On Independent Sets and Bicliques in Graphs*

Serge Gaspers[1], Dieter Kratsch[2], and Mathieu Liedloff[2]

[1] Department of Informatics, University of Bergen, N-5020 Bergen, Norway. Email: serge@ii.uib.no

[2] Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France. E-mail: (kratsch|liedloff)@univ-metz.fr

Abstract. Bicliques of graphs have been studied extensively, partially motivated by the large number of applications. One of the main algorithmic interests is in designing algorithms to enumerate all maximal bicliques of a (bipartite) graph. Polynomial-time reductions have been used explicitly or implicitly to design polynomial delay algorithms to enumerate all maximal bicliques.

Based on polynomial-time Turing reductions, various algorithmic problems on (maximal) bicliques can be studied by considering the related problem for (maximal) independent sets. In this line of research, we improve Prisner's upper bound on the number of maximal bicliques [Combinatorica, 2000] and show that the maximum number of maximal bicliques in a graph on n vertices is exactly $3^{n/3}$ (up to a polynomial factor). The main results of this paper are $O(1.3642^n)$ time algorithms to compute the number of maximal independent sets and maximal bicliques in a graph.

1 Introduction

Bicliques. Let the vertex sets X and Y be independent sets of a graph $G = (V, E)$ such that $xy \in E$ for all $x \in X$ and all $y \in Y$. The subgraph of G induced by $X \cup Y$ is called a *biclique* of G. Furthermore depending on the context and the application area, one also calls the pair (X, Y) or the vertex set $X \cup Y$ a biclique. From a graph-theoretic point of view it is natural to consider a biclique of a graph G as a complete bipartite induced subgraph of G. For technical reasons, we prefer to consider a biclique $B \subseteq V$ of a graph $G = (V, E)$ as a vertex set inducing a complete bipartite subgraph of G.

Maximal bicliques. A biclique $B \subseteq V$ of G is a *maximal biclique* of G if B is not properly contained in another biclique of G. A lot of the research on maximal bicliques and in particular on algorithms to enumerate all maximal bicliques of (bipartite) graphs with polynomial delay is motivated by the various applications of bicliques in (bipartite) graphs. Applications of bicliques in automata and language theory, graph compression, artificial intelligence and biology are discussed in [2]. An important application in data mining is based on the formal

* A large part of the research was done while Serge Gaspers was visiting the University of Metz.

concept analysis [10] where each concept is a maximal biclique of a bipartite graph.

Previous work. The complexity of algorithmic problems on bicliques has been studied extensively. First results were mentioned by Garey and Johnson [9], among them the NP-completeness of the balanced complete bipartite subgraph problem. The maximum biclique problem is polynomial for bipartite graphs [3], and NP-hard for general graphs [25]. The maximum edge biclique problem was shown to be NP-hard by Peeters [20].

Approximation algorithms for node and edge deletion biclique problems are given by Hochbaum [12]. Enumerating maximal bicliques has attracted a lot of attention in the last decade. The algorithms in [17, 18] enumerate all maximal bicliques of a bipartite graph as concepts during the construction of the concept lattice. Nowadays there are polynomial delay enumeration algorithms for maximal bicliques in bipartite graphs [5, 15] and general graphs [4, 15]. There are also polynomial delay algorithms to enumerate all maximal non-induced bicliques of a graph [1, 5].[3]

Prisner studied various aspects of bicliques in graphs. Among others, he showed that the maximum number of maximal bicliques in a bipartite graph on n vertices is $2^{n/2}$. He established a lower bound of $3^{n/3}$ and an upper bound of 1.6181^n (up to a polynomial factor) on the maximum number of maximal bicliques in a graph on n vertices [21].

Our Results. We use a simple polynomial-time Turing reduction to transform results on maximal independent sets into results on maximal bicliques. We also improve upon Prisner's upper bound and give a simple proof that the maximum number of maximal bicliques in a graph on n vertices is at most $n \cdot 3^{n/3}$. Our main result is a $O(1.3642^n)$ time algorithm to count all maximal independent sets in a graph, which is established by using the Measure & Conquer technique (see e.g. [7]). No such algorithm was known prior to our work. We show how to use it to count all maximal bicliques of a graph within the same time bound and also provide a lower bound for the running time of this algorithm.

2 Preliminaries

All graphs in this paper are simple and undirected. For a graph $G = (V, E)$, we let $n = |V|$ and $m = |E|$. An edge between vertices u and v is denoted by uv. The set of *neighbors* of a vertex $v \in V$ is the set of all vertices adjacent to v, denoted by $N(v)$. The *closed neighborhood* of a vertex v is $N[v] = \{v\} \cup N(v)$. The *distance* between two vertices u, v is the length of the shortest path from u to v. We denote by $N^k(v)$ the set of all vertices at distance k from v, and by $N^k[v]$ the set of all vertices at distance at most k from v. The *degree* of a vertex v is $d(v) = |N(v)|$. A *clique* is a set of vertices that are all pairwise adjacent,

[3] When the condition that X and Y are independent sets in the definition of a biclique is dropped, then (X, Y) is called a *non-induced biclique* of G. In this case a different maximality notion is used. See e.g. [1].

and an *independent set* is a set of vertices that are all pairwise non-adjacent. An independent set is *maximal* if it is not properly contained in another independent set. The subgraph of G induced by a vertex set $A \subseteq V$ is denoted by $G[A]$. A graph is called *bipartite* if its vertex set can be partitioned into two independent sets V and W. The *bipartite complement* of a bipartite graph $G = (V, W, E)$ is a bipartite graph having the vertices of G as its vertex set and the non-edges of G with an endpoint in V and another in W as its edge set.

3 Polynomial-Time Reductions

There is a natural relation between independent sets (and cliques) on one hand and bicliques on the other hand. Thus it is not surprising that polynomial-time Turing reductions (in fact mainly Karp reductions) have been used in various hardness proofs for problems on bicliques [9]. The famous polynomial delay algorithm of Johnson and Papadimitriou to enumerate all maximal independent sets [14] is used explicitly or implicitly in polynomial delay algorithms to enumerate maximal (non-induced) bicliques in (bipartite) graphs [1, 4, 5].

The first reduction simply recalls an often used argument.

Lemma 1 (Property A). *Let $G = (V, E)$ be a bipartite graph. Let H be the bipartite complement of G. Then B is a (maximal) biclique of G if and only if B is a (maximal) independent set of H.*

The above lemma implies, among others, that any algorithm enumerating all maximal independent sets within delay $f(n)$ can be transformed into an algorithm enumerating all maximal bicliques of a bipartite graph within delay $f(n)$. The known tight bound of $2^{n/2}$ for the maximum number of maximal bicliques in a bipartite graph given in [21] follows easily from Property A and the corresponding bound for maximal independent sets in [13]. Based on this property, Yannakakis observed that the problem of finding a maximum biclique in a bipartite graph is solvable in polynomial time [25].

The following property is central for our paper.

Lemma 2 (Property B). *Let $G = (V, E)$ be a graph. For every $v \in V$, the graph H_v is the graph with vertex set $V(H_v) = N(v) \cup N^2(v)$. Its edge set $E(H_v)$ consists of the following edges:*

- *$xy \in E(H_v)$ if $xy \in E$ and $x, y \in N(v)$,*
- *$xy \in E(H_v)$ if $xy \in E$ and $x, y \in N^2(v)$,*
- *$xy \in E(H_v)$ if $xy \notin E$, $x \in N(v)$ and $y \in N^2(v)$.*

Then $B \subseteq V$ is a (maximal) biclique of G if and only if $B - v$ is a (maximal) independent set of a graph H_v for some $v \in B$.

Proof. Let B be a (maximal) biclique of G. Take some $v \in B$. Then $B \subseteq \{v\} \cup N(v) \cup N^2(v)$ in G, where the independent sets X and Y of the biclique B satisfy $X \subseteq N(v)$ and $Y \subseteq \{v\} \cup N^2(v)$. Since B is a biclique and by the

construction of H_v, we obtain that $B - v$ is an independent set of H_v. On the other hand, if B' is a (maximal) independent set of H_v, for some $v \in V$, then $B' \cap N(v)$ is an independent set of $G[N(v)]$ and $B' \cap N^2(v)$ is an independent set of $G[N^2(v)]$. Hence B' is a biclique of $G - v$ and $B' \cup \{v\}$ is a biclique of G.

Finally, due to the correspondence between bicliques and independent sets, this also holds for maximality by inclusion of vertices. □

The corresponding Turing reduction does not increase the number of vertices, since $|V(H_v)| \leq |V| - 1$. Thus this reduction is useful for exponential-time algorithms.

Corollary 1. *Given an algorithm to find a maximum independent set (respectively to count all independent sets of size k) of a graph in time $c^n n^{O(1)}$, there exists an algorithm to find a maximum biclique (respectively to count all bicliques of size k) of a graph in time $c^n n^{O(1)}$.*

Proof. To find a maximum biclique of a graph $G = (V, E)$, compute a maximum independent set for each H_v, $v \in V$, constructed according to Property B and return the largest set of vertices found. To count all bicliques of size k of a graph $G = (V, E)$ on n vertices, order the vertices of G: $V = \{v_1, v_2, \ldots, v_n\}$. For $i = 1, \ldots, n$, compute the number of independent sets of size $k - 1$ of $H^i_{v_i}$ where $H^i_{v_i}$ is obtained from $G^i = G[V \setminus \{v_1, v_2, \ldots, v_{i-1}\}]$ using Property B. Adding up the results gives the number of bicliques of size k of G. □

By this corollary and the algorithms in [22, 24], a maximum biclique of a graph can be found in time $O(1.2109^n)$ and all maximum bicliques of a graph can be counted in time $O(1.2377^n)$.

Note that Corollary 1 is not directly applicable to use an algorithm for counting maximal independent sets to count the maximal bicliques of a graph. The issues are that double-counting has to be avoided at the same time as the maximality of each counted biclique has to be ensured.

4 Improving Prisner's Bound

The maximum number of maximal bicliques in a graph on n vertices has been studied by Prisner [21]. He settled the question for bipartite graphs. The maximum number of maximal bicliques in a bipartite graph on n vertices is precisely $2^{n/2}$. For general graphs the question remained open. He established a lower bound of $3^{n/3}$ and an upper bound of $(1.618034^n + o(1)) \cdot n^{5/2}$ for the maximum number of maximal bicliques in a graph on n vertices. We settle the question via an elegant proof based on Property B.

Theorem 1. *The maximum number of maximal bicliques in a graph is at most $n \cdot 3^{n/3}$.*

Proof. Let n be a positive integer and let G be any graph on n vertices. Applying Property B, for every vertex $v \in V$, there is a one-to-one correspondence between

the maximal bicliques B of G satisfying $v \in B$ and the maximal independent sets $B - v$ of the graph H_v. By a well-known theorem of Moon and Moser [16], the maximum number of maximal independent sets in a graph on n vertices is $3^{n/3}$. Thus the number of maximal bicliques containing vertex v is at most $3^{n/3}$ for each $v \in V$. Consequently G has at most $n \cdot 3^{n/3}$ maximal bicliques. □

Corollary 2. *The maximum number of maximal bicliques in a graph is* $3^{n/3}$ *(up to a polynomial factor).*

5 Counting Algorithms

A problem related to enumerating all maximal bicliques of a graph is to compute the number of maximal bicliques of a graph; faster than by simply enumerating all of them. By property B, an algorithm to count all maximal independent sets of a graph could be a cornerstone to design such an algorithm. However no non-trivial algorithm for counting maximal independent sets is known. It is known that the counting problem for maximal independent sets is #P-complete even when restricted to chordal graphs [19]. Hence our goal is to construct a fast exponential-time algorithm solving this problem.

5.1 Algorithm to Count All Maximal Independent Sets

Let $G = (F, M, E)$ be a *marked graph* which are the graphs dealt with by our algorithm. Vertices of F are called *free* and vertices of M are called *marked*. Let u be a vertex of $F \cup M$. The degree of u is the number of neighbors in $F \cup M$ and is denoted by $d(u)$. Given a set $D \subseteq (F \cup M)$, the set $N(u) \cap D$ is denoted by $N_D(u)$ and its cardinality is denoted by $d_D(u)$.

The following notions are crucial for our algorithm. A set $S \subseteq F$ is a maximal independent set (or shortly, MIS) of a marked graph $G = (F, M, E)$ if S is a MIS of $G[F]$. We say that the MIS S of $G = (F, M, E)$ satisfies property Π if each vertex of M has a neighbor in S.

Given a marked graph G, our algorithm computes the number of MISs of $G = (F, M, E)$ satisfying Π. Thus, a marked vertex u is used to force that each MIS S of G counted by the algorithm contains at least one free neighbor of u. This is particularly useful to guarantee that only maximal independent sets of the input graph are counted. In the remainder of this section, we suppose that G is a connected graph, otherwise the algorithm is called for each of its connected components, and the product of the results gives the number of MISs of G satisfying Π.

Given a simple graph $G' = (V, E)$, `#MaximalIS`$\big(G = (V, \emptyset, E)\big)$ returns the number of maximal independent sets of G'. (See the next page for the description of the algorithm.)

We emphasize that all the halting ((H1)–(H2)) and reduction ((R1)–(R7)) rules are necessary for our running time analysis (see Subsection 5.3). The branching rule (B) selects a vertex u, orders its free neighbors in a list $[v_1, v_2, \ldots,$

Algorithm `#MaximalIS`$\big(G = (F, M, E)\big)$
Input: A marked graph $G = (F, M, E)$.
Output: The number of MISs of G satisfying Π.

`// Reduction rules`
if *G is empty* **then**
  return 1 (H1)
if *there exists $u \in M$ s.t. $d_F(u) = 0$* **then**
  return 0 (H2)
if *there exists $u \in M$ s.t. $N_F(u) = \{v\}$* **then**
  return `#MaximalIS`$\big(G = (F \setminus N[v], M \setminus N(v), E)\big)$ (R1)
if *there exists $u \in F$ s.t. $d_F(u) = 0$* **then**
  return `#MaximalIS`$\big(G = (F \setminus N[u], M \setminus N(u), E)\big)$ (R2)
if *there exists $u, v \in M$ s.t. $\{u, v\} \in E$* **then**
  return `#MaximalIS`$\big(G = (F, M, E \setminus \{u, v\})\big)$ (R3)
if *there exists $u, v \in F$ s.t. $N[u] = N[v]$* **then**
  count $\leftarrow$ `#MaximalIS`$\big(G = (F \setminus \{v\}, M, E)\big)$
  Let MIS_u be the number of MISs computed by `#MaximalIS`$\big(G = (F \setminus \{v\}, M, E)\big)$ containing u
  return MIS_u + *count* (R4)
if *there exists $u \in M$ and $v \in N(u)$ s.t. $N[v] \subseteq N[u]$* **then**
  return `#MaximalIS`$\big(G = (F, M \setminus \{u\}, E)\big)$ (R5)
if *there exists $u, v \in M$ s.t. $N(u) = N(v)$* **then**
  return `#MaximalIS`$\big(G = (F, M \setminus \{v\}, E)\big)$ (R6)
if *there exists $u \in F \cup M$ and $v \in F$ s.t. $N(u) = N(v)$* **then**
  return `#MaximalIS`$\big(G = (F \setminus \{v\}, M, E)\big)$ (R7)
`// Branching rule` (B)
if *there exists a marked vertex u with $d(u) = 2$* **then**
  Choose u
else
  Choose a vertex $u \in (F \cup M)$ such that
    (i) u has minimum degree among all vertices in $F \cup M$
    (ii) among all vertices fulfilling (i), u has a neighbor of maximum degree
    (iii) among all vertices fulfilling (ii), u has maximum dual degree (i.e. the sum of the degrees of its neighbors)
Let `BL`$(u) \leftarrow [v_1, \ldots, v_{d_F(u)}]$ be an ordered list of $N_F(u)$ such that:
  (i) v_1 is a vertex of $N_F(u)$ having a minimum number of neighbors in $V \setminus N(u)$
  (ii) append (in any order) the vertices of $N(v_1) \cap N_F(u)$ to the ordered list
  (iii) append the vertices of $N_F(u) \setminus N[v_1]$ ordered by increasing number of neighbors in $V \setminus N(u)$
count $\leftarrow 0$
if *u is free* **then** `// select` u `(to be in the current MIS)`
  count $\leftarrow$ `#MaximalIS`$\big(G = (F \setminus N[u], M \setminus N(u), E)\big)$
foreach $v_i \in$ `BL`(u) **do** `// mark each vertex of` M' `and select` v_i
  $M' \leftarrow \{v_j \in \texttt{BL}(u) : 1 \leq j < i \text{ and } \{v_j, v_i\} \notin E\}$
  count $\leftarrow$ *count*+ `#MaximalIS`$\big(G = (F \setminus (M' \cup N[v_i]), (M \cup M') \setminus N(v_i), E)\big)$
return *count*

$v_{d_F(u)}]$ and makes a recursive call (i.e. a branching) counting all MISs containing u, and a recursive call for each $i = 1, 2, \ldots, d_F(u)$ where it counts all MISs containing v_i but none of $v_1, v_2, \ldots, v_{i-1}$.

The selected vertex u is chosen according to three criteria (i)–(iii). By (i), u has minimum degree, which ensures either that the algorithm makes few recursive calls or that many vertices are removed in each branching. By (ii), u has a neighbor of maximum degree among all vertices satisfying (i). If the degree of this neighbor is high, then many vertices are removed in at least one recursive call. If the degree of this vertex is low, every vertex of minimum degree has no high-degree neighbor. This property is exploited in the analysis of our algorithm, which considers a decrease in the degree of a vertex of small degree more advantageous than a decrease in the degree of a high-degree vertex. Similarly, (iii) ensures either many recursive calls where many vertices are removed or a knowledge on the degrees of the neighbors of a vertex of minimum degree. The ordered list $\texttt{BL}(u)$ is defined in this way to ensure that for certain configurations of $N^2[u]$, reduction rule (R1) or a (fast) subsequent branching on a marked vertex of degree 2 is applied in many recursive calls.

5.2 Correctness of `#MaximalIS`

We show the correctness of the branching and reduction rules of `#MaximalIS`. **(H1)** If the input graph is empty then the only MIS is the empty set. **(H2)** If there is a marked vertex u without any free neighbor then there is no MIS satisfying Π. **(R1)** If a marked vertex u has only one free neighbor v, the vertex v has to be in the MIS to satisfy Π. **(R2)** By maximality, each free vertex without any free neighbor has to belong to all MISs. **(R3)** Since marked vertices cannot belong to any MIS, edges between two marked vertices are irrelevant and can be removed. **(R4)** Suppose $u, v \in F$ are two free vertices and $N[u] = N[v]$. Every MIS containing a neighbor of u does not contain v. Moreover, every MIS containing u can be replaced by one containing v instead of u. Thus, it is sufficient to remove v and to return the number of MISs containing a neighbor of u plus twice the number of MISs containing u. (Note that the algorithm can easily be implemented such that the number of MISs containing u is obtained from the recursive call. E.g., keep a counter to associate to each free vertex the number of MISs containing this vertex.) **(R5)** If $u \in M$ has a neighbor v such that all neighbors of v are also neighbors of u, then every MIS of $G - u$ must contain a vertex of $N[v] \setminus \{u\}$ and thus a neighbor of u in G. **(R6)** If two marked vertices have the same neighborhood then one of them is irrelevant. **(R7)** Let v be a free vertex and u a vertex such that $N(u) = N(v)$, and thus u and v are non adjacent. Hence every MIS containing a neighbor of u does not contain v and every MIS containing u (if u is free) also contains v. Thus the number of MISs is the same as for $G - v$.

(B) The algorithm considers the two possibilities that either u or at least one neighbor of u is in the current MIS. By induction and the fact that $N[u]$ is removed if the algorithm decides to add u to the current MIS, every MIS containing u is counted and it is counted only once. Consider the possibility

that at least one neighbor of u is in the current MIS and let v_i be the first such neighbor in the ordered list $\texttt{BL}(u)$, containing all the free neighbors of u. That no MIS containing a vertex appearing before v_i in $\texttt{BL}(u)$ is counted, is ensured by either its deletion (because it is a neighbor of v_i) or the marking of this vertex. So, every MIS containing v_i but neither u (removed as it is a neighbor of v_i) nor a vertex appearing before v_i in $\texttt{BL}(u)$ is counted exactly once.

5.3 Running Time Analysis of `#MaximalIS`

We analyze the running time of our algorithm using the Measure & Conquer technique which has recently been used to establish several of today's best known exact exponential-time algorithms for NP-hard problems. For some important results and more details on the technique, we refer to [6–8, 23]. To analyze the running time of our algorithm, we use the following measure $\mu(G)$ of a marked graph G.

$$\mu = \mu\big(G = (F, M, E)\big) = \sum_{i=1}^{n-1} w_i |V_i|$$

The weights w_i, $1 \leq i \leq n-1$ are real numbers taken from $[0, 1]$ that will be fixed later. For $1 \leq i \leq n-1$, V_i denotes the set of vertices of degree i in G. The following values will be useful in the analysis.

$$\Delta w_i = \begin{cases} w_i - w_{i-1} & \text{if } 2 \leq i \leq n-1 \\ w_1 & \text{if } i = 1 \end{cases}$$

To further simplify the forthcoming analysis, we assume that $w_i = 1$ (for $4 \leq i \leq n-1$), $w_{i-1} \leq w_i$ (for $2 \leq i \leq n-1$), and $\Delta w_i \geq \Delta w_{i+1}$ (for $1 \leq i \leq n-1$). It is not hard to see that an application of a reduction rule will not increase the measure of the marked graph. Furthermore no reduction rule can be applied more than n times, respectively m times for (R3). Finally, each reduction rule can be implemented to run in polynomial time, and thus for each subproblem the running time of our algorithm, excluding the recursive calls by branching rule (B), is polynomial. Consequently we need to analyze the maximum number of such recursive calls, i.e. the maximum number of subproblems generated by a recursive call by (B), during the execution of our algorithm on a marked graph of measure μ, which we denote by $T(\mu)$.

We only have to analyze the changes in measure when applying branching rule (B).

Case 1: (B) is applied to a marked vertex u with $d(u) = 2$.
Let v_1 and v_2 be its two neighbors. By (R3), i.e. since (R3) could not be applied, $v_1, v_2 \in F$, and by (R2), $d(v_1), d(v_2) \geq 2$.

(a) Suppose $d(v_1) = d(v_2) = 2$. For $i \in \{1, 2\}$, let x_i be the other neighbor of v_i. If $d(x_1) = d(x_2) = 1$ then the algorithm deals with a component of constant size, and the number of MISs of such a component can be computed in constant time. Suppose now that $d(x_1) \geq 2$. In the first branch

(or subproblem) u, v_1 and x_1 are removed. In the second branch u, v_2 and x_2 are removed. Thus, the corresponding recurrence is majorized by $T(\mu) \leq T(\mu - 3w_2) + T(\mu - w_1 - 2w_2)$.

(b) Suppose $d(v_1) \geq 3$ and $d(v_2) \geq 2$. In the first branch u, v_1 and at least two other neighbors of v_1 are removed. In the second branch u, v_2 and the other neighbors of v_2, at least one, are removed. Thus, the corresponding recurrence is majorized by $T(\mu) \leq T(\mu - 2w_1 - w_2 - w_3) + T(\mu - w_1 - 2w_2)$. Since $w_2 \leq w_3$ and $w_2 \leq 2w_1$ (recall that $\Delta w_1 \geq \Delta w_2$), it follows that $3w_2 \leq 2w_1 + w_2 + w_3$ and thus the solution of the recurrence in case (b) is not worse than the one of case (a).

Case 2: Vertex u is chosen by the *else* statement of (B).
Thus u satisfies the conditions (i), (ii) and (iii). Let $[v_1, \ldots, v_{d_F(u)}]$ be the *Branching List*, short $\texttt{BL}(u)$, built by the algorithm. Given a vertex v_i, $1 \leq i \leq d_F(u)$, of $\texttt{BL}(u)$, we denote by $\texttt{Op}(v_i)$ the operation of adding v_i to the current MIS, removing $N[v_i]$ and marking the vertices $v_1, \ldots, v_{i-1}$ that are not adjacent to v_i.

Let Δ_u denote the gain on the measure obtained by adding u to the current MIS. Removing u and its neighbors from the graph decreases $\mu(G)$ by $w_{d(u)} + \sum_{v \in N(u)} w_{d(v)}$. Moreover, the decrease of the degrees of vertices in $N^2(u)$ implies a gain of $\sum_{x \in N^2(u)} (w_{d(x)} - w_{d(x)-d_{N(u)}(x)})$. Thus, $\Delta_u = w_{d(u)} + \sum_{v \in N(u)} w_{d(v)} + \sum_{x \in N^2(u)} (w_{d(x)} - w_{d(x)-d_{N(u)}(x)})$.

Let $\Delta_{\texttt{Op}(v_i)}$ denote the gain on the measure when $v_i \in \texttt{BL}(u)$, $1 \leq i \leq d_F(u)$, is selected and added to the maximal independent set. Again, by selecting vertex v_i the vertices of $N[v_i]$ are removed and thus a gain of $w_{d(v_i)} + \sum_{x \in N(v_i)} w_{d(x)}$ is obtained. Since neighbors of vertices of $N^2(v_i)$ have been removed, we gain $\sum_{y \in N^2(v_i)} (w_{d(y)} - w_{d(y)-d_{N(v_i)}(y)})$. The measure further decreases whenever among the marked vertices of $\{v_1, \ldots, v_{i-1}\}$, some of them have only one remaining free neighbor after the deletion of $N[v_i]$. By direct application of reduction rule (R1), these vertices and their neighbors are also removed from the graph. We denote this *extra* gain by $\text{marked}_1(\texttt{Op}(v_i))$ Thus, $\Delta_{\texttt{Op}(v_i)} = w_{d(v_i)} + \sum_{x \in N(v_i)} w_{d(x)} + \sum_{y \in N^2(v_i)} (w_{d(y)} - w_{d(y)-d_{N(v_i)}(y)}) + \text{marked}_1(\texttt{Op}(v_i))$.

Putting all together, we obtain the following general recurrence for case 2:

$$T(\mu) \leq T(\mu - \Delta_u) + \sum_{v_i \in \texttt{BL}(u)} T(\mu - \Delta_{\texttt{Op}(v_i)})$$

Finally, we conclude the time analysis by Measure & Conquer. We solve the corresponding system of linear recurrences and establish an upper bound on the worst case running time of our algorithm. Moreover, for some cases where a marked vertex of degree 2 appears, we combine the analysis of the case with the subsequent branching on this vertex. The key step is to choose the weights w_1, w_2 and w_3 such that the worst-case solution taken over all recurrences is minimized (see e.g. [7, 8]). Using the weights $w_1 = 0.8512$, $w_2 = 0.9194$ and $w_3 = 0.9877$, we obtain:

Theorem 2. *Algorithm* `#MaximalIS` *counts all MISs of a given graph G in time $O(1.3642^n)$, where n is the number of vertices of G.*

Typically using a computer program, first the collection of recurrences that are obtained for all possible cases of vertices, degrees, etc. in the general recurrence are computed and then the optimal values of the weights are computed. Although for our problem the number of recurrences is still rather moderate, due to space limitations the detailed analysis is not given in this extended abstract.

Remark 1. Given a marked graph of maximum degree 2, `#MaximalIS` takes exponential time. A dynamic programming approach can also be used to count in polynomial time all MISs of a such marked graph. Adding this polynomial time procedure to `#MaximalIS` is likely to be of help in implementations of the algorithm, however it does not improve its worst case running time.

For most non-trivial Branch-and-Reduce algorithms, it is not known whether the upper bound of the running time provided by the currently available analysis is tight or not. A lower bound for the worst case running time of such algorithms is therefore desirable (see e.g. [7, 8]).

Theorem 3. *There exists an infinite family of graphs for which* `#MaximalIS` *takes time* $\Omega(1.3247^n)$*, and thus its worst case running time is* $\Omega(1.3247^n)$*.*

Proof. The lower bound for the running time of Algorithm `#MaximalIS` established here uses the same family of graphs as the lower bound for an algorithm computing a minimum independent dominating set [11].

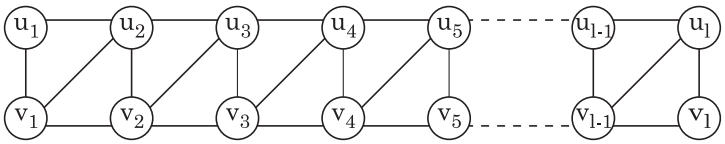

Fig. 1. The graph G_l.

Consider the graph G_l of Figure 1. It has $n = 2l$ vertices. Note that none of the reduction or halting rules are applicable to G_l. The first branching of `#MaximalIS` is on vertex u_1 or vertex v_l. Without loss of generality, suppose the algorithm always chooses the vertex with smallest index when it has more than one choice (i.e. it chooses u_1 for the first recursive call).

The branching rule (B) then makes recursive calls on graphs with $n-3$, $n-4$ and $n-5$ vertices, not marking any vertex. The structure of all resulting graphs is similar to G_l: either isomorphic to G_{l-2} or equal to $G_l \setminus N[u_1]$ or $G_l \setminus N[u_2]$. The subsequent recursive calls again remove 3, 4 and 5 vertices in each case and do not mark any vertices. Unless the graph has at most 4 vertices, each application of branching rule (B) satisfies the recurrence $T(n) = T(n-3)+T(n-4)+T(n-5)$ for this graph and therefore the running time for this class of graphs is $\Omega(\alpha^n)$ where α is the positive root of $x^{-3}+x^{-4}+x^{-5}-1$ (i.e. $1.3247 < \alpha < 1.3248$). □

5.4 Algorithm to Count All Maximal Bicliques

Finally, we consider the problem of counting all maximal bicliques of a graph $G = (V, E)$. Let $G' = (V', E')$ be a copy of G. Let $G'' = (V'', E'')$ where $V'' = V \cup V'$ and $E'' = E \cup E' \cup \{xy' : x = y \text{ or } xy \notin E\}$.

Lemma 3. *The number of MISs of G'' equals twice the number of maximal bicliques of G.*

Proof. We show that there is a one-to-one correspondence between the bicliques of G and the symmetric pairs of independent sets of G''.

Let $X \cup Y$ be a biclique of G. Clearly, X, Y are independent sets in G and their copies X', Y' are independent sets in G'. Let $x \in X$ and $y \in Y$. Then $xy, x'y' \in E''$ and $xy', x'y \notin E''$. So, $X \cup Y'$ and $X' \cup Y$ are independent sets in G''.

Let $X, Y \subseteq V$ be such that $X \cup Y'$ is an independent set in G'' where X', Y' are the copies of X, Y. Hence X, Y are independent sets in G. Let $x \in X$ and $y' \in Y'$. Then $xy \in E$. So, $X \cup Y$ is a biclique in G. By the symmetry of G'', the independent set $X' \cup Y$ in G'' also corresponds to the biclique $X \cup Y$ in G.

Clearly, this correspondence also holds for maximality by inclusion of vertices. This implies that $X \cup Y$ is a maximal biclique of G iff $X \cup Y'$, and thus also $Y \cup X'$, are MISs of G''. □

With this construction and the algorithm for counting all MISs of a graph, we are now able to give an algorithm for counting all maximal bicliques of a graph.

Theorem 4. *There is an algorithm that counts all maximal bicliques of a graph in time $O(1.3642^n)$.*

Proof. The algorithm simply calls `#MaximalIS`$\big((V'', \emptyset, E'')\big)$ and divides the result by 2. Notice that G'' has $2n$ vertices and that every vertex of G'' has degree n. The first application of branching rule (B) makes $n + 1$ recursive calls and in each one, $n + 1$ vertices are removed from the marked graph. Thus the running time is $(n + 1)(c^{n-1})n^{O(1)}$ where $c^n n^{O(1)}$ is the running time of `#MaximalIS` on a graph with n vertices. The constant $c = 1.3642$ was rounded to derive the running time for `#MaximalIS`, and thus the running time of the algorithm to count maximal bicliques is $O(1.3642^n)$. □

References

1. Alexe, G., Alexe, S., Crama, Y., Foldes, S., Hammer, P.L., Simeone, B.: Consensus algorithms for the generation of all maximal bicliques. Discrete Appl. Math. 145, 11–21 (2004)
2. Amilhastre, J., Vilarem, M.C., Janssen, P.: Complexity of minimum biclique cover and minimum biclique decomposition for bipartite dominofree graphs. Discrete Appl. Math. 86, 125–144 (1998)
3. Dawande, M., Swaminathan, J., Keskinocak, P., Tayur, S.: On bipartite and multipartite clique problems. J. Algorithms 41, 388–403 (2001)

4. Dias, V.M.F., Herrera de Figueiredo, C.M., Szwarcfiter, J.L.: Generating bicliques of a graph in lexicographic order. Theoret. Comput. Sci. 337, 240–248 (2005)
5. Dias, V.M.F., Herrera de Figueiredo, C.M., Szwarcfiter, J.L.: On the generation of bicliques of a graph. Discrete Appl. Math. 155, 1826–1832 (2007)
6. Fomin, F.V., Gaspers, S., Pyatkin, A.V., Razgon, I.: On the minimum feedback vertex set problem: Exact and enumeration algorithms. Algorithmica, Special Issue on Parameterized and Exact Algorithms, to appear.
7. Fomin, F.V., Grandoni, F., Kratsch, D.: Measure and conquer: Domination – A case study. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 192–203. Springer, Heidelberg (2005)
8. Fomin, F.V., Grandoni, F., Kratsch, D.: Measure and conquer: A simple $O(2^{0.288n})$ independent set algorithm. In: SODA 2006, pp. 18–25. ACM Press (2006)
9. Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the Theory of NP-completeness. Freeman, New York (1979)
10. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin (1996)
11. Gaspers, S., Liedloff, M.: A branch-and-reduce algorithm for finding a minimum independent dominating set in graphs. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 78–89. Springer, Heidelberg (2006)
12. Hochbaum, D.S.: Approximating clique and biclique problems. J. Algorithms 29, 174–200 (1998)
13. Hujter, M., Tuza, Z.: The number of maximal independent sets in triangle-free graphs. SIAM J. Discrete Math. 6, 284–288 (1993)
14. Johnson, D.S., Papadimitriou, C.H.: On generating all maximal independent sets. Inf. Process. Lett. 27, 119–123 (1988)
15. Makino, K., Uno, T.: New algorithms for enumerating all maximal cliques. In: Hagerup, T., Katajainen, J. (eds.) SWAT 2004. LNCS, vol. 3111, pp. 260–272. Springer, Heidelberg (2004)
16. Moon, J.W., Moser, L.: On cliques in graphs. Israel J. Math. 3, 23–28 (1965)
17. Nourine, L., Raynaud, O.: A Fast Algorithm for Building Lattices. Inf. Process. Lett. 71, 199–204 (1999)
18. Nourine, L., Raynaud, O.: A fast incremental algorithm for building lattices. J. Exp. Theor. Artif. Intell. 14, 217–227 (2002)
19. Okamoto, Y., Uno, T., Uehara, R.: Linear-Time Counting Algorithms for Independent Sets in Chordal Graphs. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 433–444. Springer, Heidelberg (2005)
20. Peeters, R.: The maximum edge biclique problem is NP-complete. Discrete Appl. Math. 131, 651–654 (2003)
21. Prisner, E.: Bicliques in Graphs I: Bounds on Their Number. Combinatorica 20, 109–117 (2000)
22. Robson, J.M.: Algorithms for maximum independent sets. J. Algorithms 7, 425–440 (1986)
23. van Rooij, J.M.M., Bodlaender, H.L.: Design by measure and conquer: a faster exact algorithm for dominating set. In: Albers, S., Weil, P. (eds.) STACS 2008, pp. 657–668. IBFI, Schloss Dagstuhl, Germany (2008)
24. Wahlström, M.: A tighter bound for counting max-weight solutions to 2SAT instances. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 202–213. Springer, Heidelberg (2008)
25. Yannakakis, M.: Node and edge deletion NP-complete problems. In: STOC 1978, pp. 253–264. ACM (1978)