Consensus algorithms for the generation of all maximal bicliques[1]

Gabriela Alexe [2], Sorin Alexe [3], Yves Crama [4], Stephan Foldes [5], Peter L. Hammer [6], and Bruno Simeone [7]

Abstract. We describe a new algorithm for generating all maximal bicliques (i.e. complete bipartite, not necessarily induced subgraphs) of a graph. The algorithm is inspired by, and is quite similar to, the consensus method used in propositional logic. We show that some variants of the algorithm are totally polynomial, and even incrementally polynomial. The total complexity of the most efficient variant of the algorithms presented here is polynomial in the input size, and only linear in the output size. Computational experiments demonstrate its high efficiency on randomly generated graphs with up to 2,000 vertices and 20,000 edges.

1 Introduction

The literature dealing with complete bipartite (not necessarily induced) subgraphs of a graph is rich and diverse. The problem of covering the edge set of a graph by a smallest family of complete bipartite subgraphs has received considerable attention. Chung [Chu80] proved the conjecture of Bermond [Ber78] asserting that $\lim_{n\to\infty} \frac{\rho(n)}{n} = 1$, where $\rho(n)$ is the minimum number of complete bipartite subgraphs that cover the edges of a graph with n vertices. In [CES83], Chung, Erdös, and Spencer examined the similar problem of partitioning the edge set of a graph into complete bipartite subgraphs. Among the many contributions to this direction of research we mention Chung [Chu81], Tuza [Tuz83], Tayur [DKT97], Hochbaum [Hoc98], Idzik [BIK99], Lundgren [DLS99].

[1] **Acknowledgements**. This work was partially supported by NSF grant DMS-9806389, ONR grant N00014-92-J-1375, and DIMACS.

[2] RUTCOR, Rutgers, the State University of New Jersey, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA. e-mail: alexe@rutcor.rutgers.edu

[3] RUTCOR, Rutgers, the State University of New Jersey, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA. e-mail: salexe@rutcor.rutgers.edu

[4] Ecole d'Administration des Affaires, University of Liège, Boulevard du Rectorat 7 (B31), 4000 Liège, Belgium. e-mail: y.crama@ulg.ac.be

[5] Tampere University of Technology, Department of Mathematics, Tampere-Hervanta, Finland. e-mail: stephan.foldes@tut.fi

[6] RUTCOR, Rutgers, the State University of New Jersey, 640 Bartholomew Road, Piscataway, NJ 08854-8003, USA. e-mail: hammer@rutcor.rutgers.edu

[7] Department of Statistics, "La Sapienza" University, Piazzale Aldo Moro 5, 00185, Rome, Italy. e-mail: marsalis@rosd.sta.uniroma1.it

Hammer [Ham78] noticed that, to every covering of the edge set of a graph G by a family $\mathcal{B}$ of complete bipartite subgraphs, one can associate a pseudo-Boolean function $f_{\mathcal{B}}$, i.e. a real-valued function in 0-1 variables, whose variables are in one-to-one correspondence with the complete bipartite subgraphs in $\mathcal{B}$, and whose maximum equals the stability number of G. This observation lead to the establishment of various links between problems in graph theory and pseudo-Boolean functions (see [HS79], [BBHS83], [BHS80], [CH89], [FH00]). In particular, *quadratic graphs* were defined in [HS79] as graphs with the property that, among the associated pseudo-Boolean functions, there is at least one which can be represented as a degree-two polynomial. Also, the same observation was central in the elaboration of the *struction* method [EHW84] for calculating the stability number of a graph; it was proved in [HMW85_1], [HMW85_2], and [GH88] that the struction method provides polynomial solutions for finding the stability number of certain classes of graphs.

Note that, since the number of variables of the pseudo-Boolean function $f_{\mathcal{B}}$ in the above construction is equal to the size of $\mathcal{B}$, it is clearly advantageous to determine families $\mathcal{B}$ consisting of the smallest possible number of complete bipartite subgraphs covering the edge set of G. It is also clear that in order to achieve this aim, we can restrict our attention to coverings which use only maximal bipartite subgraphs.

Another important problem which requires the enumeration of the maximal bipartite subgraphs of a graph arises in the context of data compression, as described in [AAAS93].

The goal of this paper is to describe an algorithm for generating all the maximal complete bipartite subgraphs of a graph. Let $G = (V, E)$ be a graph without loops and multiple edges, having the vertex set $V = \{1, 2, ..., n\}$ and the edge set E. In this paper we shall call the (maximal) complete bipartite – not necessarily induced – subgraphs of a graph, its (maximal) *bicliques.*

The *Maximal Biclique Generation Problem* ($MBGP$) consists in generating all the maximal bicliques of a given graph. As already observed by Eppstein [Epp94], the $MBGP$ cannot be solved in polynomial time with respect to the input size, since the size of the output can be exponentially large in n. Indeed, consider for example a complete graph with n vertices. Since each proper partition of its vertex set into two subsets induces exactly one maximal biclique, the number of its maximal bicliques is $2^{n-1} - 1$. Also, notice that $MBGP$ is at least as hard as the NP-hard *Edge Biclique Problem*, i.e. the problem of finding a maximum weight biclique of an edge weighted graph [DKT97].

In this paper we shall describe a consensus algorithm for solving $MBGP$.

The algorithm closely resembles the Blake and Quine classic "consensus method" for finding all the prime implicants of a Boolean function. We develop the basic steps of this algorithm in section 2. In section 3, we show that appropriate modifications of the consensus method yield *total polynomial* algorithms for $MBGP$, i.e. algorithms which are polynomial in the total combined size of the input and of the output. We also provide a reduction of $MBGP$ to the problem of generating all maximal cliques of a graph, yielding another total polynomial algorithm for $MBGP$. In section 4, we describe two classes of graphs for which the consensus algorithm provides an input-polynomial solution to $MBGP$. Section 5 presents briefly an implementation of the consensus algorithm. Finally, in section 7 we present the results of numerous computational experiments which indicate clearly the high efficiency of the proposed consensus procedures.

2 The consensus algorithm

We shall describe below a solution method of the $MBGP$, which resembles closely the consensus method of Blake [Bla37] and Quine [Qui55] for finding the prime implicants of a Boolean function. According to [KP77], [Pic77], Malgrange has developed a consensus approach in [Mal62] to find in a $0-1$ matrix the maximal submatrices consisting only of ones – a problem which is clearly equivalent to the $MBGP$ in the case of *bipartite* graphs.

Let us introduce now some definitions. Let G be a graph and let X, Y be two disjoint non-empty subsets of the vertex set, with the property that every vertex in X is adjacent to every vertex in Y. The biclique of G having the bipartition sets X and Y will be denoted by (X, Y). (Note that $(X, Y) = (Y, X)$).

Let $B_1 = (X_1, Y_1)$ and $B_2 = (X_2, Y_2)$ be two bicliques of G. We say that B_1 *absorbs* or *contains* B_2 if $X_2 \subseteq X_1$ and $Y_2 \subseteq Y_1$, or if $X_2 \subseteq Y_1$ and $Y_2 \subseteq X_1$.

If $Y_1 \cap Y_2 \neq \emptyset$, we call $(X_1 \cup X_2, Y_1 \cap Y_2)$ one of the *consensuses* of B_1 and B_2. Similarly, each of those pairs of subsets $(X_1 \cap X_2, Y_1 \cup Y_2)$, $(Y_1 \cup X_2, X_1 \cap Y_2)$, $(X_1 \cup Y_2, Y_1 \cap X_2)$ which define bicliques (i.e. which involve two non-empty subsets) are *consensuses* of B_1 and B_2. In this way, a pair of bicliques may have $0, 1, 2, 3$, or 4 consensuses.

A *consensus* approach to $MBGP$ starts with a collection $\mathcal{C}$ of bicliques which covers the edge set of a graph G. Such a collection is easily available, for instance by simply considering all the individual edges of the graph, viewed as bicliques. A similar straightforward way of obtaining $\mathcal{C}$ is to define it as the collection of all the stars centered in the vertices of the graph G.

Using the above terminology we can now define a *consensus algorithm* as a sequence of transformations on the collection $\mathcal{C}$. The method applies repeatedly two transformations, the *absorption* and the *consensus adjunction* – described below – and stops when none of these steps can be applied.

(*i*) *Absorption*: If the biclique B_1 in $\mathcal{C}$ absorbs the biclique B_2 in $\mathcal{C}$, then remove B_2 from $\mathcal{C}$.

(*ii*) *Consensus adjunction*: For any two bicliques B_1 and B_2 in $\mathcal{C}$, if any of the consensuses of B_1 and B_2 exists and is not absorbed by a biclique already in $\mathcal{C}$, then add it to $\mathcal{C}$.

Two trivial observations are in order. First, if the collection $\mathcal{C}$ covers the edge set (i.e. every edge of G is contained in at least one of the bicliques of $\mathcal{C}$), then this property will be preserved by both of the transformations above. Second, it is clear that the repeated application of the transformations above will always produce collections consisting only of bicliques of G.

The validity of the consensus approach is based on the following result.

Theorem 1. *If C is a collection of bicliques of the graph G which covers the edge set of G, and if $\widehat{\mathcal{C}}$ is the collection of bicliques obtained from C by repeating the transformations in the consensus algorithm described above as many times as possible, then $\widehat{\mathcal{C}}$ consists of all the maximal bicliques of G.*

Proof. First of all, let us notice that the algorithm terminates after a finite number of steps since the total number of bicliques of G is finite, and a biclique which is removed from $\mathcal{C}$ (because of absorption) can never re-enter $\mathcal{C}$.

We claim that, when the algorithm terminates with the list $\widehat{\mathcal{C}}$, every biclique of G is absorbed by a member of $\widehat{\mathcal{C}}$. This implies, in particular, that $\widehat{\mathcal{C}}$ contains all the maximal bicliques of G, and no other biclique (since the absorption step cannot be applied any longer).

In order to prove the claim, we consider a biclique $B^* = (X^*, Y^*)$ and we proceed by induction on $k = |X^*| + |Y^*|$. Note that $k \geq 2$ since both X^* and Y^* are nonempty, by definition of bicliques.

If $k = 2$, then the claim holds since the initial collection $\mathcal{C}$ covers the edges of G.

If $k \geq 3$, then we can assume without loss of generality that X^* contains at least two vertices. Let us partition X^* into two proper subsets X' and X''. The bicliques $B' = (X', Y^*)$ and $B'' = (X'', Y^*)$ have strictly fewer edges than B^*. Therefore, by induction, each of these bicliques is absorbed by a biclique in $\widehat{\mathcal{C}}$: say B' is absorbed by $B_1 \in \widehat{\mathcal{C}}$ and B'' is absorbed by $B_2 \in \widehat{\mathcal{C}}$. Then, the (unique) consensus $B^* = (X' \cup X'', Y^*)$ of B' and B''

is absorbed by the corresponding consensus of B_1 and B_2, which is itself absorbed by some biclique B_3 in $\widehat{\mathcal{C}}$ (since consensus adjunction cannot be performed on $\widehat{\mathcal{C}}$). This shows that B^* is absorbed by a member of $\widehat{\mathcal{C}}$, and the proof is complete. ■

Example 1. Consider the graph $G = (V, E)$ with vertex-set $V = \{a, b, c, d, e, f\}$ and edge-set $E = \{(a, b), (a, c), (a, d), (b, c), (b, e), (c, f)\}$.

Let us consider the following family of bicliques covering the edge set of G : $\mathcal{C} = \{B_1, B_2, B_3\}$, where $B_1 = (\{a\}, \{b, d\})$, $B_2 = (\{b\}, \{e\})$, and $B_3 = (\{c\}, \{a, b, f\})$. Starting with $\mathcal{C}$, the application of the above algorithm results in the following steps:

(1) The only consensus of B_1 and B_2 is $B_4 = (\{b\}, \{a, e\})$, which we add to $\mathcal{C}$.

(2) B_2, being absorbed by B_4, can be removed from $\mathcal{C}$.

(3) The consensuses of B_1 and B_3 are $B_5 = (\{a\}, \{b, c, d\})$ and $B_6 = (\{b\}, \{a, c\})$, and can be added to $\mathcal{C}$.

(4) B_1 is absorbed by B_5 and can be removed from $\mathcal{C}$.

(5) The consensuses of B_3 and B_4 are $(\{a\}, \{b, c\})$ (which, being absorbed by B_5, will not be added to $\mathcal{C}$) and $B_7 = (\{b\}, \{a, c, e\})$, which is added to $\mathcal{C}$.

(6) We remove B_4 and B_6 from $\mathcal{C}$, both being absorbed by B_7.

Any consensus of any of the remaining bicliques B_3, B_5, and B_7 is absorbed by one of these bicliques, and cannot be added to $\mathcal{C}$.

Therefore, the algorithm terminates with the collection of all maximal bicliques of G : $\widehat{\mathcal{C}} = \{(\{a\}, \{b, c, d\}), (\{b\}, \{a, c, e\}), (\{c\}, \{a, b, f\})\}$.

Obviously, the consensus adjunction and the absorption operations could have been applied in another order, but the resulting collection $\widehat{\mathcal{C}}$ – as shown by Theorem 1 – would have been the same. ■

3 Polynomial versions of the consensus algorithm

In this section we shall show that appropriate modifications of the consensus algorithm allow to substantially improve its worst-case complexity. We first need some terminology.

Let G be a graph and let $\mathcal{A}$ be an algorithm for $MBGP$. We denote by $n(G)$ (or simply n) the number of vertices of the graph G, and by $\beta(G)$ (or simply β) the number of maximal bicliques of G. When running on the instance G, $\mathcal{A}$ successively outputs the maximal bicliques $B_1, B_2, \ldots, B_\beta$. We denote by $\tau(k)$ the running time of $\mathcal{A}$ until it outputs B_k, for $k =$

$1, 2, \ldots, \beta$. Moreover, we let $\tau(0) = 0$ and we denote by $\tau(\beta + 1)$ the total running time of $\mathcal{A}$.

Following Johnson, Yannakakis, and Papadimitriou [JYP88], we say that

- $\mathcal{A}$ runs in *polynomial total time* if its total running time $\tau(\beta + 1)$ is polynomially bounded in n and β;
- $\mathcal{A}$ runs in *incremental polynomial time* if $\tau(k) - \tau(k-1)$ is polynomially bounded in n and k, for $k = 1, 2, \ldots, \beta + 1$;
- $\mathcal{A}$ runs with *polynomial delay* if $\tau(k) - \tau(k-1)$ is polynomially bounded in n, for $k = 1, 2, \ldots, \beta + 1$.

(Our definition of incremental polynomial time is slightly different from that in [JYP88], but the difference will be immaterial for our purpose.)

Note that the running time of a totally polynomial algorithm may be exponentially large in n before it even outputs the first maximal biclique, whereas the time elapsed between the output of the $(k-1)$-st and k-th maximal biclique is reasonably short (polynomially bounded in n and k) for an incrementally polynomial algorithm, and even shorter (polynomially bounded in n) for a polynomial delay algorithm.

These different versions of "output-sensitive"polynomiality have proved useful in previous investigations of algorithms for generating combinatorial objects; for early references, see e.g. Paull and Unger [PU59], Tsukiyama, Ide, Ariyoshi, and Shirakawa [TIAS77], Lawler, Lenstra, and Rinnooy Kan [LLK80], or Johnson, Yannakakis, and Papadimitriou [JYP88]. We are now going to examine different versions of the consensus algorithm for the Maximal Biclique Generation Problem, and evaluate their complexities.

3.1 Complexity of the consensus algorithm

The basic consensus algorithm, as presented in the previous section, does not run in polynomial total running time: more precisely, its running time may be exponential in the size of the input graph *and* in the number of its maximal bicliques, because it can produce along the way a large number of non-maximal bicliques. These comments are illustrated by the next example.

Example 2. Let us consider the complete bipartite graph $K_{n,n}$ on two disjoint $n-$element sets of vertices, and consider as $\mathcal{C}$ the collection of all its individual edges (each viewed as a complete bipartite subgraph). Let us apply now the consensus method in such a way that it produces all the

bicliques having exactly n vertices in the first sequence of transformations. However, the number of those bicliques is $\sum_{i=1}^{n-1} \binom{n}{i}\binom{n}{n-i} = 2^n - 2$, and therefore, the algorithm runs in exponential time with this particular order of transformations.

On the other hand, let $V' = \{v'_1, ..., v'_n\}$ and $V'' = \{v''_1, ..., v''_n\}$ be the bipartition of the vertex-set of $K_{n,n}$ such that $K_{n,n} = (V', V'')$. By forming the consensus of $(\{v'_1\}, \{v''_1\})$ with $(\{v'_1\}, \{v''_2\})$, we obtain the biclique $(\{v'_1\}, \{v''_1, v''_2\})$. In the next steps we shall form in a similar way the bicliques $(\{v'_1\}, \{v''_1, v''_2, v''_3\})$ and so on, until we form $(\{v'_1\}, V'')$. In a similar way we can produce all the bicliques $(\{v'_i\}, V'')$, $i = 2, ..., n$. The number of transformations in this process is of $n(n-1)$ consensus adjunctions and $2n(n-1)$ absorptions. Forming now the consensuses of two $K_{1,n}$'s we obtain a $K_{2,n}$. The consensus of this $K_{2,n}$ with another $K_{1,n}$ is a $K_{3,n}$. Continuing in this way, in $n-1$ steps of consensus adjunction and $2(n-1)$ steps of absorptions, we produce $K_{n,n}$, showing that with a proper order of transformations, the total running time of the algorithm is polynomial in its input and output size (and in this particular case, in the input size only). ■

In the next sections, we are going to show that, by adequate modifications of the steps of the consensus algorithm, it can be guaranteed to run in polynomial total time, and even in incrementally polynomial time.

3.2 An incrementally polynomial consensus algorithm

The new version of the consensus algorithm – which we shall call the *modular consensus algorithm (MCA)* – relies on a simple idea: whenever a consensus adjunction operation is performed, we want to make sure that a maximal biclique is added to the current list $\mathcal{C}$. In this way, $\mathcal{C}$ only contains maximal bicliques throughout the execution of the algorithm, and hence its length never exceeds β. (A similar idea has been used by Boros, Crama and Hammer [BCH90] for the generation of all prime implicants of certain Boolean functions.)

For this purpose, let us define a new operation which can be easily performed on an arbitrary biclique B:

(*iii*) *Extension of* B: Generate a maximal biclique which absorbs B.

Let us now define the

Modular consensus algorithm (*MCA*)

- **Start** with a list $\mathcal{C}_0$ of at most m maximal bicliques that cover the edges of the graph G. Let $\mathcal{C} := \mathcal{C}_0$.
- **Repeat:** For every pair of distinct bicliques $B_1 \in \mathcal{C}$ and $B_2 \in \mathcal{C}$, if B_1 and B_2 have a consensus B_3 which is not absorbed by any member of $\mathcal{C}$, then extend B_3 to a maximal biclique B_4, and add B_4 to $\mathcal{C}$.

We call *modular consensus* the sequence of operations performed on each pair of bicliques B_1 and B_2 in the **Repeat**-loop of MCA.

Theorem 2. *Upon termination of the modular consensus algorithm, the list C contains exactly the maximal bicliques of G. The algorithm can be implemented to run in incremental polynomial time, in such a way that $\tau(k) = O\left(k^2\left(m + n\log_2 k\right)\right)$ for $k = 1, 2, \ldots, \beta$. In particular, its total running time is $O\left(\beta^2\left(m + n\log_2 \beta\right)\right) = O\left(n^2\beta^2\right)$.*

Proof. Upon termination, $\mathcal{C}$ contains a collection of maximal bicliques which cover the edges of G. Moreover, no absorption or consensus adjunction can be performed on this collection. Therefore, by Theorem 1, $\mathcal{C}$ contains exactly the maximal bicliques of G.

As above, we assume that the maximal bicliques are labelled $B_1, B_2, \ldots,$ B_β, in the order in which they are produced by the algorithm. For $k = 1, 2, \ldots, \beta$, B_k is added to the list at time $\tau(k)$. Note that at most $O(k^2)$ modular consensuses can be completed until time $\tau(k)$. Each such module requires $O(1)$ operations of consensus formation, absorption testing and extension. The consensus and extension operations can easily be performed in $O(m)$ time. Until time $\tau(k)$, absorption checking can be done in $O\left(n\log_2 k\right)$ time using binary search: this requires a data structure which maintains $\{B_1, B_2, \ldots, B_k\}$ as an ordered list throughout the algorithm, and which allows insertion to be performed efficiently (see [AHU74]; more details will be given in Section).

In conclusion we see that $\tau(k) = O\left(k^2(m + n\log_2 k)\right)$ for $k = 1, 2, \ldots, \beta$. This completes the proof, since $m = O(n^2)$ and $\log_2 \beta = O(n)$. ■

Example 3. Consider the graph $G = (V, E)$ with $V = \{1, .., 6\}$ and $E = \{(1,2), (1,6), (2,3), (3,4), (3,6), (4,5), (4,6), (5,6)\}$.

The modular consensus algorithm performs the following steps:
1. Start with the following collection of maximal bicliques that cover the arcs

of G: $\mathcal{C} = \{B_1, B_2, B_3, B_4\}$, where $B_1 = (\{1,3\}, \{2,6\})$, $B_2 = (\{3\}, \{2,4,6\})$, $B_3 = (\{4\}, \{3,5,6\})$, $B_4 = (\{3,5\}, \{4,6\})$.
2. All the consensuses of B_1 and B_2 are already contained in $\mathcal{C}$.
3. The only consensus of B_1 and B_3 not already contained in $\mathcal{C}$ is $(\{1,3,4\}, \{6\})$. The maximal biclique obtained by extending this consensus is $B_5 = (\{1,3,4,5\}, \{6\})$, and it is added to $\mathcal{C}$.
4. All the consensuses of all the other pairs of bicliques are already contained in $\mathcal{C}$.

The algorithm terminates with the collection $\widehat{\mathcal{C}} = \{B_1, B_2, B_3, B_4, B_5\}$ of all maximal bicliques of G. ■

It should be noted that the modular consensus algorithm can in fact be viewed as a specialized version of the consensus algorithm. More precisely, for $i = 1, 2, \ldots, n$, let $S(i)$ denote the *star* adjacent to vertex i, i.e. $S(i)$ is the biclique $(\{i\}, \{j : \{i,j\} \in E\})$. Let us say that a collection of maximal bicliques $\mathcal{C}$ *covers the stars* of G if each of the stars $S(1)$, $S(2)$, ..., $S(n)$ is contained in some member of $\mathcal{C}$.

It can be checked that an arbitrary biclique B can be extended to a maximal biclique by performing successive consensus steps on the list (B, $S(1)$, $S(2)$, ..., $S(n)$) or, alternatively, on any list $(B, \mathcal{C})$ such that $\mathcal{C}$ covers the stars. Therefore, the iterations of MCA can be mimicked if we initialize the consensus algorithm with the list ($S(1)$, $S(2)$, ..., $S(n)$) and if we perform its operations in the appropriate order (corresponding to the successive modular consensuses). In the next section, we are going to build on this observation to further improve the complexity of the consensus algorithm.

3.3 A better incrementally polynomial consensus algorithm

Let us now define the

Modular input consensus algorithm (*MICA*)

- **Start** with a list $\mathcal{C}_0$ of at most n maximal bicliques that cover the stars of the graph G. Let $\mathcal{C} := \mathcal{C}_0$.
- **Repeat:** For every pair of distinct bicliques $B_1 \in \mathcal{C}_0$ and $B_2 \in \mathcal{C}$, if B_1 and B_2 have a consensus B_3 which is not absorbed by any member of $\mathcal{C}$, then extend B_3 to a maximal biclique B_4, and add B_4 to $\mathcal{C}$.

Note that the difference between MCA and $MICA$ is that, in $MICA$, at least one of the two bicliques used when a consensus step is applied, belongs always to the initial list $\mathcal{C}_0$. (A similar specialization of the Boolean consensus method has been investigated in the artificial intelligence literature; see e.g. [CL73]).

Theorem 3. *Upon termination of the modular input consensus algorithm, the list C contains exactly the maximal bicliques of G. The algorithm can be implemented to run in incremental polynomial time, in such a way that $\tau(k) = O\left(nk\left(m + n\log_2 k\right)\right)$ for $k = 1, 2, \ldots, \beta$. In particular, its total running time is $O\left(n\beta\left(m + n\log_2 \beta\right)\right) = O\left(n^3\beta\right)$.*

Proof. To establish the validity of $MICA$, it is sufficient to repeat the proof of Theorem 1 with minor modifications which we explicitly spell out for further reference.

In the induction argument (last part of the proof of Theorem 1), let k denote the number of vertices in X^*, i.e. $k = |X^*|$. If $k = 1$, then (X^*, Y^*) is absorbed by a star of G, and hence (X^*, Y^*) is absorbed by an element of the initial list $\mathcal{C}_0$.

If $k \geq 2$, then X^* can be partitioned into two subsets X' and X'' such that $|X'| = 1$ and $|X''| \geq 1$. Thus, the biclique $B' = (X', X^*)$ is a star, and we can assume that it is absorbed by a maximal biclique $B_1 \in \mathcal{C}_0$. The argument can now be completed as in Theorem 1.

The complexity of the algorithm can be analyzed as in the proof of Theorem 2. Just observe that, in $MICA$, at most $O(nk)$ modular input consensus steps can be completed until time $\tau(k)$. ■

Since β is typically much larger than n, the complexity of the modular input consensus algorithm is typically better than the complexity of the modular consensus algorithm. This conclusion will be amply confirmed by the numerical experiments described in section 6.

3.4 Polynomial delay algorithms

Tsukiyama, Ide, Ariyoshi, and Shirakawa [TIAS77], and later Johnson, Yannakakis, and Papadimitriou [JYP88], have proposed different polynomial delay algorithms for generating all maximal cliques of a graph. Lawler, Lenstra, and Rinnooy Kan [LLK80] have observed that a similar algorithmic principle actually yields polynomial algorithms for a rather large variety of combinatorial generation problems (namely, for the generation of all maximal independent sets in special classes of independence systems).

We now briefly describe a transformation which allows to apply these previous results directly to the $MBGP$ problem.

Given a (loopless) graph $G = (V, E)$, with $V = \{v_1, ..., v_n\}$, we associate to G its *double cover* $2G := (L \cup R; A)$, where $2G$ is the complement of a bipartite graph. In this definition, the sets $L = \{l_1, ..., l_n\}$ and $R = \{r_1, ..., r_n\}$ are two disjoint copies of V, and the edge set A is defined as

$$\begin{aligned} A &= \{\, \{l_i, r_j\}, \{l_j, r_i\} : i, j \in \{1, ..., n\},\, \{v_i, v_j\} \in E\} \\ &\quad \cup \{\, \{l_i, l_j\}, \{r_i, r_j\} : i, j \in \{1, ..., n\}, i < j \,\}. \end{aligned}$$

Remark that there is a one-to-one correspondence between the bicliques of G and the symmetric pairs of cliques of $2G$. More precisely, consider a biclique (X, Y) of G and denote the copies of the sets X and Y in L and R by L_X, L_Y, R_X, R_Y, respectively. Then, it is easy to see that there is a one-to-one correspondence between the biclique (X, Y) and the symmetric pair of cliques $((L_X, R_Y), (L_Y, R_X))$. This observation allows to reduce in $O(n^2)$ time the $MBGP$ for the general graph G to the problem of generating all maximal cliques of its double cover $2G$.

Now, the algorithms in [TIAS77, JYP88] can be used to generate all maximal cliques of $2G$ with delay $O(n(n^2 - m)) = O(n^3)$. Therefore, the total running time of these algorithms is $O(n^3\beta)$. It is interesting to note that this is the same total complexity as for the modular input consensus algorithm, although the clique generation algorithms appear conceptually very different from the consensus algorithms. A brief comparison of some computational experiments carried out with these algorithms will be presented in section 6.

4 Some classes of bigraphs for which $MBGP$ can be solved polynomially

While the number of maximal bicliques of a graph can be exponential in its size, some classes of specially structured graphs have only a polynomial number of maximal bicliques, and therefore, $MBGP$ can be solved in polynomial time for them. In this section, we present two special classes of graphs for which $MICA$ works in polynomial time.

It should be noted that specialized algorithms – based or not based on a consensus type approach – which exploit the specific nature of particular classes of graphs can lead to highly efficient solutions of the $MBGP$ for these classes of graphs. As an example we mention the linear time algorithm of Eppstein [Epp94] for the class of graphs with bounded arboricity, which includes in particular the class of bounded degree graphs discussed below.

The two examples described below concern classes of graphs having polynomially many maximal bicliques. It is clear that for such classes $MICA$ performs polynomially, even without any specific adaptation.

Graphs with bounded degree. Let d be a fixed integer and let us consider the family of all graphs whose node degrees are bounded by d. For each graph G in this class, $\beta(G) \leq n2^d$. Indeed, let us evaluate the number of maximal bicliques of G which contain some fixed vertex v. Consider such a maximal biclique, say (X, Y), with $v \in Y$. Since v has at most d neighbors, it follows that it belongs to at most 2^d maximal bicliques. Therefore, the total number of maximal bicliques of G is at most $n2^d$.

Convex bipartite graphs. Let $G = (V, E)$ be a bipartite graph and let (L, R) be the bipartition of its vertex set, with $L = \{l_1, ..., l_{n_1}\}$, $R = \{r_1, ..., r_{n_2}\}$. A bipartite graph G is called $R - convex$ (see [Glo67]) if there exists a permutation π of the vertices in R such that the neighborhood of every vertex $l \in L$ is an "interval " of the form $\{r_{\pi(i)}, r_{\pi(i+1)}, ..., r_{\pi(j)}\}$ in R. The number $\beta(G)$ of maximal bicliques of an $R-$convex graph is $O(n_2^2)$. Indeed, a maximal biclique (X, Y) of G is uniquely determined by the intersection of its vertex set with $\pi(R)$. Since $Y \cap \pi(R)$ is an interval, and the number of different intervals in $\pi(R)$ is $O(n_2^2)$, the remark follows.

5 Implementations

Substantial improvements in the practical running time of MCA and $MICA$ can be obtained by noticing that many of the consensus adjunction operations produce the same outputs, and are therefore redundant. Without going into tedious details, we are now going to present some of the main ideas that have been used in our algorithmic implementations, and which appear to be responsible for most of the observed efficiency improvements.

Let us start with some observations. With any subset X of the vertex set V of a graph G, we can associate another set of vertices denoted $\Gamma(X)$, containing all those vertices which are adjacent to every vertex in X. We also let $\Gamma^2(X) = \Gamma(\Gamma(X))$. When $\Gamma(X) \neq \emptyset$, it is easy to see that $(\Gamma^2(X), \Gamma(X))$ is a maximal biclique of G, which we call *the biclique generated by* $\Gamma(X)$. Moreover, every maximal biclique can be written in the form $(\Gamma^2(X), \Gamma(X))$ for some (not necessarily unique) set X. In our implementations, we encode a maximal biclique $(\Gamma^2(X), \Gamma(X))$ by recording the set $\Gamma(X)$ only, since this set suffices to determine completely the maximal biclique. When necessary, we recompute $\Gamma^2(X)$ from $\Gamma(X)$.

In $MICA$, the initial list $\mathcal{C}_0$ is always taken to be (up to absorptions) the list of generating sets $\Gamma(\{v_i\})$ with $v_i \in V$ ($i = 1, 2, ..., n$). In the

course of the algorithm, we maintain a list $\mathcal{C}$ of generating sets sorted in lexicographic order, which allows us to perform efficiently all insertion and absorption checking operations (as mentioned in the proof of Theorem 2).

Our implementation of $MICA$ performs successive *stages* consisting of consensus and absorption testing operations. Stage $k+1$ starts with the list W_k of maximal bicliques produced at stage k (for the first stage, W_k is a copy of the initial list $\mathcal{C}_0$). Then, the algorithm produces the consensus of each maximal biclique in W_k with each maximal bicliques in $\mathcal{C}_0$. (As a matter of fact, an accelerating ingredient of the procedure consists in avoiding the examination of the consensuses of a biclique B in W_k with a star S from C_0, in those cases when B itself is the result of consensus operations which made use at some previous stage of the star S). If the resulting consensus is not absorbed by any member of W_k, it is extended to a maximal biclique and added to the new list W_{k+1}. When this step is over, W_{k+1} is merged into the sorted list $\mathcal{C}$ (checking again for possible absorptions).

A further acceleration of $MICA$ is obtained using a technical observation which makes it possible when computing the consensus of two maximal bicliques (one from $\mathcal{C}$ and one from $\mathcal{C}_0$) to consider only one of the (up to 4 possible) consensuses.

6 Computational results

A large number of computational tests were run on uniformly generated random graphs with 25 to 2000 vertices, and with 30 to 19990 edges. The algorithms were implemented in Delphi v5. All tests were run on a Pentium III 550MHz PC.

Since we are not aware of any other algorithm designed to solve $MBGP$, in order to be able to compare the performance of the proposed algorithms with other methods, we have adapted for this purpose the lexicographic clique generation algorithm of [JYP88], as described in section 3.4; this algorithm will be called LEX in the remainder of the section.

We shall describe below the results of computational experiments using three algorithms: *MCA, MICA,* and *LEX.*

A first purpose of the experiments was to perform an empirical comparison of the relative efficiency of MCA, $MICA$, and LEX. The results of this comparison are displayed in Table 1. Each row of the table corresponds to a particular combination of parameters (n, m), and displays average running times (in seconds) over samples of 5 to 10 graphs with n vertices and m edges. An asterisk in the table indicates that the corresponding algorithm ran out of memory before termination.

The results in Table 1 establish that, for our implementations of the algorithms, and for the random graphs tested, MCA performs worse than LEX, which is itself outperformed by $MICA$. In particular, an important conclusion of this study is the ability of $MICA$ to handle very large graphs under limitations of computational time. We should also mention that, for each given graph size, the variance of the computing times is rather small.

It is also interesting to observe that the total number of maximal bicliques β grows quite rapidly with the increase of density and number of vertices.

n	*m*	Density (%)	*Beta*	Time (s)		
				LEX	*MCA*	*MICA*
25	30	10	19	0.05	0.05	0.00
25	60	20	50	0.05	0.20	0.00
25	90	30	121	0.17	1.13	0.05
25	120	40	275	0.33	6.37	0.11
25	150	50	658	0.72	69.05	0.28
25	180	60	1791	1.65	535.76	0.71
25	210	70	4905	5.71	1079.02	2.58
25	240	80	15057	16.92	*	12.14
40	78	10	53	0.22	0.16	0.00
40	156	20	199	0.66	4.61	0.06
40	234	30	628	2.20	37.79	0.49
40	312	40	2182	7.36	887.38	1.71
40	390	50	8544	23.29	*	15.33
40	468	60	33334	475.56	*	47.12
50	123	10	88	0.55	0.45	0.06
50	245	20	409	3.02	18.49	0.38
50	368	30	1637	8.85	293.74	1.70
50	490	40	6769	35.48	*	9.12
50	613	50	31591	174.50	*	56.41
50	735	60	138720	2403.92	*	904.51
75	278	10	251	5.54	8.46	0.33
75	555	20	1834	28.28	424.73	3.24
75	833	30	10068	148.13	*	24.44
100	495	10	610	23.07	80.52	0.99
100	990	20	5690	172.68	*	18.79
100	1,485	30	41604	1252.79	*	179.05
150	1,118	10	2484	75.32	*	10.00
150	2,235	20	33003	2954.71	*	275.84
200	1,990	10	6845	1516.82	*	47.56
500	2,495	2	1502	279.58	*	28.72
1,000	9,990	2	13779	*	*	920.71
2,000	19,990	1	17755	*	*	4163.08

Table 1

From the data in Table 1, we have tried to predict the empirical running time of $MICA$ as a function of the parameters n, m and β. Using least square regression (and omitting the largest outlier observations), it seems that this empirical running time is reasonably well estimated (with $R^2 = 0.98$) by a function of the form $h(n) = c\, n\, \beta \log_2 n \log_2 \beta$, where the constant c takes a value around 5×10^{-7}.The values estimated by the regression model are reported in Table 2, next to the real observed times. Of course,

the conclusion of this analysis must be interpreted carefully, as we did not perform any large scale experimental study of the running time. But it seems nevertheless interesting that, for the range of graphs tested, the observed running time $h(n)$ is smaller than the theoretical upper bound derived in Theorem 3.

n	Density (%)	*MICA* Time (s)	
		Predicted	Real
25	10	0.00	0.00
25	20	0.02	0.00
25	30	0.05	0.05
25	40	0.12	0.11
25	50	0.34	0.28
25	60	1.06	0.71
25	70	3.28	2.58
25	80	11.40	12.14
40	10	0.03	0.00
40	20	0.15	0.06
40	30	0.58	0.49
40	40	2.42	1.71
40	50	11.16	15.33
40	60	50.09	47.12
50	10	0.08	0.06
50	20	0.47	0.38
50	30	2.32	1.70
50	40	11.42	9.12
50	50	62.60	56.41
75	10	0.44	0.33
75	20	4.36	3.24
75	30	29.38	24.44
100	10	1.76	0.99
100	20	22.15	18.79
100	30	199.26	179.05
150	10	14.27	10.00
150	20	252.35	275.84
200	10	62.64	47.56
500	2	33.38	28.72
Estimated complexity coefficient		0.00000047	

Table 2

Finally, we have also made an attempt to determine whether the distinction between the theoretical complexities of $MICA$ and LEX (incremental polynomial vs. polynomial delay) would be empirically observable. We have run experiments on graphs with 40 vertices and density 40%. Figure 1 displays on the vertical axis the cumulative running time $T(k)$ of each algorithm until it outputs the k-th maximal biclique (where k is mapped on the horizontal axis). Thus, $T(k)$ is an empirical analog of the quantity $\tau(k)$ defined in section 3. In an ideal case $T(k)$ would be a linear function of k (meaning that the time spent between successive outputs would be constant). It can be seen in Figure 1 that $MICA'$ s performance is close to the desired objective.

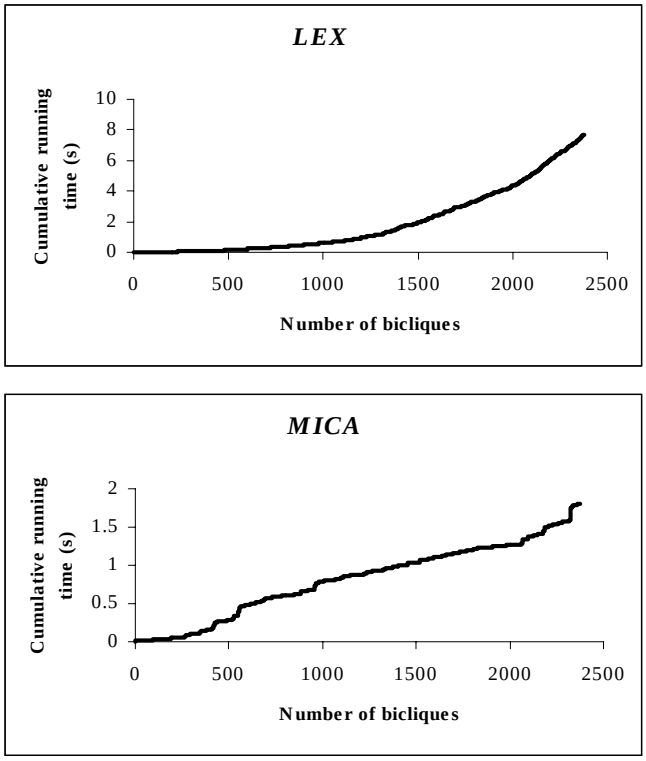

Figure 1

7 Conclusions

We have presented a new approach to the problem of generating all maximal bicliques of a graph. This approach appears interesting, in that it departs radically from previous techniques for related problems, such as the maximal clique generation problem. Moreover, certain variants of the method are provably efficient, i.e. incrementally polynomial.

Our preliminary computational experiments indicate that consensus algorithms in general, and $MICA$ in particular, constitute an efficient tool for the generation of all maximal bicliques of a graph. More extensive computational experiments may be necessary to understand more precisely the behavior of $MICA$ on large size instances and on a broader variety of graph distributions, as well as its relative performance with respect to other possible algorithmic approaches.

References

[AAAS93] Agarwal, P.; Alon, N.; Aronov, B.; Suri, S. Can visibility graphs be represented compactly*?* In *Proc. 9th ACM Symp. Computational Geometry,* (1993), 338-347.

[AHU74] Aho, A.V.; Hopcroft, J.E.; Ullman, J.D. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, Reading etc., 1974.

[BBHS83] Benzaken, Cl.; Boyd, S.; Hammer, P. L.; Simeone, B. Adjoints of pure bidirected graphs*. Congressus Numerantium* 39 (1983), 123-144.

[BHS80] Benzaken, Cl.; Hammer, P.L.; Simeone, B. Some remarks on conflict graphs of quadratic pseudo-Boolean functions. *Internat. Ser. Numer. Mathematics Birkhäuser, Basel-Boston, Mass.*, 55 (1980), 9–30.

[Ber78] Bermond, J.-C. *Couvertures des arrêtes d'un graphe par des graphes bipartis complets.* Preprint, Univ. De Paris Sud, Centre d'Orsay, Rapport de Recherche No. 10 (June 1978).

[Bla37] Blake, A. *Canonical expressions in Boolean algebra.* Ph.D. Thesis*,* University of Chicago, 1937.

[BCH90] Boros, E.; Crama, Y.; Hammer, P.L. Polynomial-time inference of all valid implications for Horn and related formulae. *Annals of Mathematics and Artificial Intelligence* 1 (1990) 21–32.

[BIK99] Bylka, S.; Idzik, A.; Komar, J. Bipartite subgraphs of graphs with maximum degree three. *Graphs and Combinatorics* 15 (1999), 129–136.

[CL73] Chang, C.L.; Lee, R.C. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York - San Francisco - London, 1973.

[Chu80] Chung, F.R.K. On the coverings of graphs. *Discrete Mathematics* 30 (1980), no. 2, 89–93.

[Chu81] Chung, F.R.K. On the decomposition of graphs. *SIAM Journal on Algebraic Discrete Methods* 2 (1981), no. 1, 1–12.

[CES83] Chung, F.R.K.; Erdös, P.; Spencer, J. On the decomposition of graphs into complete bipartite subgraphs. *Studies in Pure Mathematics*, Birkhäuser, (1983), 95–101.

[CH89] Crama, Y.; Hammer, P.L. Recognition of quadratic graphs and adjoints of bidirected graphs. *Annals of the New York Academy of Sciences, New York*, 555 (1989), 140–149.

[DKT97] Dawande, M.; Keskinocak, P.; Tayur, S. On the complexity of the biclique problem in bipartite graphs. *GSIA Working Paper* 1996-04, *Carnegie-Mellon University*, 1997.

[DLS99] Doherty, F.C.C.; Lundgren, J.R.; Siewert, D.J. Biclique covers and partitions of bipartite graphs and digraphs and related matrix ranks of {0,1}-matrices. *Congressus Numerantium* 136 (1999), 73-96.

[DM58] Dulmage, A.L.; Mendelsohn, N.S. Coverings of bipartite graphs. *Canadian Journal of Mathematics* 10 (1958), 517-534.

[EHW84] Ebenegger, Ch.; Hammer, P.L.; de Werra, D. Pseudo-Boolean functions and stability of graphs. *Annals of Discrete Mathematics* 19 (1984), 83-97.

[Epp94] Eppstein, D. Arboricity and bipartite subgraph listing algorithms. *Information Processing Letters* 51 (1994) no. 4, 207-211.

[FH00] Foldes, S.; Hammer, P.L. Disjunctive and conjunctive normal forms of pseudo-Boolean functions. *Discrete Applied Mathematics 107 (2000),* no. *1-3, 1-26.*

[Glo67] Glover, F. Maximum matching in a convex bipartite graph. *Naval Research Logistics Quarterly* 14 (1967), 313-316.

[GH88] Golumbic, M.C.; Hammer, P.L. Stability in circular arc graphs. *Journal of Algorithms* 9 (1988), no. 3, 314–320.

[Ham78] Hammer, P.L. The conflict graph of a pseudo-Boolean function. *Technical Report, Bell Laboratories, West Long Branch*, NJ, 1978.

[HS79] Hammer, P.L.; Simeone, B. Quasimonotone Boolean functions and bistellar graphs. *Annals of Discrete Mathematics* 9 (1980), 107–119.

[HMW85_1] Hammer, P.L.; Mahadev, N.V.R.; de Werra, D. The struction of a graph: application to CN-free graphs. *Combinatorica* 5 (1985), no. 2, 141–147.

[HMW85_2] Hammer, P.L.; Mahadev, N.V.R.; de Werra, D. Stability in CAN-free graphs. *Journal of Combinatorial Theory Ser. B* 38 (1985), no. 1, 23–30.

[Hoc98] Hochbaum, D.S. Approximating clique and biclique problems. *Journal of Algorithms* 29 (1998), no. 1, 174–200.

[JYP88] Johnson, D.S.; Yannakakis, M.; Papadimitriou, C.H. On generating all maximal independent sets. *Information Processing Letters* 27 (1988), no. 3, 119–123.

[KP77] Kaufmann, A.; Pichat, E. *Méthodes Mathématiques Non Numériques et leurs Algorithmes*, *Tome I, Algorithmes de Recherche des Eléments Maximaux,* Masson, Paris, 1977.

[LLK80] Lawler, E.L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. Generating all maximal independent sets: NP-hardness and polynomial-time algorithms. *SIAM Journal on Computing* 9 (1980), no. 3, 558–565.

[Mal62] Malgrange, Y. *Recherche des sous-matrices premières d'une matrice à coefficients binaires. Applications à certains problèmes de graphe.* In Deuxième Congrès de l'AFCALTI, October 1961, Gauthier-Villars, 1962, 231-242.

[Pic77] Pichat, E. The disangagement algorithm or a new generalization of the exclusion algorithm. *Discrete Mathematics* 17 (1977), no. 1, 95-106.

[PU59] Paull, M.C.; Unger, S.H. Minimizing the number of states in incompletely specified sequential switching functions. *IRE Transactions on Electronic Computers,* EC-8 (1959), 356-367.

[Qui55] Quine, W. A way to simplify truth functions. *American Mathematical Monthly*, 62, (1955), 627-631.

[TIAS77] Tsukiyama, S.; Ide, M.; Ariyoshi, H.; Shirakawa, I. A new algorithm for generating all the maximal independent sets. *SIAM Journal on Computing* 6 (1977), no. 3, 505–517.

[Tuz83] Tuza, Z.: Covering of graphs by complete bipartite subgraphs: complexity of 0-1 matrices. *Combinatorica* 4 (1984), no.1, 111-116.