Algorithmic aspects of counting independent sets

Min-Jen Jou
Ling Tong College
Tai Chung, Taiwan

Gerard J. Chang*
Department of Applied Mathematics
National Chiao Tung University
Hsinchu 30050, Taiwan
Email: gjchang@math.nctu.edu.tw

ABSTRACT. This paper studied the problems of counting independent sets, maximal independent sets, and maximum independent sets of a graph from an algorithmic point of view. In particular, we present linear-time algorithms for these problems in trees and unicyclic graphs.

1 Introduction

All graphs in this paper are simple, i.e., finite, undirected, loopless, and without multiple edges. In a graph G, an *independent set* is a subset S of $V(G)$ such that no two vertices in S are adjacent. A *maximal independent set* is an independent set that is not a proper subset of any other independent set. A *maximum independent set* is an independent set of maximum size. Note that a maximum independent set is maximal, but the converse is not always true.

Erdős and Moser raised the problem of determining the largest number of maximal independent sets in (the complement of) a general graph of order n and those graphs achieving the maximum value. This problem was solved by Erdős, and later Moon and Moser [21]. It then has been extensively studied for various classes of graphs, including trees [3, 7, 20, 22, 23], forests [14], (connected) graphs with at most one cycle [14], bipartite

*Supported in part by the National Science Council under grant NSC89-2115-M009-037 and the Lee and MTI Center for Networking Research at NCTU.

graphs [18, 19], connected graphs [6, 8], k-connected graphs [4], triangle-free graphs [11] and connected triangle-free graphs [1]; for a survey see [13]. Upper bounds for the number of maximum independent sets were studied in [15, 24]. Chang and Yeh [2] gave an algorithm for counting the number of maximum independent sets of a functional graph.

The purpose of this paper is to study the problems of counting independent sets, maximal independent sets, and maximum independent sets from an algorithmic points of view. In particular, we present linear-time algorithms for these problems in trees and unicyclic graphs. In the rest of this section, we fix some notation.

Denote by $I(G)$ the set of all independent sets of a graph G. For a vertex x, let $I_x(G) = \{S \in I(G) \colon x \in S\}$ and $I_{-x}(G) = \{S \in I(G) \colon x \notin S\}$. The cardinalities of $I(G)$, $I_x(G)$ and $I_{-x}(G)$ are denoted by $i(G)$, $i_x(G)$ and $i_{-x}(G)$, respectively. It is clear that $i(G) = i_{-x}(G) + i_x(G)$.

The set of all maximal independent sets of G is denoted by $\mathrm{MI}(G)$. For a vertex x, let $\mathrm{MI}_x(G) = \{S \in \mathrm{MI}(G) \colon x \in S\}$ and $\mathrm{MI}_{-x}(G) = \{S \in \mathrm{MI}(G) \colon x \notin S\}$. The cardinalities of $\mathrm{MI}(G)$, $\mathrm{MI}_x(G)$ and $\mathrm{MI}_{-x}(G)$ are denoted by $\mathrm{mi}(G)$, $\mathrm{mi}_x(G)$ and $\mathrm{mi}_{-x}(G)$, respectively. It is clear that $\mathrm{mi}(G) = \mathrm{mi}_{-x}(G) + \mathrm{mi}_x(G)$.

The set of all maximum independent sets of G is denoted by $\mathrm{XI}(G)$. For a vertex x, let $\mathrm{XI}_x(G) = \{S \in \mathrm{XI}(G) \colon x \in S\}$ and $\mathrm{XI}_{-x}(G) = \{S \in \mathrm{XI}(G) \colon x \notin S\}$. The cardinalities of $\mathrm{XI}(G)$, $\mathrm{XI}_x(G)$ and $\mathrm{XI}_{-x}(G)$ are denoted by $\mathrm{xi}(G)$, $\mathrm{xi}_x(G)$ and $\mathrm{xi}_{-x}(G)$, respectively. It is clear that $\mathrm{xi}(G) = \mathrm{xi}_{-x}(G) + \mathrm{xi}_x(G)$.

In a graph G, the *neighborhood* $N_G(x)$ of a vertex x is the set of vertices adjacent to x, and the *closed neighborhood* $N_G[x]$ is $N_G(x) \cup \{x\}$. A vertex x is a *leaf* if $|N_G(x)| = 1$. The *deletion* of a subset $U \subseteq V(G)$ from G is the graph $G - U$ obtained from G by removing all vertices in U and all edges incident to these vertices.

2 Independent sets

This section presents linear-time algorithms for computing $i(T)$ of a tree T and $i(H)$ of a unicyclic graph H. For technical reasons, we investigate the following weighted version of the problem. In a graph G, suppose c and d are two functions from $V(G)$ to the set $\mathbb{N}$ of all positive integers. Define

$$i(G,c,d) = \sum_{S\in I(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right),$$

$$i_x(G,c,d) = \sum_{S\in I_x(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \quad \text{for } x \in V(G),$$

$$i_{-x}(G,c,d) = \sum_{S\in I_{-x}(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \quad \text{for } x \in V(G).$$

We can interpret the functions c and d as follows: c_z (respectively, d_y) is the weight when vertex z is *contained* (respectively, y is *not contained*) in an independent set S. When we count the number of independent sets of G, an independent set S contributes $\prod_{z\in S} c_z \prod_{y\notin S} d_y$ copies of itself. Thus, the total number of *weighted copies* of independent sets is $i(G,c,d)$. It is obvious that if $c_v = d_v = 1$ for each vertex v in G, then $i(G,c,d) = i(G)$, $i_x(G,c,d) = i_x(G)$ and $i_{-x}(G,c,d) = i_{-x}(G)$ for any vertex x.

The following theorems are the base of the algorithms for computing $i(T,c,d)$ of a tree T and $i(H,c,d)$ of a unicyclic graph H.

Theorem 2.1. *For any vertex v in G,*

$$i(G,c,d) = c_v \left(\prod_{y\in N_G(v)} d_y\right) i(G - N_G[v], c, d) + d_v i(G - v, c, d).$$

Proof: By definition, we have

$$
\begin{aligned}
c_v & \left(\prod_{y\in N_G(v)} d_v\right) i(G - N_G[v], c, d) + d_v i(G - v, c, d)) \\
&= c_v \left(\prod_{y\in N_G(v)} d_v\right) \sum_{S\in I(G-N_G[v])} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&\quad + d_v \sum_{S\in I(G-v)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&= \sum_{S\in I_v(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) + \sum_{S\in I_{-v}(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&= \sum_{S\in I(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&= i(G, c, d).
\end{aligned}
$$

□

Theorem 2.2. *If v is a leaf adjacent to x in G, then $i(G, c, d) = i(G - v, c', d')$, where*

$$
c'_z = \begin{cases} c_z, & \text{if } z \neq x, \\ c_x d_v, & \text{if } z = x; \end{cases}
$$

$$
d'_y = \begin{cases} d_y, & \text{if } y \neq x, \\ d_x(d_v + c_v), & \text{if } y = x. \end{cases}
$$

Proof: Since $(I_x(G)\cap I_{-v}(G))\cup(I_{-x}(G)\cap I_{-v}(G)) = I_{-v}(G)$ and $I_{-x}(G)\cap I_v(G) = I_v(G)$, we have

$$
\begin{aligned}
&i(G-v,c',d') \\
&\quad = i_x(G-v,c',d') + i_{-x}(G-v,c',d') \\
&\quad = \sum_{S\in I_x(G-v)} \left(\prod_{z\in S} c'_z\right)\left(\prod_{y\notin S} d'_y\right) + \sum_{S\in I_{-x}(G-v)} \left(\prod_{z\in S} c'_z\right)\left(\prod_{y\notin S} d'_y\right) \\
&\quad = \sum_{S\in I_x(G-v)} \left(\prod_{z\in S, z\neq x} c'_z\right) c'_x \left(\prod_{y\notin S} d'_y\right) \\
&\qquad + \sum_{S\in I_{-x}(G-v)} \left(\prod_{z\in S} c'_z\right)\left(\prod_{y\notin S, y\neq x} d'_y\right) d'_x \\
&\quad = \sum_{S\in I_x(G-v)} \left(\prod_{z\in S, z\neq x} c_z\right) c_x d_v \left(\prod_{y\notin S} d_y\right) \\
&\qquad + \sum_{S\in I_{-x}(G-v)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S, y\neq x} d_y\right) d_x(d_v+c_v) \\
&\quad = \sum_{S\in I_x(G)\cap I_{-v}(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&\qquad + \sum_{S\in I_{-x}(G)\cap I_{-v}(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&\qquad + \sum_{S\in I_{-x}(G)\cap I_{v}(G)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&\quad = i_{-v}(G,c,d) + i_v(G,c,d) \\
&\quad = i(G,c,d).
\end{aligned}
$$

□

We first give a linear-time algorithm for computing $i(T,c,d)$ for a tree T.

Algorithm 2.3 *Compute $i(T,c,d)$ for a tree T.*

$G \leftarrow T$;
for (G has more than one vertex) **do**
  choose a leaf v adjacent to x in G;
  $c_x \leftarrow c_x d_v$;
  $d_x \leftarrow d_x(d_v + c_v)$;
  $G \leftarrow G - v$;

end do;
$i(T,c,d) \leftarrow d_v + c_v$ where v is the only vertex in G.

We then have the following linear-time algorithm for computing $i(H,c,d)$ for a unicyclic graph H.

Algorithm 2.4 *Compute $i(H,c,d)$ for a unicyclic graph H.*

$G \leftarrow H$;
while (G is not a cycle) **do**
  choose a leaf x adjacent to v in G;
  $c_x \leftarrow c_x d_v$;
  $d_x \leftarrow d_x(d_v + c_v)$;
  $G \leftarrow G - v$;
end do;
choose a vertex x adjacent to y and z in G;
$i(H) \leftarrow c_x d_y d_z i(G - \{x,y,z\}, c, d) + d_x i(G - x, c, d)$.
/* apply Algorithm 2.3 to $G - \{x,y,z\}$ and $G - x$. */

3 Maximum independent sets

This section gives linear-time algorithms for finding $\mathrm{xi}(T)$ for a tree T and $\mathrm{xi}(H)$ for a unicyclic graph H. As in Section 2, we consider a weighted version of the problem as follows. In a graph G, suppose w is a function from $V(G)$ to the set $\mathbb{Z}$ of all integers, and c and d are two functions from $V(G)$ to $\mathbb{N}$. For any subset S of $V(G)$, let $w(S) = \sum_{x\in S} w_x$. The *w-stability number* of G is $\alpha(G,w) = \max_{S\in I(G)} w(S)$. Denote by $\mathrm{XI}(G,w)$ the set of all independent sets S with $w(S) = \alpha(G,w)$. For every vertex x, let $\mathrm{XI}_x(G,w) = \{S \in \mathrm{XI}(G,w) \colon x \in S\}$ and $\mathrm{XI}_{-x}(G,w) = \{S \in \mathrm{XI}(G,w) \colon x \notin S\}$. If $w_v = 1$ for any vertex v in G, then $\alpha(G,w) = \alpha(G)$, $\mathrm{XI}(G,w) = \mathrm{XI}(G)$, $\mathrm{XI}_x(G,w) = \mathrm{XI}_x(G)$ and $\mathrm{XI}_{-x}(G,w) = \mathrm{XI}_{-x}(G)$. Define

$$\mathrm{xi}(G,w,c,d) = \sum_{S\in \mathrm{XI}(G,w)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right),$$

$$\mathrm{xi}_x(G,w,c,d) = \sum_{S\in \mathrm{XI}(G,w)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \quad \text{for } x \in V(G),$$

$$\mathrm{xi}_{-x}(G,w,c,d) = \sum_{S\in \mathrm{XI}_{-x}(G,w)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \quad \text{for } x \in V(G).$$

It is clear that if $w_v = c_v = d_v = 1$ for each vertex v in G, then $\mathrm{xi}(G,w,c,d) = \mathrm{xi}(G)$, $\mathrm{xi}_x(G,w,c,d) = \mathrm{xi}_x(G)$ and $\mathrm{xi}_{-x}(G,w,c,d) = \mathrm{xi}_{-x}(G)$ for any vertex x.

Similar to Theorem 2.1 we have

Theorem 3.1. *For any vertex v, let $G_1 = G - N_G[v]$, $G_2 = G - v$, $\alpha_1 = \alpha(G_1, w) + \max\{0, w_v\}$ and $\alpha_2 = \alpha(G_2)$. Then $\alpha(G, w) = \max\{\alpha_1, \alpha_2\}$ and*

$$xi(G,c,d) = \begin{cases} c_v \left(\prod_{y \in N_G(v)} d_y\right) xi(G_1, c, d), & \text{if } \alpha_1 > \alpha_2, \\ d_v xi(G_2, c, d), & \text{if } \alpha_1 < \alpha_2, \\ c_v \left(\prod_{y \in N_G(v)} d_y\right) xi(G_1, c, d) + d_v xi(G_2, c, d), & \text{if } \alpha_1 = \alpha_2. \end{cases}$$

For any family $\mathcal{F}$ of sets and any element x, $\mathcal{F} + x$ denotes $\{S \cup \{x\} : S \in \mathcal{F}\}$.

Theorem 3.2. *If v is a leaf adjacent to x in G, then $\alpha(G, w) = \alpha(G - v, w') + \max\{0, w_v\}$ and $xi(G, w, c, d) = xi(G - v, w', c', d')$, where*

$$w'_z = \begin{cases} w_z, & \text{if } z \neq x, \\ w_x - \max\{0, w_v\}, & \text{if } z = x; \end{cases}$$

$$c'_z = \begin{cases} c_z, & \text{if } z \neq x, \\ c_x d_v, & \text{if } z = x; \end{cases}$$

$$d'_y = \begin{cases} d_y, & \text{if } y \neq x, \\ d_x d_v, & \text{if } y = x \text{ and } w_v < 0, \\ d_x c_v, & \text{if } y = x \text{ and } W_v > 0, \\ d_x(c_v + d_v), & \text{if } y = x \text{ and } w_v = 0. \end{cases}$$

Proof: We first prove the following three facts.

(1) $\alpha(G, w) = \alpha(G - v, w') + \max\{0, w_v\}$.

(2) $\mathrm{XI}_x(G, w) = \mathrm{XI}_x(G - v, w')$.

$$(3)\ \mathrm{XI}_{-x}(G, w) = \begin{cases} \mathrm{XI}_{-x}(G - v, w'), & \text{if } w_v < 0, \\ \mathrm{XI}_{-x}(G - v, w') + v, & \text{if } w_v > 0, \\ \mathrm{XI}_{-x}(G - v, w') \cup (\mathrm{XI}_{-x}(G - v, w') + v), & \text{if } w_v = 0. \end{cases}$$

Choose two sets $S \in \mathrm{XI}_x(G, w)$ and $S' \in \mathrm{XI}_x(G - v, w')$. Note that $S' \in I(G)$ and $S \in I(G - v)$. Therefore,

$$\begin{aligned} \alpha(G - v, w') &= w'(S') = w'(S' - \{x\}) + w'_x \\ &= w(S' - \{x\}) + w_x - \max\{0, w_v\} = w(S') - \max\{0, w_v\} \\ &\le \alpha(G, w) - \max\{0, w_v\} = w(S) - \max\{0, w_v\} \\ &= w'(S - \{x\}) + w'_x = w'(S) \le \alpha(G - v, w'). \end{aligned}$$

Thus, all inequalities are equalities. Hence, (1) holds. Also, $w'(S) = \alpha(G - v, w')$ and $w(S') = \alpha(G, w)$ imply $S \in \mathrm{XI}_x(G - v, w')$ and $S' \in \mathrm{XI}_x(G, w)$. So, (2) holds.

Next, choose $S \in \mathrm{XI}_{-x}(G, w)$ and $S' \in \mathrm{XI}_{-x}(G - v, w')$. Note that $S' \in I(G)$. For the case in which $w_v < 0$, $v \notin S$ and so $S \in I(G - v)$; otherwise, $v \in S$ would imply $\alpha(G, w) = w(S) < w(S - \{v\})$, a contradiction. Then

$$\alpha(G, w) = w(S) = w'(S) \leq \alpha(G - v, w') = w'(S') = w(S') \leq \alpha(G, w).$$

Thus, all inequalities are equalities, which implies that (1) and (3) hold. If $w_v > 0$, $v \in S$ and so $S - \{v\} \in I(G - v)$; otherwise, $v \notin S$ would imply $\alpha(G, w) = w(S) < w(S \cup \{v\})$, a contradiction. Then

$$\begin{aligned} \alpha(G, w) = w(S) &= w'(S - \{v\}) + w_v \leq \alpha(G - v, w') + w_v \\ &= w'(S') + w_v = w(S' \cup \{v\}) \leq \alpha(G, w). \end{aligned}$$

Again, all inequalities are equalities, which implies that (1) and (3) hold. For the case in which $w_v = 0$, either $v \notin S$ or $v \in S$. Exactly the same arguments for the above two cases imply (1) and (3).

We then have:

$$\begin{aligned} &\mathrm{xi}_x(G - v, w', c', d') \\ &= \sum_{S \in \mathrm{XI}_x(G - v, w')} \left(\prod_{z \in S} c'_z\right) \left(\prod_{y \notin S} d'_y\right) \\ &= \sum_{S \in \mathrm{XI}_x(G - v, w')} \left(\prod_{z \in S, z \neq x} c'_z\right) c'_x \left(\prod_{y \notin S} d'_y\right) \\ &= \sum_{S \in \mathrm{XI}_x(G - v, w')} \left(\prod_{z \in S, z \neq x} c_z\right) c_x d_v \left(\prod_{y \notin S} d_y\right) \\ &= \sum_{S \in \mathrm{XI}_x(G - v, w)} \left(\prod_{z \in S, z \neq x} c_z\right) c_x \left(\prod_{y \notin S, y \neq v} d_y\right) d_v \\ &= \sum_{S \in \mathrm{XI}_x(G, w)} \left(\prod_{z \in S} c_z\right) \left(\prod_{y \notin S} d_y\right) \\ &= \mathrm{xi}_x(G, w, c, d). \end{aligned}$$

$$
\begin{aligned}
&\mathrm{xi}_{-x}(G-v,w',c',d') \\
&= \sum_{S\in \mathrm{XI}_{-x}(G-v,w')} \left(\prod_{z\in S} c'_z\right)\left(\prod_{y\notin S} d'_y\right) \\
&= \sum_{S\in \mathrm{XI}_{-x}(G-v,w')} \left(\prod_{z\in S} c'_z\right)\left(\prod_{y\notin S, y\neq x} d'_y\right) d'_x \\
&= \begin{cases}
\sum_{S\in \mathrm{XI}_{-x}(G-v,w')} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S,y\neq x} d_y\right) d_x d_v, & \text{if } w_v < 0,\\
\sum_{S\in \mathrm{XI}_{-x}(G-v,w')} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S,y\neq x} d_y\right) d_x c_v, & \text{if } w_v > 0,\\
\sum_{S\in \mathrm{XI}_{-x}(G-v,w')} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S,y\neq x} d_y\right) d_x (c_v+d_v), & \text{if } w_v = 0,
\end{cases} \\
&= \begin{cases}
\sum_{S\in \mathrm{XI}_{-x}(G,w)\cap \mathrm{XI}_{-v}(G,w)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S,y\neq x,y\neq v} d_y\right) d_x d_v, & \text{if } w_v < 0,\\
\sum_{S\in \mathrm{XI}_{-x}(G,w)\cap \mathrm{XI}_{v}(G,w)} \left(\prod_{z\in S,z\neq v} c_z\right) c_v \left(\prod_{y\notin S,y\neq x} d_z\right) d_x, & \text{if } w_v > 0,\\
\sum_{S\in \mathrm{XI}_{-x}(G,w)\cap \mathrm{XI}_{v}(G,w)} \left(\prod_{z\in S,z\neq v} c_z\right) c_v \left(\prod_{y\notin S,y\neq x} d_y\right) d_x + & \\
\sum_{S\in \mathrm{XI}_{-x}(G,w)\cap \mathrm{XI}_{-v}(G,w)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S,y\neq x,y\neq v} d_y\right) d_x d_v, & \text{if } w_v = 0,
\end{cases} \\
&= \sum_{S\in \mathrm{XI}_{-x}(G,w)} \left(\prod_{z\in S} c_z\right)\left(\prod_{y\notin S} d_y\right) \\
&= \mathrm{xi}_{-x}(G,w,c,d).
\end{aligned}
$$

Therefore, $\mathrm{xi}(G,w,c,d) = \mathrm{xi}_x(G,w,c,d) + \mathrm{xi}_{-x}(G,w,c,d) = \mathrm{xi}_x(G-v,w',c',d') + \mathrm{xi}_{-x}(G-v,w',c',d') = \mathrm{xi}(G-v,w',c',d')$. □

We first present a linear-time algorithm for finding $\mathrm{xi}(T,w,c,d)$ for a tree T.

Algorithm 3.3 *Compute $\alpha(T,w)$ and $\mathrm{xi}(T,w,c,d)$ for a tree T.*

$G \leftarrow T$;
$\alpha \leftarrow 0$;
while (G has more than one vertex) **do**
  choose a leaf v adjacent to x in G;
  $w_x \leftarrow w_x - \max\{0, w_v\}$;
  $c_x \leftarrow c_x d_v$;
  if $w_v < 0$ **then** $d_x \leftarrow d_x d_v$;
  if $w_v > 0$ **then** $d_x \leftarrow d_x c_v$;
  if $w_{v_i} = 0$ **then** $d_x \leftarrow d_x(c_v + d_v)$;
  $\alpha \leftarrow \alpha + \max\{0, w_v\}$;
  $G \leftarrow G - v$;
end do;
assume v is the only vertex of G;

$\alpha(T, w) \leftarrow \alpha + \max\{0, w_v\}$;
if $w_v < 0$ **then** $\mathrm{xi}(T, w, c, d) \leftarrow d_v$;
if $w_v > 0$ **then** $\mathrm{xi}(T, w, c, d) \leftarrow c_v$;
if $w_v = 0$ **then** $\mathrm{xi}(T, w, c, d) \leftarrow c_v + d_v$.

Next we give a linear-time algorithm for finding $\mathrm{xi}(H, w, c, d)$ for a unicyclic graph H.

Algorithm 3.4 *Compute* $\alpha(G, w)$ *and* $\mathrm{xi}(H, w, c, d)$ *for a unicyclic graph* H.

```
G ← H;
α ← 0;
while (G is not a cycle) do
    choose a leaf v adjacent to x in G;
    w_x ← w_x − max{0, w_v};
    c_x ← c_x d_v;
    if w_v < 0 then d_x ← d_x d_v;
    if w_v > 0 then d_x ← d_x c_v;
    if w_v = 0 then d_x ← d_x(c_v + d_v);
    α ← α + max{0, w_v};
    G ← G − v;
end do;
choose a vertex x adjacent to y and z in G;
G_1 ← G − {x, y, z};
G_2 ← G − x;
α_1 ← α(G_1, w) + max{0, w_x}; /* apply Algorithm 3.3 to G_1 */
α_2 ← α(G_2, w); /* apply Algorithm 3.3 to G_2 */
if α_1 > α_2 then xi(H, w, c, d) ← c_x d_y d_z xi(G_1, w, c, d);
if α_1 < α_2 then xi(H, w, c, d) ← d_x xi(G_2, w, c, d);
if α_1 = α_2 then xi(H, w, c, d) ← c_x d_v d_z xi(G_1, w, c, d) + d_x xi(G_2, w, c, d).
```

4 Maximal independent sets

This section gives linear-time algorithms for counting the numbers of maximal in dependent sets of a tree and of a unicyclic graph by using a dynamic programming approach.

The following theorem is easy to see. We use $\mathrm{mi}_{0v}(G)$ for $\mathrm{mi}(G - v)$.

Theorem 4.1. *Suppose x is a vertex in graph G_1 and v is a vertex in graph G_2. If G is the graph obtained from the disjoint union of G_1 and G_2 by adding a new edge xv, then*

(1) $mi_x(G) = mi_x(G_1) mi_{0v}(G_2)$;

(2) $mi_{-x}(G) = mi_{-x}(G_1) mi_{-v}(G_2) + mi_{0x}(G_1) mi_v(G_2)$;

(3) $mi_{0x}(G) = mi_{0x}(G_1)(mi_v(G_2) + mi_{-v}(G_2)$.

Based on this theorem, we have the following linear-time algorithm for computing mi(T) for a tree T.

Algorithm 4.2 *Compute* mi(T) *for a tree* T.

for (any vertex v in T) **do**
  $\mathrm{mi}_v \leftarrow 1,;\ \mathrm{mi}_{-v} \leftarrow 0;\ \mathrm{mi}_{0v} \leftarrow 1;$
end do;
$G \leftarrow T;$
while (G has more than one vertex) **do**
  choose a leaf v adjacent to x in G;
  $\mathrm{mi}_x \leftarrow \mathrm{mi}_x \mathrm{mi}_{0v};$
  $\mathrm{mi}_{-x} \leftarrow \mathrm{mi}_{-x}\mathrm{mi}_{-v} + \mathrm{mi}_{0x}\mathrm{mi}_v;$
  $\mathrm{mi}_{0x} \leftarrow \mathrm{mi}_{0x}(\mathrm{mi}_v + \mathrm{mi}_{-v});$
  $G \leftarrow G - v;$
end do;
$\mathrm{mi}(T) \leftarrow \mathrm{mi}_v + \mathrm{mi}_{-v}$, where v is the only vertex in G.

We then have the following linear-time algorithm for computing mi(H) for a unicyclic graph H.

Algorithm 4.3 *Compute* mi(H) *for a unicyclic graph* H.

for (any vertex v in H) **do**
  $\mathrm{mi}_v \leftarrow 1;\ \mathrm{mi}_{-v} \leftarrow 0;\ \mathrm{mi}_{0v} \leftarrow 1;$
end do;
$G \leftarrow H;$
while (G is not a cycle) **do**
  choose a leaf v adjacent to x in G;
  $\mathrm{mi}_x \leftarrow \mathrm{mi}_x \mathrm{mi}_{0v};$
  $\mathrm{mi}_{-x} \leftarrow \mathrm{mi}_{-x}\mathrm{mi}_{-v} + \mathrm{mi}_{0x}\mathrm{mi}_v;$
  $\mathrm{mi}_{0x} \leftarrow \mathrm{mi}_{0x}(\mathrm{mi}_v + \mathrm{mi}_{-v});$
  $G \leftarrow G - v;$
end do;
Suppose G is the cycle $\ldots, z', y', x, y, x, \ldots;$
$G_1 \leftarrow G - \{x, y, y'\};$
$G_2 \leftarrow G - \{x, y, z\};$
$G_3 \leftarrow G - \{x, y', z'\};$
$G_4 \leftarrow G - \{x, y, y', z, z'\};$
$\mathrm{mi}(H) \leftarrow \mathrm{mi}_x \mathrm{mi}_{0y} \mathrm{mi}_{0y'} \mathrm{mi}(G_1) + \mathrm{mi}_{0x}\mathrm{mi}_y \mathrm{mi}_{0z} \mathrm{mi}(G_2) +$
  $\mathrm{mi}_{0x}\mathrm{mi}_{y'}\mathrm{mi}_{0z'}\mathrm{mi}(G_3) - \mathrm{mi}_{0x}\mathrm{mi}_y \mathrm{mi}_{y'}\mathrm{mi}_{0z}\mathrm{mi}_{0z'}\mathrm{mi}(G_4).$
/* Apply Algorithm 4.2 to graphs G_1, G_2, G_3, G_4. */

References

[1] G.J. Chang and M.J. Jou, The number of maximal independent sets in connected triangle-free graphs, *Discrete Math.* **197/198** (1999), 169–178.

[2] S.C. Chang and Y.N. Yeh, The cardinality of the collection of maximum independent sets of a graph, *Advances in Applied Math.* **18** (1997), 286–299.

[3] D. Cohen, Counting stable sets in trees, in *Seminaire Lotharingien de Combinatoire*, 10*eme* session, R. König, ed., Institute de Recherche Mathématique Avancée Pub., Strasbourg, France (1984), 48–52.

[4] C. Fan and J. Liu, On clique of graphs, in *Graph Theory, Combinatorics, Algorithms and Applications*, Y. Alavi, F.R.K. Chung, R.L. Graham and D.F. Hsu, ed., SIAM, Philadelphia, PA (1991), 140–150.

[5] M. Farber, M. Hujter, and Z. Tuza, An upper bound on the number of cliques in a graph, *Networks* **23** (1993), 207–210.

[6] Z. Füredi, The number of maximal independent sets in connected graphs, *J. Graph Theory* **11** (1987), 463–470.

[7] J.R. Griggs and C.M. Grinstead, unpublished result (1986).

[8] J.R. Griggs, C.M. Grinstead, and D.R. Guichard, The number of maximal independent sets in a connected graph, *Discrete Math.* **68** (1988), 211–220.

[9] B. Hedman, The maximum number of cliques in dense graphs, *Discrete Math.* **54** (1985), 161–166.

[10] G. Hopkins and W. Staton, Graphs with unique maximum independent sets, *Discrete Math.* **57** (1985), 245–251.

[11] M. Hujter and Z. Tuza, The number of maximal independent sets in triangle-free graphs, *SIAM J. Discrete Math.* **6** (1993), 284–288.

[12] D.S. Johnson, M. Yannakakis, and C.H. Papadimitriou, On generating all maximal independent sets, *Inform. Process. Letters* **27** (1988), 119–123.

[13] M.J. Jou and G.J. Chang, Survey on counting maximal independent sets, in: *Proceedings of the Second Asian Mathematical Conference*, S. Tangmance and E. Schulz ed., World Scientific (1995), 265–275.

[14] M.J. Jou and G.J. Chang, Maximal independent sets in graphs with at most one cycle, *Discrete Applied Math.* **79** (1997), 67–73.

[15] M.J. Jou and G.J. Chang, The number of maximum independent sets in graphs, *Taiwanese J. Math.* **4** (2000), 685–695.

[16] M.J. Jou, G.J. Chang, C. Lin, and T.H. Ma, A finiteness theorem for maximal independent sets, *Graphs and Combin.* **12** (1996), 321–326.

[17] S.B. Lin and C. Lin, Trees and forests of the same order with large and small independent indices, *Chinese J. Math.* **32** (1995), 199–210.

[18] V. Linek, Bipartite graphs can have any number of independent sets, *Discrete Math.* **76** (1989), 131–136.

[19] J. Liu, Maximal independent sets in bipartite graphs, *J. Graph Theory* **17** (1993), 495–507.

[20] A. Meir and J.W. Moon, On maximal independent sets of nodes in trees, *J. Graph Theory* **12** (1988), 265–283.

[21] J.W. Moon and L. Moser, On cliques in graphs, *Israel J. Math.* **3** (1965), 23–28.

[22] B.E. Sagan, A note on independent sets in trees, *SIAM J. Discrete Math.* **1** (1988), 105–108.

[23] H.S. Wilf, The number of maximal independent sets in a tree, *SIAM J. Algebraic Discrete Methods* **7** (1986), 125–130.

[24] J. Zito, The structure and maximum number of maximum independent sets in trees, *J. Graph Theory* **15** (1991) 207–221.