Algorithmica (2012) 62:637–658
DOI 10.1007/s00453-010-9474-1

On Independent Sets and Bicliques in Graphs

Serge Gaspers · Dieter Kratsch · Mathieu Liedloff

Received: 13 August 2009 / Accepted: 2 November 2010 / Published online: 11 November 2010

Abstract Bicliques of graphs have been studied extensively, partially motivated by the large number of applications. In this paper we improve Prisner's upper bound on the number of maximal bicliques (Combinatorica, 20, 109–117, 2000) and show that the maximum number of maximal bicliques in a graph on n vertices is $\Theta(3^{n/3})$. Our major contribution is an exact exponential-time algorithm. This branching algorithm computes the number of distinct maximal independent sets in a graph in time $O(1.3642^n)$, where n is the number of vertices of the input graph. We use this counting algorithm and previously known algorithms for various other problems related to independent sets to derive algorithms for problems related to bicliques via polynomial-time reductions.

Keywords Counting algorithms · Combinatorial bound · Maximal bicliques · Maximal independent sets · Exact exponential time algorithm

A large part of the research was done while Serge Gaspers was visiting the University of Metz.
A preliminary version of this paper appeared in the proceedings of WG 2008 [18]. Serge Gaspers acknowledges partial support of NFR and of Conicyt Chile via the project Basal-CMM.

S. Gaspers (✉)
Institute of Information Systems, Vienna University of Technology, Favoritenstraße 9-11, 1040 Vienna, Austria
e-mail: gaspers@kr.tuwien.ac.at

D. Kratsch
LITA, Université Paul Verlaine-Metz, 57045 Metz Cedex 01, France
e-mail: kratsch@univ-metz.fr

M. Liedloff
LIFO, Université d'Orléans, 45067 Orléans Cedex 2, France
e-mail: liedloff@univ-orleans.fr

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 $y \in Y$. 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.

Note that our definition allows X or Y to be an empty set, and thus $X \cup Y$ to be an independent set. In [9, 10], such $X \cup Y$ are not considered to be bicliques, whereas independents sets are considered to be bicliques in [19, 27] and in the present paper. Bicliques with at least one edge are called *proper* 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.

Applications Research on maximal bicliques and algorithms to enumerate all maximal bicliques of (bipartite) graphs with polynomial delay is motivated by various applications of bicliques in (bipartite) graphs. Applications of bicliques in automata and language theory, graph compression, artificial intelligence and biology are discussed in [3]. An important application in data mining is based on the formal concept analysis [15] 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 [16], among them the NP-completeness of the balanced complete bipartite subgraph problem. The maximum biclique problem is polynomial for bipartite graphs [7], and NP-hard for general graphs [32]. The maximum edge biclique problem was shown to be NP-hard by Peeters [26].

Approximation algorithms for node and edge deletion biclique problems are given by Hochbaum [19]. Enumerating maximal bicliques has attracted a lot of attention in the last decade. The algorithms in [23, 24] 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 (proper) bicliques in bipartite graphs [10, 21] and general graphs [9]. There are also polynomial delay algorithms to enumerate all maximal non-induced bicliques of a graph [2, 10].[1]

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 also 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 [27].

[1] When the condition that X and Y are independent sets in the definition of a biclique is omitted, then (X, Y) is called a *non-induced biclique* of G. In this case a different maximality notion is used. See for example [2].

Our Results We use polynomial-time Turing reductions to transform results on maximal independent sets into results on maximal bicliques. In this way we improve upon Prisner's upper bound and show that the maximum number of maximal bicliques in a graph on n vertices is at most $\frac{1}{3^{1/3}-1} \cdot 3^{n/3}$. Our main contribution is an algorithm to count all maximal independent sets in a graph. This branching algorithm has worst-case running time $O(1.3642^n)$ and this upper bound is established by making use of the Measure & Conquer technique, see e.g. [13]. We also provide a lower bound for the running time of this counting algorithm. Finally we show how to use this algorithm to count all maximal bicliques of a graph within the same time bound.

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, 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 Improving Prisner's Bound

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 poly nomial-time Turing reductions (in fact mainly Karp reductions) have been used in various hardness proofs for problems on bicliques [16]. The following property is central for our purpose.

Lemma 1 *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 in 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 exact exponential-time algorithms.

Corollary 2 *Given an algorithm to find a maximum independent set of a graph in time $O^*(c^n)$, it can be used to establish an algorithm to find a maximum biclique of a graph in time $O^*(c^n)$. Given an algorithm to count all independent sets of size k of a graph in time $O^*(c^n)$, it can be used to establish an algorithm to count all bicliques of size k of a graph in time $O^*(c^n)$.*[2]

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 Lemma 1 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 $G^i = G[\{v_i, v_{i+1}, \ldots, v_n\}]$ using Lemma 1. Adding up the results gives the number of bicliques of size k of G. □

By this corollary and the algorithms in [28, 31], 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 $O(1.2377^n)$.

We emphasize that the approach of Corollary 2 is not directly applicable to use an algorithm counting the maximal independent sets of a graph to establish one 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. Such counting algorithms are established in the next section.

We finish this section with a combinatorial problem. The maximum number of maximal bicliques in a graph on n vertices has been studied by Prisner [27]. 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 significantly improve the upper bound.

[2] Throughout the paper we write $f(n) = O^*(g(n))$ if $f(n) \leq p(n) \cdot g(n)$ for some polynomial $p(n)$.

Theorem 3 *The maximum number of maximal bicliques in a graph on n vertices is at most* $c \cdot 3^{n/3}$ *where* $c = \frac{1}{3^{1/3}-1} < 2.2612$.

Proof Let n be a positive integer and let G be any graph on n vertices. Let $v_1, v_2, \ldots, v_n$ be the vertices of G and let $\mathcal{B}$ be the set of maximal bicliques of G. We show that $|\mathcal{B}|$ is at most $c \cdot 3^{n/3}$.

Let $\mathcal{B}_i \subseteq \mathcal{B}$, $1 \leq i \leq n$, be the set of bicliques B of $\mathcal{B}$ such that $v_i \in B$ and $\{v_1, v_2, \ldots, v_{i-1}\} \cap B = \emptyset$. Note that by definition the $\mathcal{B}_i$'s form a partition of $\mathcal{B}$.

Consider a $\mathcal{B}_i$ for any $1 \leq i \leq n$. Any maximal biclique $B \in \mathcal{B}_i$ contains no v_j with $j < i$. Thus B is also a maximal biclique of $G \setminus \{v_1, \ldots, v_{i-1}\}$ which contains $n - i + 1$ vertices. Applying Lemma 1, there is a one-to-one correspondence between the maximal bicliques B of $\mathcal{B}_i$ and the maximal independent sets $B - v_i$ of the graph $H_{v_i} \setminus \{v_1, \ldots, v_{i-1}\}$. Note that H_{v_i} contains $n - i$ vertices. By a well-known theorem of Moon and Moser [22], the maximum number of maximal independent sets in a graph on n' vertices is $3^{n'/3}$. Thus the number of maximal bicliques of $\mathcal{B}_i$ is at most $3^{(n-i)/3}$.

As a consequence, $|\mathcal{B}| \leq \sum_{i=1}^{n} 3^{(n-i)/3} < \frac{1}{3^{1/3}-1} \cdot 3^{n/3}$. □

Corollary 4 *The maximum number of maximal bicliques in a graph is* $\Theta(3^{n/3})$.

4 Counting Algorithms

There are decision, optimization, counting and enumeration problems. Algorithms solving hard problems of any of these types are studied in the domain of exact exponential-time algorithms using worst-case running times to measure the quality of algorithms. Obviously, each enumeration algorithm can be used to solve corresponding counting, optimization and decision problems. On the other hand, the worst-case running time of an enumeration algorithm is lower bounded by the number of objects to be enumerated. Thus, for example, each algorithm to enumerate all maximal independent sets of a graph or to enumerate all maximal bicliques of a graph has a worst-case running time of $\Omega(3^{n/3})$ ([22, 27]).

Counting problems are a classical subject in algorithms and complexity. Recently within the domain of exact exponential-time algorithms the time complexity of counting problems attracts a lot of attention. For example, it is interesting that the best known algorithm to compute the chromatic number of a graph [4] and the best known algorithm to compute a minimum dominating set of a graph [30], both solve in fact a corresponding counting problem (within the same running time).

In this section we present an exact exponential-time algorithm to count all maximal independent sets of a graph. Based on a polynomial-time Turing reduction this algorithm can be used to establish an algorithm to count all maximal bicliques of a graph. No such counting algorithms of running time $O^*(c^n)$, with $c < \sqrt[3]{3}$ were known prior to our work. On the other hand, the problem to count the maximal independent sets is known to be $\#P$-complete even when restricted to chordal graphs [25]. Our goal is to construct and analyze a branching algorithm solving the counting problem.

4.1 Algorithm to Count All Maximal Independent Sets

We would first like to say a word of precaution. Even if the problem of counting all maximal independent sets of a graph seems very similar to the problems of counting all maximum independent sets of a graph, or all independent sets of a given size k, there is a fundamental difference coming from the notion of maximality. The best known exact exponential-time algorithm to count all independent sets of maximum size or of size k [5, 6, 14, 31] rely on a branching strategy which has the following properties: vertices that are decided not to be in the counted independent sets of a subproblem (generated by a branching algorithm) can be deleted and removed from further consideration, and graphs of maximum degree 2 can be handled in polynomial time. But if the algorithm is supposed to count all maximal independent sets, this strategy does not work (unless P = #P). Consider a graph $G = (F \cup M, E)$ for which we would like to count all maximal independent sets of G that are included in F. In other words, M is the set of vertices that have been decided not to be in any maximal independent set in the current subproblem, but for each of them, a neighbor must be added to ensure the maximality of the counted independent sets. By a simple reduction from the #Satisfiability problem, requiring to count all satisfying assignments to a boolean formula, it can be shown that this problem is #P-hard even if $G[F]$ has maximum degree 1 (an edge in $G[F]$ corresponds to a variable, its end points to the true/false value of this variable, and the vertices in M correspond to the clauses of the formula).

Our algorithm deals with *marked graphs* $G = (F, M, E)$, where 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 $d_D(u)$. For a marked graph $G = (F, M, E)$, the marked graph induced by the vertex sets $F' \subseteq F$ and $M' \subseteq M$ is $G[F', M'] = (F', M', E \cap ((F' \cup M') \times (F' \cup M')))$.

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

Given a marked graph G, our algorithm computes the number of maximal independent sets of $G = (F, M, E)$ satisfying Π. Thus, a marked vertex u is used to force that each maximal independent set 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 maximal independent sets of G satisfying Π.

Given a simple graph $G' = (V, E)$, `#MaximalIS(`$G = (V, \emptyset, E)$`)` returns the number of maximal independent sets of G'. See Fig. 1 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 in Subsects. 4.3 and 4.4. The branching rule (B) selects a vertex u, orders its free neighbors in a list $\texttt{BL}(u) = [v_1, v_2, \ldots, v_{d_F(u)}]$

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

`// Reduction rules`
if $F \cup M$ *is empty* **then**
  return 1; `(H1)`
if *there exists* $u \in M$ *such that* $d_F(u) = 0$ **then**
  return 0; `(H2)`
if *there exists* $u \in M$ *such that* $N_F(u) = \{v\}$ **then**
  return `#MaximalIS`$\big(G[F \setminus N[v], M \setminus N(v)]\big)$; `(R1)`
if *there exists* $u \in F$ *such that* $d_F(u) = 0$ **then**
  return `#MaximalIS`$\big(G[F \setminus N[u], M \setminus N(u)]\big)$; `(R2)`
if *there exists* $u, v \in M$ *such that* $\{u, v\} \in E$ **then**
  return `#MaximalIS`$\big((F, M, E \setminus \{u, v\})\big)$; `(R3)`
if *there exists* $u, v \in F$ *such that* $N[u] = N[v]$ **then**
  $count \leftarrow$ `#MaximalIS`$\big(G[F \setminus \{v\}, M]\big)$;
  Let MIS_u be the number of maximal independent sets computed by `#MaximalIS`$\big(G[F \setminus \{v\}, M]\big)$ containing u;
  return $\mathrm{MIS}_u + count$; `(R4)`
if *there exists* $u \in M$ *and* $v \in N(u)$ *such that* $N[v] \subseteq N[u]$ **then**
  return `#MaximalIS`$\big(G[F, M \setminus \{u\}]\big)$; `(R5)`
if *there exists* $u, v \in M$ *such that* $N(u) = N(v)$ **then**
  return `#MaximalIS`$\big(G[F, M \setminus \{v\}]\big)$; `(R6)`
if *there exists* $u \in F \cup M$ *and* $v \in F$ *such that* $N(u) = N(v)$ **then**
  return `#MaximalIS`$\big(G[F \setminus \{v\}, M]\big)$; `(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, and
    (iii) among all vertices fulfilling (i) and (ii), u has maximum dual degree (that is the sum of the degrees of its neighbors);
Let $\mathtt{BL}(u) \leftarrow [v_1, \ldots, v_{d_F(u)}]$ 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 $V \setminus N(u)$; if there are several choices, choose v_1 of minimum degree,
  (ii) append (in any order) the vertices of $N(v_1) \cap N_F(u)$ to the ordered list, and
  (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 maximal independent set)`
  $count \leftarrow$ `#MaximalIS`$\big(G[F \setminus N[u], M \setminus N(u)]\big)$;
foreach $v_i \in \mathtt{BL}(u)$ **do** `// mark each vertex of` M' `and select` v_i
  $M' \leftarrow \{v_j \in \mathtt{BL}(u) : 1 \le j < i \text{ and } \{v_j, v_i\} \notin E\}$;
  $count \leftarrow count +$ `#MaximalIS`$\big(G[F \setminus (M' \cup N[v_i]), (M \cup M') \setminus N(v_i)]\big)$;
return *count*;

Fig. 1 Algorithm `#MaximalIS` counting all maximal independent sets

and makes a recursive call (that is a branching) counting all maximal independent sets containing u, and a recursive call for each $i = 1, 2, \ldots, d_F(u)$ where it counts all maximal independent sets 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.

4.2 Correctness of `#MaximalIS`

We show the correctness of the branching and reduction rules of `#MaximalIS`. (H1) If the input graph has no vertices then the only maximal independent set is the empty set. (H2) If there is a marked vertex u without any free neighbor then there is no maximal independent set satisfying Π. (R1) If a marked vertex u has only one free neighbor, it has to be in the maximal independent set to satisfy Π. (R2) By maximality, each free vertex without any free neighbor has to belong to all maximal independent sets. (R3) Since marked vertices cannot belong to any maximal independent set, 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 maximal independent set containing a neighbor of u does not contain v. Moreover, every maximal independent set 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 maximal independent sets containing a neighbor of u plus twice the number of maximal independent sets containing u. (Note that the algorithm can easily be implemented such that the number of maximal independent sets containing u is obtained from the recursive call. For example, keep a counter to associate to each free vertex the number of maximal independent sets 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 maximal independent set 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 maximal independent set containing a neighbor of u does not contain v and every maximal independent set containing u (if u is free) also contains v. Thus the number of maximal independent sets 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 maximal independent set. By induction and the fact that $N[u]$ is removed if the algorithm decides to add u to the current maximal independent set, every maximal independent set containing u is counted and it is counted only once. Consider the possibility that at least one neighbor of u is in the current maximal independent set 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 maximal independent set 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 maximal independent set 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.

4.3 Running Time Analysis of `#MaximalIS`

The goal is to analyze the running of the branching algorithm. Measure & Conquer is a technique available since a few years for this purpose. For an introduction to Measure & Conquer we refer the reader to [13]. Measure & Conquer has been used to establish several of the fastest known exact exponential-time algorithms for well-studied NP-hard problems [11–13, 17, 20, 29].

To analyze the running time of our algorithm, we use the following measure $\mu(G)$ of a marked graph G.

$$\begin{aligned}\mu &:= \mu\big(G = (F, M, E)\big)\\ &:= \sum_{i=1}^{n-1} w_i|V_i| + m_2 \cdot \mathsf{K}_\delta(G \text{ has no marked vertex of degree 2})\end{aligned}$$

The weights m_2 and w_i, $1 \le i \le n-1$ are real numbers taken from $[0, 1]$ that will be fixed later. For $1 \le i \le n-1$, V_i denotes the set of vertices of degree i in G and K_δ is the logical Kronecker Delta returning 1 if its argument is true and 0 otherwise. The following values will be useful in the analysis.

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

To further simplify the forthcoming analysis, we assume:

$$\begin{aligned} w_i &= 1, && 4 \le i \le n-1\\ w_{i-1} &\le w_i, && 2 \le i \le n-1, \quad \text{and}\\ \Delta w_i &\ge \Delta w_{i+1}, && 1 \le i \le n-1. \end{aligned}$$

It is not hard to see that an application of a reduction rule will not increase $\sum_{i=1}^{n-1} w_i|V_i|$. 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, that is 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), that is since (R3) could not be applied, $v_1, v_2 \in F$, and by (R2), $d(v_1), d(v_2) \ge 2$.

(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 maximal independent sets 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. In both branches, the graph might not have a marked vertex of degree 2 any more. Thus, the corresponding recurrence is majorized by

$$T(\mu) \geq T(\mu - 3w_2 + m_2) + T(\mu - w_1 - 2w_2 + m_2).$$

(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

$$T(\mu) \leq T(\mu - 2w_1 - w_2 - w_3 + m_2) + T(\mu - w_1 - 2w_2 + m_2).$$

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 maximal independent set, 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 maximal independent set. 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 $\sum_{x \in N^2(u)} (w_{d(x)} - w_{d(x)-d_{N(u)}(x)})$. Let $m_2(u)$ be equal to m_2 if the subinstance obtained from adding u to the current maximal independent set has a marked vertex of degree 2 after exhaustively applying all the reduction rules, and equal to 0 otherwise. Then,

$$\Delta_u = w_{d(u)} + \sum_{v \in N(u)} w_{d(v)} + \sum_{x \in N^2(u)} (w_{d(x)} - w_{d(x)-d_{N(u)}(x)}) + m_2(u).$$

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,

$$\Delta_{\texttt{Op}(v_i)} = w_{d(v_i)} + \sum_{x \in N(v_i)} w_{d(x)} + \sum_{y \in N^2(v_i)} (w_{d(y)} - w_{d(y)-d_{N(v_i)}(y)}) + \text{marked}_1(\texttt{Op}(v_i)) + m_2(v_i).$$

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

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

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. The key step is to choose the weights m_2, w_1, w_2 and w_3 such that the worst-case solution taken over all recurrences is minimized (see for example [13]). Using the weights $w_1 = 0.8473$, $w_2 = 0.9181$, $w_3 = 0.9875$ and $m_2 = 0.4$, we obtain:

Theorem 5 *Algorithm* `#MaximalIS` *counts all maximal independent sets 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. For our problem the number of recurrences is still rather moderate and therefore we are able to provide for the interested reader the details of the analysis and list all possible worst cases in the next subsection.

4.4 Detailed Running Time Analysis of Algorithm `#MaximalIS`

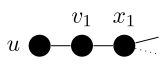
In this subsection we provide a detailed running time analysis of Algorithm `#MaximalIS`. The branching corresponding to the selection of a marked vertex of degree 2 has already been analyzed in detail in our high level analysis in Subsect. 4.3. Here we give a list of cases, corresponding to the analysis in Case 2 in Subsect. 4.3. Each case has a number, a condition telling us in which case we are, a picture and a recurrence based on the measure of the created subinstances in this case. For those cases, where it is not immediate how the recurrence is obtained, a comment is added observing facts needed to obtain it.

Denote the neighbors of u by $v_1, v_2, \ldots, v_{d(u)}$. For a selected vertex u, we say that x is an *external* neighbor of a vertex $v \in N(u)$ if x is a vertex of $N(v) \setminus N[u]$.

Note that the algorithm can apply the branching rule on a r-regular graph, $2 \leq r \leq 4$. However, when dealing with such an r-regular graph any subsequent recursive calls will never be on an r-regular graph again (see for example [28]). Thus, these graphs are not relevant to establish the running time bound. If the graph is 1-regular, then the algorithm would treat it in polynomial time since the size of each connected component is bounded by a constant.

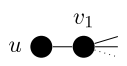
In the following case analysis, cases number 1 (with $d(x_1) = 4$), 18 and 21 correspond to the tight cases.

1)	$d(u) = 1, d(v_1) = 2$



$T(\mu) \leq T(\mu - w_1 - w_2 - \Delta w_{d(x_1)}) + T(\mu - w_1 - w_2 - w_{d(x_1)})$

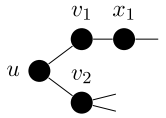
2)	$d(u) = 1, d(v_1) \geq 3$



$T(\mu) \leq T(\mu - w_1 - w_{d(v_1)}) + T(\mu - w_{d(v_1)} - (d(v_1) - 1) \cdot w_1 - w_2)$

Comment: v_1 has a neighbor of degree at least 2, otherwise $N[v_1]$ is a connected component.

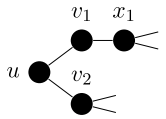
3)	$d(u) = 2, d(v_1) = 2, d(v_2) = 3, d(x_1) = 2$, x_1 being the other neighbor of v_1



$T(\mu) \leq T(\mu + w_1 - 3w_2 - w_3) + T(\mu - 2w_2 - w_3) + T(\mu - 5w_2 - w_3)$

Comment: $\{v_1, v_2\} \notin E$, as $d(x_1) \neq d(v_2)$. In the branch where v_2 is selected, x_1 is also selected by (R1) as v_1 becomes marked and has a unique neighbor. As $N(u) \neq N(x_1)$, which is ensured by (R6) and (R7), x_1 and v_2 are not adjacent.

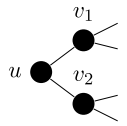
4)	$d(u) = 2, d(v_1) = 2, d(v_2) = 3, d(x_1) \geq 3$



$T(\mu) \leq T(\mu - 2w_2 - w_3) + T(\mu - w_2 - 2w_3) + T(\mu - 2w_2 - 4w_3)$

Comment: $\{v_1, v_2\} \notin E$, otherwise $N[u] = N[v_1]$ and (R4) or (R5) would apply. When v_2 is selected, x_1 is also selected by (R1). By the selection rule of u, $d(x_1) = 3$ and no common neighbor of v_2 and x_1 has degree 2. If v_2 and x_1 are adjacent, the last branch can be ignored as the instance has no maximal independent set by halting rule (H2). For analyzing the last branch, also note that $w_3 \leq 2w_2$ as $\Delta w_3 \leq \Delta w_2$.

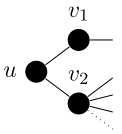
5)	$d(u) = 2, d(v_1) = 3, d(v_2) = 3$



$T(\mu) \leq T(\mu - w_2 - 2w_3) + 2T(\mu - 4w_3)$

Comment: The vertices of degree 2 in $N^2(u)$ are not adjacent to both v_1 and v_2 (otherwise they have the same open neighborhood as u). Moreover, two adjacent vertices in $N^2(u)$ of degree 2 are not adjacent to the same vertex in $N(u)$ due to the reduction rules. So, they have neighbors outside $N[u]$ of degree at most 3.

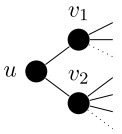
6)	$d(u) = 2, d(v_1) = 2, d(v_2) \geq 4$



$$T(\mu) \leq T(\mu - 2w_2 - w_4) + T(\mu - 3w_2) + T(\mu - 6w_2 - w_4)$$

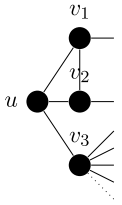
Comment: v_1 and v_2 are not adjacent due to (R4) and (R5). If they have a common neighbor, ignore the last branch. In the last branch, v_2 and the external neighbor of v_1 are selected.

7)	$d(u) = 2, d(v_1) \geq 3, d(v_2) \geq 4$



$$T(\mu) \leq T(\mu - w_2 - w_3 - w_4) + T(\mu - 3w_2 - w_3) + T(\mu - 4w_2 - w_4)$$

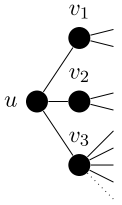
8)	$d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) \geq 5$, v_1 and v_2 are adjacent



$$T(\mu) \leq T(\mu - 3w_3 - w_4) + 2T(\mu - 4w_3) + T(\mu - 9w_3 - w_4)$$

Comment: v_1 and v_2 are not adjacent to v_3, otherwise (R4) or (R5) would apply as v_1 or v_2 would have the same closed neighborhood as u. Moreover, v_1 and v_2 do not share the same external neighbor otherwise v_1 and v_2 have the same closed neighborhood. If v_3 has a common neighbor in $N^2(u)$ with v_1 or v_2, then ignore the last branch, otherwise v_3 and both external neighbors of v_1 and v_2 are selected.

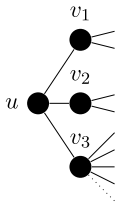
9)	$d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) \geq 5$, $N(u)$ is independent, in the last branch v_1 and v_2 disappear by reduction rules



$$T(\mu) \leq T(\mu - 3w_3 - w_4) + 2T(\mu + w_2 - 5w_3) + T(\mu - 7w_3 - w_4)$$

Comment: In this case, when v_3 is selected, v_1 and v_2 are removed by recursively applying the reduction rules.

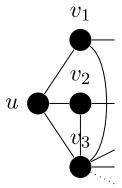
10)	$d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) \geq 5$, $N(u)$ is independent, in the last branch v_1 (or v_2) does not disappear by reduction rules



$$T(\mu) \leq T(\mu - 3w_3 - w_4) + 2T(\mu + w_2 - 5w_3) + T(\mu + 2w_2 - 7w_3 - w_4 - m_2)$$

Comment: In the last branch v_1 and v_2 are marked and become of degree 2. Therefore a marked vertex of degree 2 appears ($-m_2$).

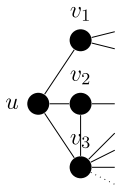
11)	$d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) \geq 5$, v_1 and v_2 are not adjacent, v_3 is adjacent to v_1 and v_2



$$T(\mu) \leq T(\mu + 2w_2 - 5w_3 - w_4) + T(\mu + w_1 - 4w_3 - w_4) + 2T(\mu - 5w_3 - w_4)$$

Comment: The external neighbors of v_1 and v_2 have degree 3, otherwise v_1 or v_2 would have a neighbor of higher degree or higher dual degree and would have been selected for branching instead of u. Moreover, the external neighbors of v_1 and v_2 are distinct, otherwise (R6) or (R7) would apply. Finally, note that $\texttt{BL}(u) = [v_1, v_3, v_2]$ or $\texttt{BL}(u) = [v_2, v_3, v_1]$. and are distinct.

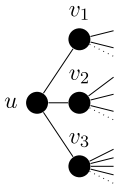
12)	$d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) \geq 5$, v_1 and v_2 are not adjacent, v_3 is adjacent to v_2 (or v_1)



$$T(\mu) \leq 2T(\mu + w_2 - 4w_3 - w_4) + T(\mu + w_2 - 6w_3 - w_4) + T(\mu - 6w_3)$$

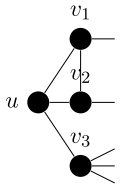
Comment: $\texttt{BL}(u) = [v_2, v_3, v_1]$ and the external neighbor of v_2 has degree 3, otherwise v_2 would have been selected for branching as it has either a neighbor of higher degree or higher dual degree than u.

13) $d(u) = 3, d(v_1) \geq 3, d(v_2) \geq 4, d(v_3) \geq 5$



$$T(\mu) \leq T(\mu - 2w_3 - 2w_4) + T(\mu - 4w_3) + T(\mu - 4w_3 - w_4) + T(\mu - 5w_3 - w_4)$$

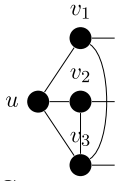
14) $d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) = 4$, v_1 and v_2 are adjacent



$$T(\mu) \leq T(\mu - w_3 - 3w_4) + 2T(\mu - 2w_3 - 2w_4) + T(\mu - 8w_3 - w_4)$$

Comment: v_1 and v_2 are not adjacent to v_3 because of (R4) and (R5) and they have distinct (by (R4) and (R5)) external neighbors of degree 3 or 4 (by the selection rule of u). If v_3 has a common neighbor with v_1 or v_2 (except u), ignore the last branch.

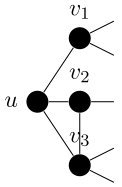
15) $d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) = 4$, v_1 and v_2 are not adjacent, v_3 is adjacent to v_1 and v_2



$$T(\mu) \leq T(\mu + 2w_2 - 5w_3 - w_4) + T(\mu + w_1 + w_2 - 5w_3 - w_4) + T(\mu + 2w_2 - 6w_3 - w_4) + T(\mu - 5w_3 - w_4)$$

Comment: Note that $\texttt{BL}(u) = [v_1, v_3, v_2]$ or $\texttt{BL}(u) = [v_2, v_3, v_1]$ and that v_1 and v_2 have distinct external neighbors of degree 3.

16) $d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) = 4$, v_1 and v_2 are not adjacent, v_3 is adjacent to v_2 (or v_1)

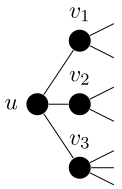


$$T(\mu) \leq T(\mu + 2w_2 - 4w_3 - 2w_4) + T(\mu + 2w_2 - 5w_3 - w_4) + T(\mu + 2w_2 - 6w_3 - w_4) + T(\mu + 2w_2 - 7w_3 - w_4)$$

Comment: $\texttt{BL}(u) = [v_2, v_3, v_1]$. The external neighbor of v_2 has degree 3 and neighbors of degree 3 and 3 or 4. In the third branch where v_3 is selected, $N[v_3]$ is deleted ($-4w_3 - w_4$), v_1 has its degree decreased ($+w_2 - w_3$), and another

vertex has its degree decreased from 3 to 2 ($+w_2 - w_3$): the external neighbor x of v_2 if it is not adjacent to v_3, or a neighbor of x if x and v_3 are neighbors and $N[x] \not\subseteq N[v_3]$, or the vertex in $N^2(x) \setminus N^2[u]$ in the remaining case.

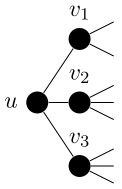
17) $d(u) = 3, d(v_1) = 3, d(v_2) = 3, d(v_3) = 4$, $N(u)$ is independent



$T(\mu) \leq T(\mu + 2w_2 - 3w_3 - 3w_4) + T(\mu + w_2 - 2w_3 - 3w_4) + T(\mu + w_2 - 2w_3 - 3w_4 - m_2) + T(\mu + 2w_2 - 6w_3 - w_4 - m_2)$

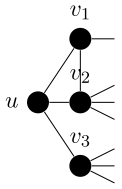
Comment: The external neighbors of v_1 and v_2 have degree 3 and 3 or 4. In the last two branches, a marked vertex of degree 2 is created.

18) $d(u) = 3, d(v_1) = 3, d(v_2) = 4, d(v_3) = 4$, v_1 is not adjacent to v_2 and v_3



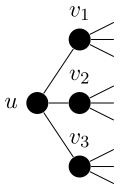
$T(\mu) \leq T(\mu - w_3 - 3w_4) + T(\mu - 2w_3 - 2w_4) + 2T(\mu + w_2 - 4w_3 - 2w_4)$

19) $d(u) = 3, d(v_1) = 3, d(v_2) = 4, d(v_3) = 4$, v_1 is adjacent to v_2 (or v_3)



$T(\mu) \leq T(\mu - w_3 - 3w_4) + T(\mu - 2w_3 - 2w_4) + T(\mu - 3w_3 - 2w_4) + T(\mu - 5w_3 - 2w_4)$

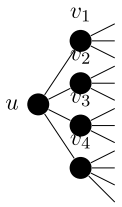
20) $d(u) = 3, d(v_1) = 4, d(v_2) = 4, d(v_3) = 4$



$T(\mu) \leq T(\mu - w_3 - 3w_4) + 3T(\mu - 2w_3 - 3w_4)$

Comment: Consider the branch where v_1 is selected. A total of 5 vertices disappear and at least 3 vertices of degree 4 either disappear or have their degree reduced from 4 to 3: the vertices in $N(u)$.

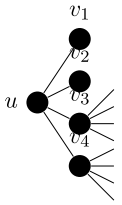
21) $d(u) = 4, d(v_1) = 4, d(v_2) = 4, d(v_3) = 4, d(v_4) = 5$



$$T(\mu) \leq 4T(\mu - 5w_4) + T(\mu + 3w_3 - 9w_4)$$

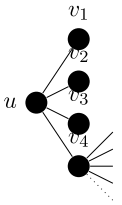
Comment: Consider the branch where v_4 is selected. A total of 6 vertices disappear and at least 3 vertices have their degree reduced from 4 to 3. We use the same argument for v_1, v_2 and v_3. Consider v_1.
If v_4 is not adjacent to v_1: the degree of v_4 is reduced.
If v_4 is adjacent to v_1 and $N[v_1] \not\subseteq N[v_4]$: a neighbor of v_1 has its degree reduced from 4 to 3.
If v_4 is adjacent to v_1 and $N[v_1] \subseteq N[v_4]$: Let y_1 and y_2 be the two common neighbors of v_1 and v_4 (except u). y_1 and y_2 have degree 4 and neighbors of degree 4, 4, 4 and 5. At least one of y_1 and y_2 has a neighbor of degree 4 outside $N[v_4]$, otherwise $N[y_1] = N[y_2]$.

22) $d(u) = 4, d(v_3) = 5, d(v_4) = 5$



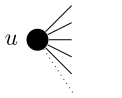
$$T(\mu) \leq 3T(\mu - 5w_4) + 2T(\mu - 6w_4)$$

23) $d(u) = 4, d(v_4) \geq 6$



$$T(\mu) \leq 4T(\mu - 5w_4) + T(\mu - 7w_4)$$

24) $d(u) \geq 5$



$$T(\mu) \leq 6T(\mu - 6w_4)$$

4.5 Count all Maximal Independent Sets in a Marked Graph of Maximum Degree Two

On input graphs of maximum degree 2 the algorithm `#MaximalIS` has an exponential worst-case running time. We show in this subsection, that all maximal independent sets of a marked graph of maximum degree 2 can be counted in polynomial time. Adding this polynomial time procedure to `#MaximalIS` is likely to be of help in implementations of the algorithm; it does however not improve its worst case running time.

Suppose first that G is a path $P_n = (v_1, v_2, \ldots, v_n)$. Let $V_i = \{v_1, v_2, \ldots, v_i\}$ for $i = 1, \ldots, n$. We define three values for the vertices of G with the following meaning:

- $is(v_i)$—the number of maximal independent sets of $G[V_i]$ containing v_i
- $od(v_i)$—the number of maximal independent sets of $G[V_{i-1}]$ containing v_{i-1}
- $ond(v_i)$—the number of maximal independent sets of $G[V_{i-1}]$ not containing v_{i-1}

The algorithm gives the following values to v_1:

- $is(v_1) = 0$ if v_1 is marked, and 1 otherwise,
- $od(v_1) = 0$, and
- $ond(v_1) = 1$.

Suppose the values for v_{i-1} are known, then the values for v_i are computed by simple dynamic programming as follows:

- $is(v_i) = 0$ if v_i is marked, and $od(v_{i-1}) + ond(v_{i-1})$ otherwise,
- $od(v_i) = is(v_{i-1})$, and
- $ond(v_i) = od(v_{i-1})$.

The number of maximal independent sets of G satisfying property Π (defined in Subsect. 4.1) of G is $is(v_n) + od(v_n)$.

If G is a cycle C_n, select an arbitrary vertex v_i with neighbors v_{i-1} and v_{i+1} and return the sum of the number of maximal independent sets

- containing v_i if v_i is not marked, or 0 otherwise,
- containing v_{i-1} if v_{i-1} is not marked, or 0 otherwise, and
- containing v_{i+1} but not v_{i-1} if v_{i+1} is not marked, or 0 otherwise.

This can easily be done by 3 recursive calls on the instances $G \setminus N[v_i]$, $G \setminus N[v_{i-1}]$ and $G \setminus N[v_{i+1}]$ and by marking v_{i-1} in the last recursive call.

Lemma 6 *Let G be a marked graph with maximum degree 2. The number of maximal independent sets of G satisfying property Π can be computed in linear time.*

Remark 1 As $od(v_i) = is(v_{i-1})$, the value $od(\cdot)$ is redundant. But the above description makes it easier to see that a slight generalization of this algorithm, which is very similar to the algorithm in [1], makes it possible to count all maximal independent sets of a marked graph satisfying property Π in time $3^k n^{O(1)}$ when a path decomposition of width k of the graph is known.

4.6 Lower Bound on the Running Time of the Algorithm

We do not know whether the current techniques to analyze the running time of branching algorithms establish the worst-case running time (up to a polynomial factor). Even Measure & Conquer provides only upper bounds of the running time, but it is not known how far this upper bound might be from the (usually unknown) worst-case running time of the algorithm. Therefore a lower bound for the worst case running time of branching algorithms is desirable (see for example [13]). Here we establish a lower bound the running time of Algorithm `#MaximalIS`.

Theorem 7 *There exists an infinite family of graphs for which Algorithm* `#MaximalIS` *runs in time* $\Omega(1.3247^n)$. *Thus its worst case running time is* $\Omega(1.3247^n)$.

Proof The lower bound for the running time of `#MaximalIS` established here uses the same family of graphs as the lower bound for an algorithm computing a minimum independent dominating set [17].

Consider the graph G_l of Fig. 2. 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 (that is 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.

The first levels of the corresponding search tree are depicted in Fig. 3. Unless the graph has at most 4 vertices, each application of branching rule (B) satisfies the

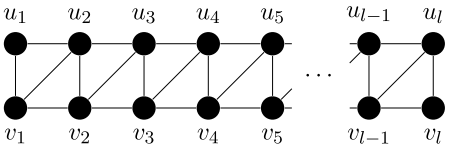


Fig. 2 Graph G_l used to lower bound the running time of Algorithm `#MaximalIS`

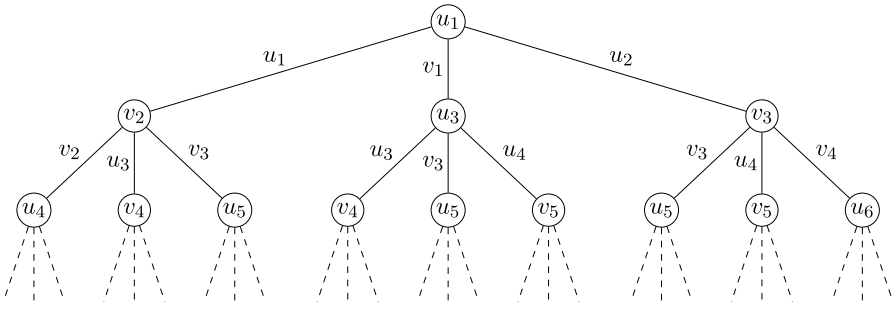


Fig. 3 A part of the search tree of the execution of Algorithm `#MaximalIS` on the graph G_l

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 $x^{-3} + x^{-4} + x^{-5} - 1$, that is $1.3247 < \alpha < 1.3248$. □

4.7 Algorithm to Count All Maximal Bicliques

Finally we show how to use the algorithm to count the maximal independent sets of a graph to establish an algorithm to count the maximal bicliques of a graph $G = (V, E)$.

We use the following polynomial-time Turing reduction of Dias et al. [9]. Let $G' = (V', E')$ be a copy of G. Let $G'' = (V'', E'')$ where $V'' = V \cup V'$ and $E'' = E \cup E' \cup \{xy' : x, y \in V, y' \text{ is a copy of } y \text{ in } V', \text{and } (x = y \text{ or } xy \notin E)\}$.

The following lemma is an immediate consequence of the 2–1 correspondence between the maximal cliques of the complement of G'' and the maximal bicliques of G shown by Dias et al. [9]. For the sake of completeness, a proof is provided.

Lemma 8 *The number of maximal independent sets 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 if and only if $X \cup Y'$, and thus also $Y \cup X'$, are maximal independent sets of G''. □

Using Lemma 8 and the algorithm to count the maximal independent sets of a graph, we establish an algorithm to count the maximal bicliques of a graph.

Theorem 9 *There is an algorithm to count the maximal bicliques of a graph in time $O(1.3642^n)$, where n is the number of vertices of the input graph.*

Proof The algorithm simply calls `#MaximalIS((`V''`,` $\emptyset$`,` E''`))` 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 $O(1.3642^n)$. □

5 Conclusion

We have seen in this paper that various results for independent sets translate to results for bicliques. The reverse questions are also interesting. For example, given an algorithm to find a maximum biclique in a graph of running time $O^*(c^n)$, is it possible to establish a $O^*(c^n)$ time algorithm for finding a maximum independent set in a graph?

Given a graph $G = (V, E)$ on n vertices, finding a maximum independent set in G could be done by constructing a graph G' obtained from G by adding an independent set I of size n such that every vertex of I is adjacent to every vertex of V. Then G has an independent set of size k if and only if G' has a biclique of size $n + k$. This shows that it is possible to obtain a $O^*(c^{2n})$ algorithm for computing a maximum independent set from an algorithm for computing a maximum biclique in a graph in time $O^*(c^n)$.

A simple variant of this reduction also shows that it is W[1]-hard to find an induced $K_{k,k}$, that is a biclique with k vertices in each part of its bipartition, in a graph, where the parameter is k (now only k independent vertices need to be added to G and made adjacent to every vertex in V). However the following question [8] about non-induced bicliques is still open.

Open Question Determine the parameterized complexity of the following problem: given a graph G and a parameter k, does G have a $K_{k,k}$ as a subgraph.

Acknowledgements The authors would like to thank the coordinating editor, Richard Cole, for his suggestion how to improve the upper bound of Theorem 3 from $n \cdot 3^{n/3}$ to $\frac{1}{3^{1/3}-1} \cdot 3^{n/3}$.

References

1. Alber, J., Niedermeier, R.: Improved tree decomposition based algorithms for domination-like problems. In: Proc. of LATIN 2002. LNCS, vol. 2286, pp. 613–627. Springer, Berlin (2002)
2. 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)
3. 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)
4. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion–exclusion. SIAM J. Comput. **39**, 546–563 (2009)
5. Dahllöf, V., Jonsson, P.: An algorithm for counting maximum weighted independent sets and its applications. In: Proc. of SODA 2002, pp. 292–298. ACM and SIAM, Philadelphia (2002)
6. Dahllöf, V., Jonsson, P., Wahlström, M.: Counting models for 2SAT and 3SAT formulae. Theor. Comput. Sci. **332**, 265–291 (2005)
7. Dawande, M., Swaminathan, J., Keskinocak, P., Tayur, S.: On bipartite and multipartite clique problems. J. Algorithms **41**, 388–403 (2001)
8. Demaine, E.D., Gutin, G., Marx, D., Stege, U.: Open Problems—Structure Theory and FPT Algorithmcs for Graphs, Digraphs and Hypergraphs. Dagstuhl Seminar Proceedings 07281 (2007), IBFI, Schloss Dagstuhl, Germany
9. Dias, V.M.F., Herrera de Figueiredo, C.M., Szwarcfiter, J.L.: Generating bicliques of a graph in lexicographic order. Theor. Comput. Sci. **337**, 240–248 (2005)
10. 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)
11. Fernau, H., Kneis, J., Kratsch, D., Langer, A., Liedloff, M., Raible, D., Rossmanith, P.: An exact algorithm for the maximum leaf spanning tree problem. In: Proc. of IWPEC 2009. LNCS, vol. 5917, pp. 161–172. Springer, Berlin (2009)

12. Fomin, F.V., Gaspers, S., Pyatkin, A.V., Razgon, I.: On the minimum feedback vertex set problem: Exact and enumeration algorithms. Algorithmica **52**, 293–307 (2008)
13. Fomin, F.V., Grandoni, F., Kratsch, D.: A measure & conquer approach for the analysis of exact algorithms. J. ACM **56** (2009)
14. Fürer, M., Kasiviswanathan, S.P.: Algorithms for counting 2-SAT solutions and colorings with applications. In: Proc. of AAIM 2007. LNCS, vol. 4508, pp. 47–57. Springer, Berlin (2007)
15. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical Foundations. Springer, Berlin (1996)
16. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-completeness. Freeman, New York (1979)
17. Gaspers, S., Liedloff, M.: A branch-and-reduce algorithm for finding a minimum independent dominating set. ArXiv Report 1009.1381 [CoRR abs] (2010)
18. Gaspers, S., Kratsch, D., Liedloff, M.: On independent sets and bicliques in graphs. In: Proc. of WG 2008. LNCS, vol. 5344, pp. 171–182. Springer, Berlin (2008)
19. Hochbaum, D.S.: Approximating clique and biclique problems. J. Algorithms **29**, 174–200 (1998)
20. Kneis, J., Langer, A., Rossmanith, P.: A fine-grained analysis of a simple independent set algorithm. In: Proc. of FSTTCS 2009. LIPICS, vol. 4, pp. 287–298, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Germany
21. Makino, K., Uno, T.: New algorithms for enumerating all maximal cliques. In: Proc. of SWAT 2004. LNCS, vol. 3111, pp. 260–272. Springer, Berlin (2004)
22. Moon, J.W., Moser, L.: On cliques in graphs. Isr. J. Math. **3**, 23–28 (1965)
23. Nourine, L., Raynaud, O.: A fast algorithm for building lattices. Inf. Process. Lett. **71**, 199–204 (1999)
24. Nourine, L., Raynaud, O.: A fast incremental algorithm for building lattices. J. Exp. Theor. Artif. Intell. **14**, 217–227 (2002)
25. Okamoto, Y., Uno, T., Uehara, R.: Counting the number of independent sets in chordal graphs. J. Discrete Algorithms **6**, 229–242 (2008)
26. Peeters, R.: The maximum edge biclique problem is NP-complete. Discrete Appl. Math. **131**, 651–654 (2003)
27. Prisner, E.: Bicliques in graphs I: Bounds on their number. Combinatorica **20**, 109–117 (2000)
28. Robson, J.M.: Algorithms for maximum independent sets. J. Algorithms **7**, 425–440 (1986)
29. van Rooij, J.M.M., Bodlaender, H.L.: Design by measure and conquer: a faster exact algorithm for dominating set. In: Proc. of STACS 2008. LIPIcs, pp. 657–668. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Germany
30. van Rooij, J.M.M., Nederlof, J., van Dijk, T.C.: Inclusion/exclusion meets measure and conquer. In: Proc. of ESA 2009. LNCS, vol. 5757, pp. 554–565. Springer, Berlin (2009)
31. Wahlström, M.: A tighter bound for counting max-weight solutions to 2SAT instances. In: Proc. of IWPEC 2008. LNCS, vol. 5018, pp. 202–213. Springer, Berlin (2008)
32. Yannakakis, M.: Node and edge deletion NP-complete problems. In: Proc. of STOC 1978, pp. 253–264. ACM, New York (1978)