_4 of G, a contradiction. Therefore, without loss of generality, we may assume that $B_1 \subseteq Dom(v_1)$. Now, as B is maximal, it follows that all the vertices that dominate v_1 are included in B_1 and all the neighbors of v_1 are included in B_2.

The converse is trivial. □

Corollary 6.2 *A C_4-dominated graph with no triangles has at most n maximal bicliques.*

Say that $v \in V(G)$ is a *repeated* vertex if $|Dom(v)| = |N(w)|$ and $|N(v)| = |Dom(w)|$, for some $w \in N(v) \cap \mathrm{MAX}(v)$. The following lemma shows how to avoid the listing of duplicate maximal bicliques.

Lemma 6.3 *Let G be a degree ordered graph that is C_4-dominated and has no triangles nor twins, and $v \in V(G)$. Then $B(v) = B(w)$ for some $w \in \mathrm{MAX}(v)$ if and only if v is a repeated vertex.*

Proof: If $B(v) = B(w)$ for some $w \in \mathrm{MAX}(v)$, then, by definition, either $N(v) = N(w)$ or $N(v) = Dom(w)$ and $Dom(v) = N(w)$. Since $N(v) \neq N(w)$ because G has no twins, the former is impossible. For the converse, suppose that v is a repeated vertex, i.e., $|Dom(v)| = |N(w)|$ and $|N(v)| = |Dom(w)|$ for some $w \in N(v) \cap \mathrm{MAX}(v)$. Now, as every vertex in $Dom(v)$ is adjacent to w and every vertex in $Dom(w)$ is adjacent to v, it follows that $Dom(v) = N(w)$ and $N(v) = Dom(w)$, thus $B(v) = B(w)$. □

Theorem 6.1 and Lemma 6.3 yield a simple $O(nm)$ time and $O(n^2)$ space algorithm for enumerating all the maximal bicliques of G. First, find the set R of repeated vertices by using the domination matrix; then, output $B(v)$ for every $v \in V(G) \setminus R$. In the remaining of this section, we show an $O(m\alpha(G)+o)$ time and $O(m\alpha(G))$ space algorithm for enumerating all the maximal bicliques of G, where $o = \sum_{v \in V(G)} |B(v)|$ is the size of the output. Our main tool is, in this case, a digraph that encodes all the dominations of the input graph as paths. This digraph is obtained from the squares domination digraph $S(G)$ by adding all the missing dominations.

Recall that $S(G)$ is obtained from $U(G)$ by inserting the edge $w \to v$, for every safe $(v, w) \in \mathcal{S}$. Say that $w \in V(G)$ is *degenerated* when w has no out-neighbors in $S(G)$ and $N(w) \subseteq \mathrm{MAX}(w)$. We define the *dominator set* of w according to the following rules. If $L(S) = \{w\}$ for some $S \in \mathcal{S}$, then the dominator set of w is empty. Otherwise, the dominator set of w is $N(z) \setminus \{w\}$, where $z = \min N(w)$. The following lemma shows that degenerated vertices, together with their dominator sets, are all we need to build the domination digraph from $S(G)$.

Lemma 6.4 *Let G be a degree ordered graph that is C_4-dominated and has no triangles nor twins, and $w \in V(G)$ be degenerated. Then, w is dominated by $v \in V(G)$ if and only if v belongs to the dominator set of w.*

Proof: Suppose first that w is dominated by $v \in V(G)$, and let $z = \min N(w)$. Observe that $v < z$; otherwise, (v, w) would be a safe triple, contradicting the fact that w has no out-neighbors in $S(G)$. Then, since v dominates w, it follows that $v \in L(S)$ for every $S \in \mathcal{S}$ such that $w \in L(S)$. Therefore,

$L(S) \neq \{w\}$ for every $S \in \mathcal{S}$, thus the dominator set of w is $N(z) \setminus \{w\}$. Hence, v belongs to the dominator set of w.

For the converse, suppose that v belongs to the dominator set of w, and call $z = \min N(w)$. By definition, $v \in N(z)$. Note that, if $N(w) = \{z\}$, then w is dominated v. Suppose then that $d(w) > 1$, and take $a \in N(w) \setminus \{z\}$. Since $N(w) \subseteq \mathrm{MAX}(w)$, we obtain that $a > z > w$, thus $(a, z) \in \mathcal{S}$. Also, $L(a, z) \neq \{w\}$, because otherwise the dominator set of w would be empty, and this cannot happen as it contains v. Furthermore, since w has no out-neighbors in $S(G)$ and $|L(a, z)| > 1$, it follows that (a, z) is a safe triple of $\mathcal{S}$. Therefore, a dominates z, hence a is adjacent to v. Since a is any vertex in $N(w) \setminus \{z\}$, it follows that v is adjacent to all the vertices in $N(w)$, hence v dominates w. □

The *domination digraph* of G is the digraph $D(G)$ that is obtained from $S(G)$ by inserting an edge $w \to v$, for every degenerated $w \in V(G)$ and every v in the dominator set of w. As its name indicates, $D(G)$ encodes all the dominations in G, as the following theorem resumes.

Theorem 6.5 *Let G be a degree ordered graph that is C_4-dominated and has no triangles nor twins, and $v, w \in V(G)$. Then, v dominates w in G if and only if there is a path from w to v in $D(G)$. Furthermore, if v dominates w and $w \not\to_{D(G)} v$, then there is a path from w to v in $U(G)$.*

Proof: Suppose that w is dominated by v. If w is degenerated, then $w \to v$ is an edge of $D(G)$, by Lemma 6.4. Otherwise, either w has a neighbor in $\mathrm{MIN}(w)$ or w has an out-neighbor in $S(G)$. Consider these alternatives:

Alternative 1: w is adjacent to $z \in \mathrm{MIN}(w)$. In this case, $z \in L(v, w)$, thus (v, w) is a safe triple of $\mathcal{S}$. Therefore, v is an out-neighbor of w in both $S(G)$ and $D(G)$.

Alternative 2: $N(w) \subseteq \mathrm{MAX}(w)$ and $d^+_{U(G)}(w) = 0$. In this case, w has an out-neighbor z in $S(G)$ such that (z, w) is a safe triple of $\mathcal{S}$. Then, as v dominates w, it follows that v is adjacent to all the vertices in $L(z, w)$. Therefore, $L(v, w) \neq \emptyset$, thus (v, w) is a safe triple of $\mathcal{S}$. Consequently, $w \to v$ is an edge of both $S(G)$ and $D(G)$.

Alternative 3: w has an out-neighbor in $U(G)$, say z. For this to happen, there must be an unsafe triple $S \in \mathcal{S}$ such that $L(S)$ contains both w and z. If $v(S) > v$, then $L(S)$ contains also v, and so there is a path from w to v in both $U(G)$ and $D(G)$. Otherwise, $v(S) < v$ and, since v dominates w, it follows that $v(S) \in L(v, w)$. Hence, (v, w) is a safe triple of $\mathcal{S}$, and $w \to v$ is an edge of both $S(G)$ and $D(G)$.

The converse follows from Lemmas 4.1 and 6.4 and the definition of $D(G)$, while alternatives above are enough to conclude that there is a path from w to v in $U(G)$ for every v, w such that v dominates w and $w \not\to_{D(G)} v$. □

Corollary 6.6 *Let G be a degree ordered graph that is C_4-dominated and has no triangles nor twins. For every $v \in V(G)$, $Dom(v)$ is equal to the set of ancestors of v in $D(G)$.*

The above corollary can be used to improve the algorithm for listing the maximal bicliques, as it is shown in Algorithm 6. The correctness of Algorithm 6 follows from Lemmas 6.3 and 6.4 and Corollary 6.6. With respect to its implementation, the domination matrix is represented simply with successors and predecessors lists, $d(v)$ and $dom(v)$ are stored together with v, and R is stored in an vertex with n positions, so that each membership query takes $O(1)$ time.

Algorithm 6 Maximal bicliques of a C_4-dominated graph with no triangles.

Input: a degree ordered graph G that is C_4-dominated and has no triangles nor twins.
Output: a listing, without duplicates, of all the maximal bicliques in G.

1. Compute the domination matrix $D(G)$.

2. For each v, set $dom(v)$ as the number of ancestors of v in $D(G)$.

3. Set $R := \{v \in V(G) \mid d(w) = dom(v) \text{ and } d(v) = dom(w), \text{ for some } w \in N(v)\}$.

4. For every $v \in V(G) \setminus R$, write $\{Dom(v), N(v)\}$.

6.1 Complexity of Algorithm 6

The following observation is useful in the complexity analysis.

Proposition 6.7 *If G is a degree ordered graph that is C_4-dominated and has no triangles nor twins, then $D(G)$ has at most $2m$ edges more than $S(G)$.*

Proof: By definition, if $w \to v$ is and edge that belongs to $D(G)$ and does not belong to $S(G)$, then w is a degenerated vertex and v belongs to the dominator set of w. Recall that, by Theorem 6.5, the dominator set of w is $Dom(w)$, because w has no out-neighbors in $S(G)$.

Suppose, to obtain a contradiction, that w_1 and w_2 are degenerated vertices with nonempty dominator sets, and call $z_1 = \min N(w_1)$ and $z_2 = \min N(w_2)$. If $z_1 = z_2$, then $w_1 \in N(z_2) \setminus w_2 = Dom(w_1)$; analogously, $w_2 \in Dom(w_1)$. But this contradicts the fact that w_1 and w_2 are not twins. Consequently, $\sum\{|Dom(w)| \mid w \in V(G) \text{ and } w \text{ is degenerated}\} \leq \sum_{z \in V(G)} |N(z)|$, thus $D(G)$ has at most $2m$ edges more than $S(G)$. □

Matrix $D(G)$ can be computed with a five steps procedure, as follows. First, call Algorithm 4 so as to obtain $U(G)$. Second, traverse each $S \in \mathcal{S}$ as in Algorithm 4 so as to determine whether S is safe or not, and, if S is safe, insert $w(S) \to v(S)$ into $U(G)$ to obtain $S(G)$. Third, find and mark all the degenerated vertices, by traversing each $w \in V(G)$ while querying whether w has out-neighbors in $S(G)$ and $N(w) \subseteq \mathrm{MAX}(w)$. Forth, find those degenerated vertices with nonempty Dom, by removing the mark to all those vertices that belong to $L(S)$, for some $S \in \mathcal{S}$ such that $|L(S)| = 1$. This forth step only requires the traversal of $\mathcal{S}$. Finally, $D(G)$ is obtained by inserting an edge $w \to v$ into $S(G)$, for every marked w and every $v \in Dom(w)$. By Proposition 6.7, $O(m)$ edges are inserted into $S(G)$ in this last step. So, since the first four steps take $O(\alpha(G)m)$ time, Step 1 of Algorithm 6 takes $O(\alpha(G)m)$ time.

Step 2 can be easily implemented so as to run in $O(\alpha(G)m)$ time, by observing that, by Theorem 6.5 $dom(v) = d^+_{D(G)} + \ell(v)$, where $\ell(v)$ is the length of the unique maximal path of $U(G)$ beginning at v. Step 3 clearly takes $O(n+m)$ time. Finally, Step 4 takes $O(o)$ time, where $o = \sum_{v \in V(G)} |B(v)| \leq n^2$ is the size of the output.

As for the space complexity, $D(G)$ is the heaviest structure stored by Algorithm 6. Thus, Algorithm 6 requires $O(\alpha(G)m)$ space by Proposition 6.7.

6.2 Some biclique problems

Theorems 6.1 and 6.5 can also be used to solve many problems that involve finding a biclique with a certain property. Just observe that $|Dom(v)|$ can be computed in $O(\alpha(G)m)$ time and $O(n+m)$ space. Indeed, as in Algorithm 6, $|Dom(v)|$ is either the level of v in $U(G)$ plus the number of safe triples $(w, v) \in \mathcal{S}$ with $v \not\to_{U(G)} w$ (when v is not degenerate) or the number of vertices in its dominator set (when v is degenerate). (Twin vertices are ignored by Theorems 6.1 and 6.5; nevertheless, they can be counted as well when $|Dom(v)|$ is being computed.) Therefore, for $k \in \mathbb{N}$, we can test, in $O(\alpha(G)m)$ time and $O(n+m)$ space, whether G has a biclique (B_1, B_2) such that: a) $|B_1| = |B_2| = k$, b) $|B_1| + |B_2| \geq k$, or $|B_1| \cdot |B_2| \geq k$. These problems are respectively called: a) balanced biclique problem, b) maximum vertex biclique problem, and c) maximum edge biclique problem, and are all NP-complete for general graphs [4, 16]. Furthermore, problem c) is NP-complete even when G is a bipartite graph [16].

7 Concluding remarks and open problems

In this paper we devised an efficient algorithm for the recognition of hereditary biclique-Helly graphs. In this section we leave some questions whose answers could imply a better algorithm.

In Section 5, we showed how to find an induced C_6 in a C_4-dominated graph with no triangles. As part of Algorithm 5, we had to find some vertices that appear at distance at most 3 from a given vertex v. We showed how to do this in $O(n)$ time, by traversing each vertex outside $N(v)$ at most twice. However, not all the vertices outside $N(v)$ are at distance 3 from G, unless G is somehow dense. An open question that follows is, then, can an induced C_6 be found in $O(\alpha(G)m)$ time? Furthermore, is it possible to do find such C_6 by using only $O(n+m)$ space?

In Section 6, we develop an algorithm for enumerating the bicliques in $O(o + \alpha(G)m)$ time and $O(\alpha(G)m)$ space. Algorithm 6 uses the domination digraph $D(G)$ that encodes all the dominations in $O(\alpha(G)m)$ space. We can divide the dominations encoded in $D(G)$ into three classes. Those corresponding to paths in $U(G)$; those corresponding to edges in $S(G) - U(G)$; and those corresponding to edges in $D(G) - S(G)$. Although there can be as many as $O(\sum_{v \in V(G)} Dom(v)) = O(o)$ dominations of the first class, only $O(n)$ space is used by $D(G)$ to encode this information. On the other hand, Proposition 6.7 guaranties that there are only $O(m)$ dominations of the third class. Finally, $O(\alpha(G)m)$ bits are used to store the $O(\alpha(G)m)$ dominations second class. Could it be possible to encode such dominations in a different way (e.g. as paths), so that only $O(n+m)$ space is used, without affecting the time complexity of the algorithm?

The somehow opposite question is also interesting. Instead of implicitly storing each domination of the first class as a path, we can store it explicitly. That is, we can insert an edge $w \to v$ into $D(G)$ for every path of $U(G)$ beginning at w and ending at v. Such digraph would require $O(o)$ space. We know that $o < n^2$; however, for w to be dominated by a vertex at distance k from v in $D(G)$, there should be at least k vertices in $N(w) \setminus N(v)$ (always assuming that G is twin-free). Thus if $D(G)$ has many large paths, then G is dense and $\alpha(G)$ is large. So, it would be nice to prove or disprove that $o \in O(\alpha(G)m)$, or that $o \in O(m^{3/2})$.

Let $\mathcal{B}$ be the class of graph formed by those graphs G such that $\{B(v)\}_{v \in V(G)}$ is the family of maximal bicliques of G. By Theorem 6.1, every C_4-dominated graph with no triangles belongs to $\mathcal{B}$. The converse is false, since the graph in Figure 3 belongs to $\mathcal{B}$ and it contains a non-dominated C_4. The graphs in $\mathcal{B}$ have at most n bicliques, and Lemma 6.3 holds for all the graphs in $\mathcal{B}$. Hence, the simple algorithm that lists $\{B(v)\}_{v \in V(G)}$ by using the domination matrix, implies that the maximal bicliques of the graphs in $\mathcal{B}$ can be enumerated in $O(nm)$ time and $O(n^2)$ space. Is it possible to list the maximal bicliques of the graphs in $\mathcal{B}$ faster? Is it possible obtain the same time and space bounds of Algorithm 6 for a larger class of graphs?

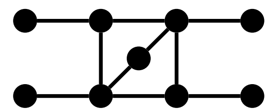

Fig. 3: A graph G that contains a non-dominated C_4, whose family of maximal bicliques is $\{B(v)\}_{v \in V(G)}$.

References

[1] Norishige Chiba and Takao Nishizeki. Arboricity and subgraph listing algorithms. *SIAM J. Comput.*, 14(1):210–223, 1985.

[2] Vânia M. F. Dias, Celina M. H. de Figueiredo, and Jayme L. Szwarcfiter. Generating bicliques of a graph in lexicographic order. *Theoret. Comput. Sci.*, 337(1-3):240–248, 2005.

[3] Mitre C. Dourado, Fábio Protti, and Jayme L. Szwarcfiter. Complexity aspects of the Helly property: graphs and hypergraphs. *Electron. J. Comb.*, DS17:53 pp., 2009.

[4] Michael R. Garey and David S. Johnson. *Computers and intractability*. W. H. Freeman and Co., San Francisco, Calif., 1979. A guide to the theory of NP-completeness, A Series of Books in the Mathematical Sciences.

[5] Alain Gély, Lhouari Nourine, and Bachir Sadi. Enumeration aspects of maximal cliques and bicliques. *Discrete Appl. Math.*, 157(7):1447–1459, 2009.

[6] Marina Groshaus. *Bicliques, cliques, neighborhoods, and the Helly property*. PhD thesis, Universidad de Buenos Aires, 2006.

[7] Marina Groshaus and Leandro Montero. On the iterated biclique operator. In Isabel Méndez-Díaz and Paula Zabala, editors, *Proceedings of the VI ALIO/EURO Workshop on Applied Combinatorial Optimization*, pages CD–ROM version, 2008.

[8] Marina Groshaus and Leandro Montero. The number of convergent graphs under the biclique operator with no twin vertices is finite. In *Proceedings of the V Latin-American Algorithms, Graphs and Optimization Symposium - LAGOS' 2009*, Electronic Notes in Disc. Math. Elsevier, 2009.

[9] Marina Groshaus and Jayme L. Szwarcfiter. Biclique-Helly graphs. *Graphs Combin.*, 23(6):633–645, 2007.

[10] Marina Groshaus and Jayme L. Szwarcfiter. On hereditary Helly classes of graphs. *Discrete Math. Theor. Comput. Sci.*, 10(1):71–78, 2008.

[11] Marina Groshaus and Jayme L. Szwarcfiter. Biclique graphs and biclique matrices. *J. Graph Theory*, 63(1):1–16, 2010.

[12] Kazuhisa Makino and Takeaki Uno. New algorithms for enumerating all maximal cliques. In *Algorithm theory—SWAT 2004*, volume 3111 of *Lecture Notes in Comput. Sci.*, pages 260–272. Springer, Berlin, 2004.

[13] Terry A. McKee and F. R. McMorris. *Topics in intersection graph theory*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1999.

[14] Leandro Montero. Convergencia y divergencia del grafo biclique iterado. Master's thesis, Universidad de Buenos Aires, 2008.

[15] Lhouari Nourine and Olivier Raynaud. A fast algorithm for building lattices. *Inform. Process. Lett.*, 71(5-6):199–204, 1999.

[16] René Peeters. The maximum edge biclique problem is NP-complete. *Discrete Appl. Math.*, 131(3): 651–654, 2003.

[17] Erich Prisner. Bicliques in graphs. I. Bounds on their number. *Combinatorica*, 20(1):109–117, 2000.

[18] Jayme Luiz Szwarcfiter. A survey on clique graphs. In C. Linhares Sales and B. Reed, editors, *Recent advances in algorithms and combinatorics*, volume 11 of *CMS Books Math./Ouvrages Math. SMC*, pages 109–136. Springer, New York, 2003.

[19] Pablo Terlisky. Biclique-coloreo de grafos. Master's thesis, Universidad de Buenos Aires, 2010.