Un algoritmo de tiempo lineal para el problema del Mínimo Conjunto Independiente a-Maximal restringido a grafos con treewidth acotada

Manrique Mata-Montero*
Virginia Berrón Lara**

Resumen

En este artículo describimos algunas variaciones del Problema Conjunto Independiente de Nodos en un Grafo y mostramos un algoritmo con complejidad de tiempo lineal para el problema del Mínimo Conjunto Independiente a-Maximal, cuando a = 1 y las instancias del problema se restringen a árboles. Luego, mostramos que aun cuando a las instancias se les permite variar sobre el conjunto de grafos con treewidth acotada por k, donde k >1 es una constante entera cualquiera, el problema todavía se puede resolver en tiempo lineal. El problema del Mínimo Conjunto Independiente a-Maximal es NP-difícil para instancias no restringidas.

Palabras clave: treewidth acotada, k-árboles parciales, árboles, conjunto independiente, algoritmos de tiempo lineal, programación dinámica.

Abstract

In this article we describe some variations of the Independent Set Problem of Nodes in a Graph and present an algorithm with linear time complexity for the problem of the Minimum Independent a-Maximal Independent Set, where a-1 and the instances of the problem are restricted to trees. We further show that even in cases where variation is allowed over the group of graphs with treewidth limited by k, where k > 1 is any whole number, the problem can still be solved in linear time. The problem of the Minimun a-Maximal Independent Set is NP- hard in unrestricted cases.

Keywords: limited treewidth, partial k-trees, independent set, linear time algorithms, dynamic programming.

Introducción

Los problemas del Conjunto Independiente Máximo y del Conjunto Independiente Maximal han jugado papeles importantes en la teoría de la complejidad. El primero es uno de los problemas NP-completos clásicos [11] y el segundo ha sido muy difícil de resolver en paralelo.

El Conjunto Independiente Maximal se resuelve trivialmente con un algoritmo secuencial tacaño en tiempo lineal, pero soluciones paralelas eficientes fueron difíciles de encontrar. Dadoun and Kirkpatrick [9] mostraron un algoritmo que requiere tiempo polilogarítmico y un número polinomial de procesadores, esto es, un algoritmo NC. Cuando las instancias del problema se restringen a grafos planares y Karp and Wigderson [12] dieron un algoritmo NC para instancias no restringidas[1]. Varias soluciones más simples se dieron después [13].

La versión maximal del problema ha recibido alguna atención, Cook [7] probó que el problema del Conjunto Independiente Maximal Lexicográficamente Primero es completo para P (el conjunto de problemas solubles en tiempo polinomial) y Sagan [18] mostró que el número de conjuntos maximales independientes en un árbol T de *n* vértices es:

$$m(T) = \begin{cases} 2^{k-1}+1 & \text{si } n = 2k \\ 2^{k} & \text{si } n = 2k+1 \end{cases}$$

En este artículo mostramos primero un algoritmo con tiempo lineal para resolver el Problema del Mínimo Conjunto Independiente a-Max restringido a árboles

* *Profesor-investigador de tiempo completo de la U.T.M*
** *Profesora-investigadora de tiempo completo de la U.T.M*

1. NC significa la clase de Nicholas.

con a = 1. Más tarde, desarrollaremos otro algoritmo con tiempo lineal (siguiendo esencialmente las mismas ideas usadas para el algoritmo para árboles), el cual admite como entrada cualquier grafo con treewidth acotada por *k*, donde *k* ³ 1 es cualquier constante entera, y a= 1. Este algoritmo usa la reducción del *k*-árbol parcial dado como entrada, esto no disminuye su generalidad ya que se sabe que las clases de *k*-árboles parciales y los grafos con treewidth acotada por *k* son las mismas [19].

1 El problema del conjunto independiente Mínimo a-Maximal (MIN a-MIS)

Bollobás et al. [5] introdujeron los conceptos de a-maximalidad y a-minimalidad de conjuntos independientes. Algunos resultados para cotas superiores e inferiores para estos parámetros, cuando las instancias se restringen a caminos y ciclos, fueron dados por Cockayne et al. [6]. Nosotros consideraremos el concepto de a-maximalidad y un problema de optimización relacionado. A continuación definimos lo que es un conjunto independiente, a-maximal independiente y el problema que consideraremos en este artículo: Problema del MÍNIMO CONJUNTO INDEPENDIENTE a-MAXIMAL.

Definición 1. *Sea G un grafo. Un conjunto V', donde V'* Í *V es un* conjunto independiente si *para todo par de vértices u, v* Î *V', la arista {u, v}* Ï *E.*

Definición 2. *Sea G un grafo. Para cualquier entero positivo* a, *X* Í *V(G) es un* CONJUNTO INDEPENDIENTE a-MAXIMAL (a-MIS) si:

- X es un conjunto independiente y,
- *no existen Y ó Z, donde Y* Í *X y Z* Í *X^c tal que*
- |Y| < a,
- |Z| > |Y| y
- *(X - Y)* È *Z es un conjunto independiente.*

La versión del problema de decisión del problema MIN a-MIS se define como sigue:

El Problema del MÍNIMO CONJUNTO INDEPENDIENTE a-MAXIMAL (MIN a-MIS)

Instancia: Sean G = (V, *E)* un grafo, l y a enteros positivos.

Pregunta: ¿Existe un a-MIS X tal que | X | < l ?

El problema MIN a-MIS es claramente NP-difícil, puesto que el problema CONJUNTO MÁXIMO INDEPENDIENTE, el cual es NP-completo, se reduce trivialmente en tiempo polinomial a éste. Cuando a = 1 el problema es NP-completo.

En la siguiente subsección examinaremos la complejidad del PROBLEMA MÍNIMO a-MIS en el caso cuando G es un árbol y a = 1. En la subsección que le sigue generalizaremos la estrategia usada para diseñar el algoritmo para árboles para construir otro algoritmo, el cual acepta como entrada un *k*-árbol parcial junto con su redución para cualquier entero constante k ³ 1 y a = 1

1.1 El problema MÍNIMO 1-MIS restringido a árboles

Sea *T* un árbol con raíz *r*, C el conjunto de descendientes inmediatos de *r*, y I Í V(T) un MIN 1-MIS en el árbol T. Sean *T[c]*, donde c Î C, árboles en el bosque obtenido borrando la raíz *r* de *T* y las aristas incidentes en ella. Considere cualquier subárbol *T[c]*, donde c Î C, y el conjunto I_c = IÇ *V(T[c])*. En general, I_c no es un conjunto independiente maximal en el árbol *T[c]*, puesto que el nodo c puede no estar en I_c ni ser adyacente a un nodo en I_c. El nodo c en relación a I_c presenta tres posibilidades:

- *c* está en I_c, o
- *c* no está en I_c y no es adyacente a ningún nodo en I_c, o
- *c* no está en I_c pero es adyacente a un nodo en I_c.

Note que el nodo c es el único nodo que puede pertenecer a la segunda categoría listada anteriormente, de otra manera, I_c no sería un MIN a-MIS en el árbol T. Vea la ilustración dada en la Figura 1.

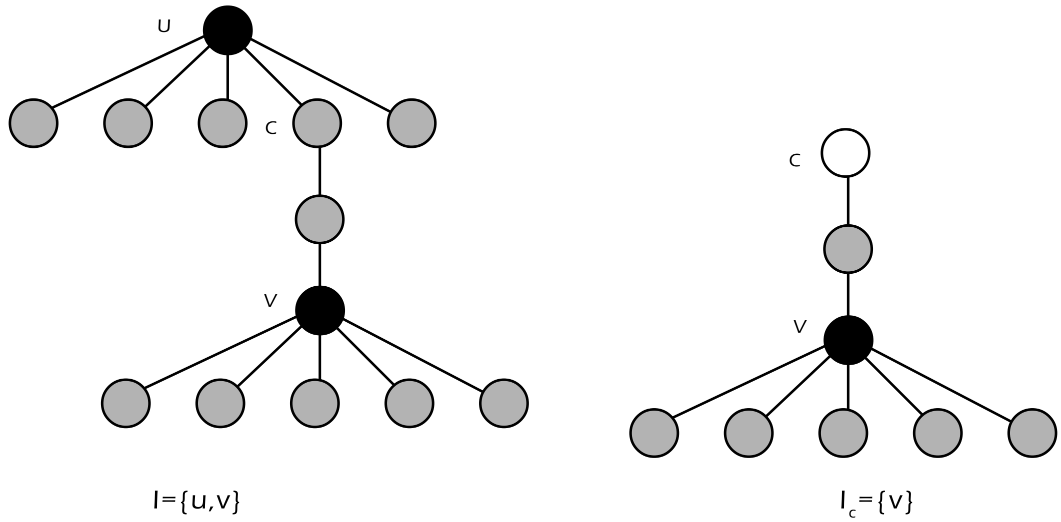


FIGURA 1: PARTICIÓN DE LOS NODOS DE UN ÁRBOL CON RESPECTO A UN MIN 1-MIS

Las afirmaciones 1, 2, y 3 dadas a continuación se siguen directamente de la definición de MIN a-MIS.

Afirmación 1. Si c Î I_c *entonces* I_c *es un* MIN 1-MIS *bajo la restricción de que este conjunto independiente incluya c.*

Afirmación 2. *Si c* Ï I_c *pero es adyacente a un nodo en* I_c*, entonces* I_c *es de nuevo un* MIN 1-MIS *bajo la restricción de que el conjunto independiente excluya a c.*

Afirmación 3. Si *c* Ï I_c *y no es adyacente a un nodo en* I_c*, entonces para cada nodo d, descendiente inmediato del nodo c, el conjunto I*Ç *V(T[d]) es un* MIN 1-MIS *en el árbol T[d], provisto que este conjunto se restrinja a excluir d.*

Se sigue de las afirmaciones 1, 2, y 3 que las soluciones al problema del MIN 1-MIS exhiben subestructura óptima [8]. Basados en esta observación desarrollamos un algoritmo de programación dinámica para resolver el problema.

Por conveniencia, definamos la siguiente partición de los nodos del árbol *T* con respecto a un conjunto independiente *I* en *T* que consta de los siguientes tres conjuntos:

- el conjunto de todos los nodos de *I*, cuyos elementos serán llamados nodos negros,
- el conjunto de todos los nodos que no están en *I* y que no son adyacentes a ningún nodo en *I*, a los que llamaremos nodos blancos y
- el conjunto de todos los nodos que no están en *I* y que son adyacentes a algún nodo en *I*, a los que llamaremos nodos grises.

Claramente, dada una partición de los nodos de un árbol *T* con respecto a un conjunto independiente, si el conjunto de nodos *blancos* no es vacío, tal conjunto independiente se puede aumentar con al menos uno de los nodos *blancos.* También se sigue que si el conjunto independiente es maximal, entonces el conjunto de los nodos *blancos* es vacío, i.e., todos los nodos de *V(T)* son *grises o negros.* Vea un ejemplo en la Figura 1.

Lema 1. *Sea T un árbol con raíz y sea I* Í *V(T) un* MIN 1-MIS *en T. Sea c cualquier nodo en V(T), T[c] el subárbol de T inducido por el nodo c y todos sus descendientes, desc(c), e* I_c =IÇ *V(T[c]). Sea D el conjunto de descendientes inmediatos del nodo c.*

- *Si* c Ï I_c *entonces* I_c *es un* MIN 1-MIS *en el subárbol T[c] bajo la restricción de que incluya al nodo c -el nodo c es* negro- y *todos los nodos d en D son* grises o blancos *con respecto a los conjuntos independientes* $I \cup$ *({d} U desc(d)) respectivamente, o*

- *si* $c \notin I_c$ *pero es adyacente a un nodo en* I_c *entonces el conjunto* I_c *es un* Min 1-MIS *en el subárbol T[c] bajo la restricción de que excluya al nodo c -el nodo c es* gris- y *todos los nodos en D son* negros o grises *-al menos uno de los cuales debe ser* negro- *con respecto a los conjuntos independientes* $I \cap$ *({d} U desc(d)) respectivamente, o*

- *si* $c \notin I_c$ *ni es adyacente a ningún nodo en* I_c *entonces* I_c *es un conjunto independiente donde c es el único nodo* blanco y *para cada* $d \in D$*, el conjunto* $I \cap V(T[d])$ *es un* Min 1-MIS, en *el subárbol T[d], bajo la restricción de que excluya el nodo d, y todos los nodos* $d \in D$ *son* grises *con respecto a los conjuntos independientes* $I \cap$ *({d} U desc(d)) respectivamente.*

Prueba

Se sigue por indución sobre el número de nodos en *T* y el hecho de que un Min 1-MIS en *T* exhibe subestructura óptima.

■

Se sigue del Lema 1 y del hecho que un Min 1-MIS tiene subestructura óptima, que para encontrar un Min 1-MIS en un árbol con raíz T es suficiente encontrar en cada subárbol inducido por los nodos $u \in V(T)$ y sus descendientes conjuntos independientes restringidos y de cardinalidad mínima. Para cada $u \in V(T)$, estos conjuntos independientes restringidos de cardinalidad mínima deberían incluir:

- un 1-MIS de cardinalidad mínima restringido a incluir el nodo *u*, i.e., donde *u* es *negro* y
- un 1-MIS de cardinalidad mínima restringido a excluir el nodo *u*, pero incluyendo al menos un descendiente del nodo *u*, i.e, donde *u* es *gris* y
- un conjunto independiente de cardinalidad mínima que no incluye a *u* ni a ninguno de sus sucesores inmediatos *d* pero tal que este conjunto independiente restringido a cada uno de los subárboles *T[d]* es un Min 1-MIS bajo la restricción de que excluya al nodo *d*, i.e., donde *u* es *blanco* y es el único nodo *blanco.*

Estos conjuntos independientes se pueden encontrar de *abajo hacia arriba,* desde las hojas de *T* hacia su raíz.

Un algoritmo para decidir si existe un Min 1-MIS de cardinalidad a lo más l en un árbol T se exhibe en la Figura 2. Este algoritmo se puede modificar fácilmente para encontrar los conjuntos independientes de interés. El algoritmo organiza el cómputo de los nodos de interés por turnos. Primeramente, estos conjuntos se computan para las hojas de *T*, luego para los predecesores inmediatos de las hojas y se continua de esa manera hasta alcanzar la raíz. Para guiar este proceso, definimos una *reducción* de un árbol, este es un ordenamiento de los nodos del árbol donde las hojas aparecen primero, luego los predecesores inmediatos de las hojas, y así sucesivamente.

Definición 3. Una reducción de un árbol *T* = (V, E), donde $|V| = n + 1$, es un ordenamiento $r_0, \ldots, r_n$ de V tal que para todo $i \in \{0, \ldots, n-1\}$, el vértice r_i tiene grado 1 en el grafo inducido por $\{r_i, \ldots, r_n\}$.

Afirmación 4. *Dado un árbol T, una de sus reduciones se puede encontrar en tiempo lineal, i.e., tiempo en* O(|V(T)|).

Prueba

Una simple variación de recorrido por *amplitud* del árbol se puede usar para producir su redución en tiempo lineal.

■

Teorema 1 *El algoritmo que se muestra en la Figura 2 para el 1-MIS restringido a árboles es correcto y tiene complejidad de tiempo en O(n).*

Prueba

Para $x \in$ *{negro, gris} y r* cualquier nodo en V(T), denotemos por $IS_r(x)$ la cardinalidad de un *Min 1-MIS* restringido al árbol inducido por *r y sus* descendientes, donde *r* es un nodo *x*.

```
    Algoritmo Min 1-MIS (T,X);
    ENTRADA: Un árbol T y un entero positivo l.
    SALIDA: Indicación acerca de la existencia de un conjunto MIN 1-MIS de cardinalidad a lo más l en T.
    R {r_0, . . .,r_n} una reducción de T, where r_0, . . ., r_j son las hojas de T
for r Î {r_0, . . ., r_n} do
    IS_r(b) ← 1
    IS_r(g) ← ¥
    IS_r(w) ← 0
endfor
for r Î {r_j+1 , ..., r_n} do
    D ← descendientes inmediatos de r
    IS_r(b) ← å_dÎD min{IS_d(w), IS_d(g)}+1
    IS_r(w) ← å_dÎD IS_d(g)
    if existe dÎ D tal que IS_d(b) £ IS_d(g) then
        IS_r(g) ← å_dÎD min{IS_d(b), IS_d(g)}
    else
        diferencia ← |V(T)| + 1
        for dÎ D do
            if IS_d(b)- IS_d (g) < diferencia then
                differencia ← IS_d (b) - IS_d(g)
                d_0 ← d
            endif
    endfor
        IS_r(g) ← å_d0¹dÎD IS_d(g) + IS_d0 (b)
    endif
endfor
if min{IS_rn (b), IS_rn (g)} £ l then
    output (' Existe un MIN 1-MIS de cardinalidad a lo más l en T ')
else
    output (' No existe un Min 1-MIS de cardinalidad a lo más l en T ')
endif
end
```

FIGURA 2: ALGORITMO PARA DECIDIR SI EXISTE UN MIN 1-MIS DE CARDINALIDAD A LO MÁS l EN UN ÁRBOL DADO.

Denotemos por $IS_r(blanco)$ la cardinalidad de un conjunto independiente de nodos de cardinalidad mínima donde el único nodo *blanco* es *r*.

El Lema 1 implica que las expresiones usadas en el algoritmo para $IS_r(n)$ e $IS_r(b)$ están bien definidas. Para justificar la expresión para $IS_r(g)$, lo único que queda por justificar es la elección del conjunto representado por $IS_d(n)$, i.e., el descendiente *negro* de *r,* cuando ninguno de los valores $IS_d(n)$ son menores que los valores $IS_d(g)$ correspondientes, donde *d* es un descendiente inmediato de *r.*

La elección hecha en el algoritmo es correcta porque en la expresión para $IS_r(g)$ debe incluirse el valor de la cardinalidad de al menos un MIN 1-MIS en un árbol *T[d]*, donde *d* es un descendiente inmediato de *r, y* con respecto a este MIN 1-MIS el nodo *d* debe ser *negro.* Se sigue que la elección debe minimizar la diferencia entre las cardinalidades de los conjuntos representados por $IS_d(n)$ *y* aquellos representados por $IS_d(g)$, puesto que cualquier otra elección permitiría que el valor $IS_r(g)$ se pudiera mejorar. Consecuentemente, por inducción, se sigue que el algoritmo está bien definido.

La complejidad de tiempo del algoritmo incluye:

- el tiempo $C'(n)$ necesario para encontrar la redución de *T,* el cual, de acuerdo a la afirmación 4 está en *O(n);* más
- el tiempo que toma el primer ciclo, el cual está claramente en *O(n);* más
- el tiempo que toma el segundo ciclo, el cual es a lo más $\sum_{i\in\{j+1,..,n\}} 5|D(r_i)|$, donde $D(r_i)$ incluye los descendientes inmediatos del nodo , claramente, está suma está en *O(n).*

De esta manera, la complejidad de tiempo del algoritmo está dada por:

$$C(n) \leq \sum_{i\in\{j+1,..,n\}} 5|D(r_i)| + C'(n) + j$$

donde para toda $i \in \{j+1, \ldots, n\}$, el conjunto $D(r_i)$ contiene los descendientes inmediatos del nodo r_i, $C'(n)$ es la complejidad de encontrar una redución del árbol *T* y *j* es el número de hojas de *T*.

Concluimos que $C(n) \in O(n)$.

■

1.2 Algoritmo para el problema del Min 1-MIS restringido a *k*-árboles parciales

En esta subsección consideramos el caso cuando las instancias del problema del Min α-MIS son *k*-árboles parciales y $\alpha = 1$, donde $k \geq 1$ es un entero positivo.

La clase de los *k*-árboles es una generalización de la clase de los árboles, los cuales resultan ser 1-árboles. En general, un *k*-árbol se construye a partir de un grafo completo con *k* vértices agregando repetidamente otros vértices, cada uno conectado con *k* aristas a un subgrafo completo con *k* vértices. Un *k*-árbol parcial es un *k*-árbol del cual algunas de sus aristas se han borrado.

El problema de decidir si un grafo es un *k*-árbol parcial, dado el grafo y *k* como entrada, es NP-completo [1], pero si *k* es fijo, el reconocimiento de *k*-árboles parciales se puede hacer en $O(n^2)$ pasos [4].

La clase de *k*-árboles parciales o equivalentemente la clase de grafos con treewidth acotada por *k* [19] es rica. Esta incluye grafos con bandwidth y cutwidth restringidas, grafos series-parallel, grafos k-outerplanar, grafos cordales con tamaño máximo de clique igual a *k*, grafos (*k*, 2)-partitas y algunos otros. Algunas relaciones entre *k*-árboles parciales y algunas clases de grafos bien conocidas se han dado en [16], [19], [10] y [15].

Definimos ahora formalmente los *k*-árboles parciales, la reducción de un *k*-árbol y otros conceptos relacionados. En lo que sigue *k* siempre será un entero positivo y los grafos serán siempre finitos.

Definición 4. *Sea $G = (V, E)$ un grafo. Un conjunto $K \subseteq V$, donde $|K| = k$, es un* k-clique *si K induce un grafo completo.*

Damos ahora dos definiciones equivalentes de *k*-árboles, de las cuales la segunda en más conveniente para nuestro propósitos.

Definición 5 *Un* *k*-árbol *es:*

- *un grafo completo de k vértices, o*
- el grafo $(V \cup \{v\}, E \cup \{\{u_1, v\}, \ldots, \{u_k, v\}\})$ obtenido de un k-árbol $T = (V, E)$, *donde* $\{u_1, \ldots, u_k\} \subseteq V$ *es un k-clique en T y* $v \notin V$.
- *k*-árboles son los grafos obtenidos aplicando las dos reglas anteriores solamente.

Llamaremos el vecindario abierto de un vértice *v* al conjunto de vértices adyacentes a *v*, que denotaremos por *adj(v)*. El vecindario cerrado de un vértice *v* es $adj(v) \cup \{v\}$.

Definición 6. *Un grafo $T = (V, E)$ es un k*-árbol si:

- V es un k-clique, o
- *existe un vértice $v \in V$ tal que $grad(v) = k$, $adj(v)$ es un k-clique y $T' = (V-\{v\}, E-\{\{x, v\} \mid \{x, v\} \in E\})$ es un* *k*-árbol.

El proceso dado en la Figura 3 muestra la construcción de un *k*-árbol de acuerdo con la definición 5. Dado el último grafo en la secuencia, se puede mostrar, usando definición 6, que es un *k*-árbol invirtiendo las mostrados en la secuencia.

La caracterización de los *k*-árboles dada en la definición 6 puede usarse para definir un proceso de reducción, puesto que ella implica que un grafo es un *k*-árbol si y sólo si sus vértices se pueden ir quitando por pasos, donde un vértice se puede remover en un paso particular si tiene grado *k* y si está conectado a un *k*-clique, se continúa de esta manera hasta que sólo quede un grafo completo con *k* vértices [3]. Este *k*-clique que queda al final es llamado la raíz del *k*-árbol. Los vértices que se pueden remover en cada etapa son las *k*-hojas de los *k*-árboles intermedios. Esta reducción define una relación de precedencia entre *k*-cliques y los vértices. Estos conceptos se formalizan en las siguientes definiciones:

Definición *7. Sea G un grafo. Un vértice v Î V es* simplicial si su *vecindario cerrado es un grafo completo [17].*

Definición 8 *Una reducción de un k-árbol T=(V,E), donde* |V|= *n+1, es una secuencia parcial* $r_0,\ldots,r_{m=n-k}$ *de un ordenamiento* $r_0,\ldots,r_n$ *de V tal que para todo iÎ {0, . . ., m}, el vértice es simplicial en el grafo inducido por {*r_i*, . . ., *r_n*}, y su vecindario cerrado tiene cardinalidad k+1 [14].*

Vea la Figura 3 para un ejemplo de una reducción de un *k*-árbol. Note que en un *k*-árbol *T* cualquier vértice de grado *k*, i.e., una *k*-hoja, tiene un *k*-clique como su vecindario abierto y que el *k*-clique que corresponde al vecindario abierto de r_m es la raíz de *T.*

Dada una reducción, podemos definir un orden parcial de los vértices con respecto a los *k*-cliques en *T*, de manera que cada vértice que no es parte de la raíz de *T* es un descendiente de algún *k*-clique en *T.* Definimos un orden de este tipo:

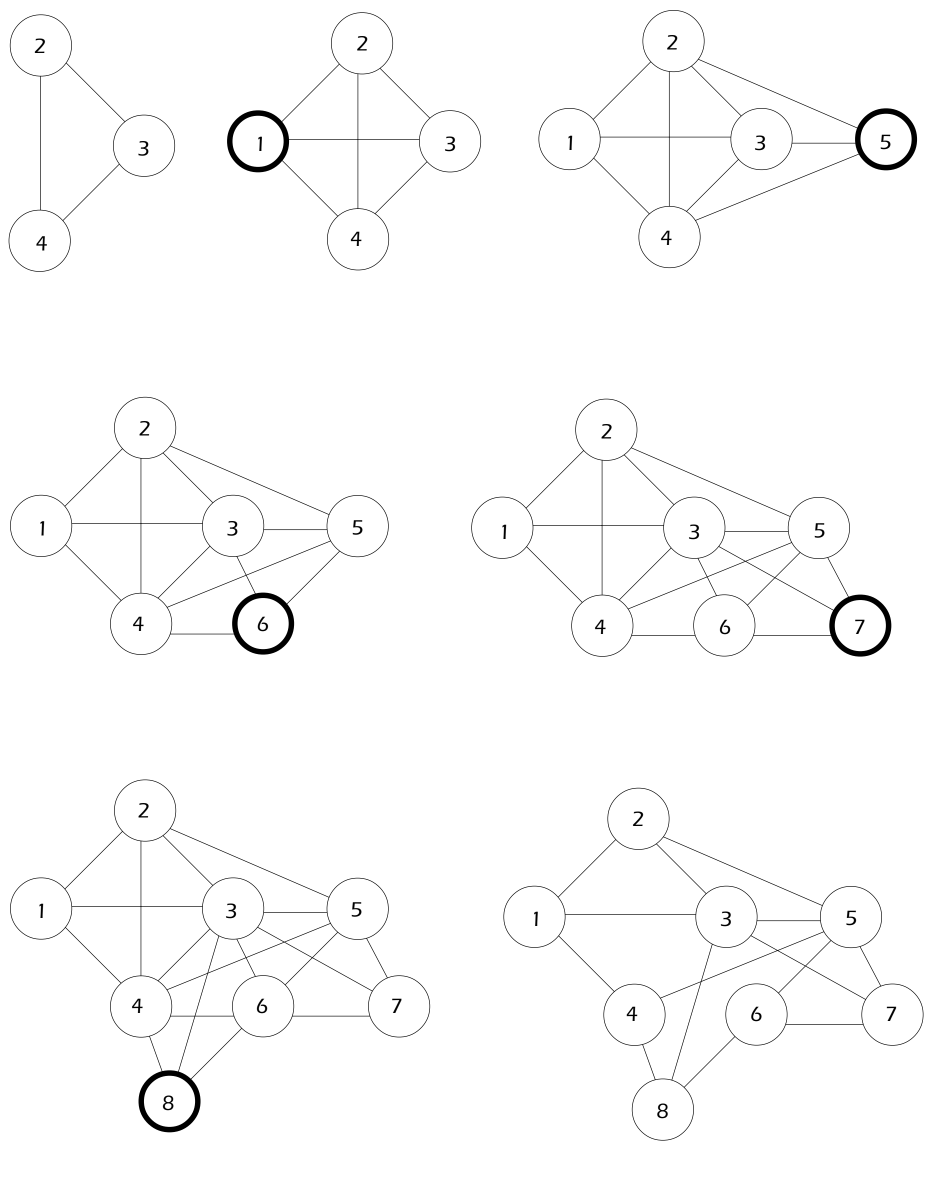
Definición 9. *Sean T = (V,E) un k-árbol, la secuencia* $r_0,\ldots,r_m$ *una de sus reducciones* y |V| = *n + 1. Un vértice* r_i*, donde 0£ i£ m, es un* descendiente *de un k-clique K si y sólo si* r_i Ï *K y* r_i*, en el grafo inducido por* $\{r_i,\ldots,r_n\}$*, tiene como vecindario abierto a K o su vecindario abierto es K´ y todos los vértices en K´ que no están en K son descendientes de K.*

Dado un *k*-árbol *T* y una reducción *R*, decimos que *T* es ordenado de acuerdo con el ordenamiento dado en la Definición 9.

Estamos interesados en la clase más amplia de los *k*-árboles parciales.

Definición 10. *Un k-árbol* parcial *de un k-árbol T = (V, E) es un grafo G = (V, E´) donde E´Í E; i.e., G se obtiene de un k-árbol borrando algunas de sus aristas.*

Note que si *G* es un *k*-árbol parcial, entonces una reducción de G se puede encontrar en tiempo lineal. De igual manera, dado un *k*-árbol parcial G y su reducción, un *k*-árbol del cual G es un *k*-árbol parcial se puede encontrar en tiempo lineal. Note adicionalmente que este *k*-árbol no es necesariamente único. Podemos también



Reducción R = 8, 7, 6, 5, 1

FIGURA 3. CONSTRUCCIÓN DE UN 3-ÁRBOL, SU REDUCCIÓN Y UN 3-ÁRBOL PARCIAL transferir el ordenamiento del *k*-árbol *T* al *k*-árbol parcial G de una manera natural.

Sea *K* un *k*-clique en el *k*-árbol *T* y sea $R = r_0, \ldots, r_m$ su reducción, las componentes conexas inducidas por los descendientes de *K, desc(K),* son los ramas de *K y K* es la base de una rama de *K.* Adicionalmente, si $|V| = n + 1$, entonces para todo r_i la reducción de *R* (usaremos la notación de conjuntos $r \in R$, aún cuando *R* es una secuencia, para referirnos al hecho de que *r* es un elemento de la secuencia *R.* El contexto en el que tales expresiones ocurren debe de indicar sin ambigüedad su significado):

- $K(r_i)$ o simplemente *K,* cuando r_i es sobreentendido, denota el vecindario abierto de r_i en el grafo inducido por $\{r_i, \ldots, r_n\}$,
- $K'(r_i)$ es el vecindario cerrado de r_i, i e., $K(r_i) \cup \{r_i\}$,
- $K^u = K' - \{u\}$, para todo $u \in K'$ y
- $B(K(r_i))$ es el conjunto de vértices *u* descendientes de $K(r_i)$ tal que $u = r_j \in R$ y $j < i$.

Note que *K′* induce un *(k+ 1)*-clique; i.e., *(k+ 1) k*-cliques que se translapan, y que el *k*-clique $K(r_i)$ es un separador de los nodos en *T* y que ningún separador de nodos de *T* tiene cardinalidad mayor que *k* [2].

Ahora establecemos dos propiedades de los *k*-árboles parciales relevantes a los argumentos que usamos para mostrar que nuestro algoritmo está correctamente definido.

Lema 2. *Sea K un k-clique en un k-árbol T. El grafo T′ inducido por* $K \cup \bigcup_{i \in I} V(C_i)$, *donde* $(C_i)_{i \in I}$ *son ramas de K, es un k-árbol.*

Prueba. Vea [15].

■

Lema 3. Si *K es un k-clique en un k-árbol T,* $R = r_0, \ldots, r_m$ *es una reducción de T, y w es un descendiente de K tal que* $K = K(w)$, *entonces para todo* $u, v \in K'(w)$,

$$desc(K^u) \cap desc(K^v) = \emptyset,$$

donde $u \neq v$; *i.e., descendientes de los (k + 1) k-cliques en K′ son disjuntos dos-a-dos.*

Prueba. Vea [16].

■

Sea *H* un *k*-árbol parcial y sea I un MIN 1-MIS. Sea *K* cualquier *k*-clique en un *k*-árbol del cual *H* es un subgrafo y sea *R* una reducción de *H.* Por conveniencia notacional nos referiremos a esos *k*-cliques *K* en el *k*-árbol del cual *H* es un subgrafo como *k*-cliques en *H.*

Considere el *k*-árbol parcial *H′* el cual por el Lema 2 es inducido por $K \cup desc(K)$, para cualquier *k*-clique K en H, y $I_K = I \cap (K \cup desc(K))$. Claramente, I_K es un conjunto independiente en el subárbol *H′.* Siguiendo la nomenclatura introducida en la subsección anterior, note que el *k*-clique *K* se particiona en tres subconjuntos con respecto a I_K: un conjunto de nodos *negros, N,* un conjunto de nodos *grises, G, y* un conjunto de nodos *blancos, B.* Por el Lema 3, los nodos *blancos* sólo pueden aparecer en *K,* porque si hubiera nodos *blancos* en cualquier otra parte de *H′,* ellos permanecerían *blancos* en *H* con respecto a I, contradiciendo la hipótesis de que I es un MIN 1-MIS.

Lema 4. *El conjunto* I_K *es un conjunto independiente de cardinalidad mínima entre todos los conjuntos independientes en H′ que induce una partición de K en los conjuntos N de los nodos negros, G de los nodos grises, B de los nodos blancos y donde los únicos nodos blancos son aquellos en* $B \subseteq K$.

Prueba

Suponga que I_K no es como se afirma. Sea I_K un conjunto independiente de cardinalidad menor que la de I_K, el cual particiona el *k*-clique *K* en los conjuntos *N* de nodos *negros, G* de nodos *grises, B* de nodos *blancos y* donde los únicos nodos *blancos* son aquellos en *B.* Considere $I' = (I \setminus I_K) \cup I'_K$, claramente el conjunto I′ es un 1-MIS en el grafo *H y* tiene cardinalidad menor que la de I. Esto contradice la hipótesis que I es un MIN 1-MIS.

■

El Lema 4 establece que cualquier solución al problema MIN 1-MIS exhibe subestructura óptima [8]. Este hecho se utilizará para desarrollar un algoritmo de programación dinámica para resolver el problema MIN 1-MIS.

Definición 11. *Sea H un k-árbol parcial, R su reducción y K un k-clique en H. Sea P(K(r)), donde $r \in R$, el conjunto de todos los conjuntos independientes en el k-árbol parcial inducido por KUB(K(r)), donde solamente nodos* blancos *(si hay algunos) aparecen en K.*

La cardinalidad de *℘(K(r))* puede ser muy alta pero dado el hecho que *K* es un separador de los nodos en *H y* que cualquier solución al MIN 1-MIS tiene subestructura óptima, indica que sólo debemos mantener un *representante,* aquél de cardinalidad menor entre todos los conjuntos independientes que inducen una partición particular de *K* en nodos *negros, grises y blancos,* donde nodos *blancos (si* hay algunos) aparecen en *K* solamente. Formalizamos estas ideas a continuación:

Definición 12. *Sean I y I′ dos conjuntos en ℘(K(r)), para algún $r \in R$.*

Sean [N, G, B] and [N′, G′, B′] dos particiones de K en nodos negros, grises, y blancos *con respecto a I y I′, respectivamente. Decimos que I y I′ son* similares si y *sólo si:*

- N = N′ y
- G = G′

Se sigue de la definición 12 que si *I es similar* a *I′* entonces B también es igual a *B′.* Se puede mostrar fácilmente que *similitud* es una relación de equivalencia donde los índices pueden identificarse con los tripletes *[N, G, B]* de nodos *negros, grises y blancos* correspondiendo a la partición de *K* con respecto a todos los conjuntos independientes que particionan *K* de esa manera. Esta relación de equivalencia tiene índice finito, en realidad tiene un índice cuya cardinalidad es acotada por una constante, puesto que el número de clases de equivalencia *|S(K)|*, donde *S(K)* es el conjunto que contiene las clases de equivalencia en *℘(K(r)),* es acotado por una constante.

$$|S(K)| \le \sum_{i=0}^{k} \binom{k}{i} 2^{k-i} \le 2^{k+1}$$

Nuestro objetivo es encontrar el conjunto $S(K_0)$ del *k*-árbol parcial *H,* donde K_0 es la raíz de *H* y el valor de la cardinalidad del conjunto independiente de menor cardinalidad por clase de equivalencia. Si *IS* es un índice de una clase de equivalencia, llamemos la cardinalidad del conjunto independiente de menor cardinalidad representado por tal índice *f (IS).* Se sigue que si hay un índice [N, G, ∅] en $S(K_0)$ tal que $f([N, G, \emptyset]) \le \lambda$, para algunos conjuntos *N y* G, entonces existe un MIN 1-MIS en *H* de cardinalidad no mayor que λ.

En la figura 4 mostramos un algoritmo serial para el MIN CONJUNTO INDEPENDIENTE 1-MAX restringido a *k*-árboles parciales, donde $k \ge 1$ es cualquier constante entera. Este algoritmo asume que los conjuntos *S(K),* para todos los *k*-cliques en *H,* son inicialmente vacíos.

Teorema 2. *El algoritmo para el problema del* MIN 1-MIS *restringido a k-árboles parciales es correcto y tiene complejidad en O(n).*

Prueba

Usando inducción sobre el número de nodos en la reducción del *k*-árbol parcial T podemos concluir que el algoritmo está correctamente definido. El paso inductivo se sigue de la observación de que el índice para cada clase de equivalencia en *℘(K(r))* se puede construir con los índices de las clases de equivalencia en los conjuntos *℘(K*^{u}*),* donde $u \in K'(r)$. Estos índices representan conjuntos independientes que se pueden agrupar para formar un conjunto independiente para el *k*-árbol parcial inducido por *K(r) U B(K(r)).* Esta condición se verifica en el tercer ciclo (el segundo ciclo anidado) del algoritmo. La condición es directa, se reduce a verificar que:

- la unión de los nodos *negros* incluidos en los índices forman un conjunto independiente en $K(r)$,
- no hay nodos *grises* representados en algún índice que aparecen *negros* en otro índice y
- el nodo r que se borrará de H, aparece como un nodo *negro o gris* en uno de los índices que se usará para construir el nuevo índice en el conjunto $S(K(r))$.

El tiempo lineal se sigue del hecho de que el número de todas las instrucciones ejecutadas en todos los ciclos del algoritmo, excepto el ciclo exterior, está acotado por una constante en k. El ciclo exterior está claramente acotado por el número de nodos en H. Así, el tiempo total es lineal en el número de nodos en H. Note que hay algunas constantes exponenciales en k, pero puesto que k es fijo, no afectan la complejidad del algoritmo.

■

Debemos notar que el algoritmo como se muestra determina la existencia de un Min 1-MIS de cardinalidad a lo más λ. El algoritmo puede ser fácilmente modificado para encontrar un Min 1-MIS, si es necesario.

2 Ejemplo: Determinación de un Min 1-MIS en un 3-árbol parcial

Sea G el 3-árbol parcial dado en la figura 3, λ = 3, y sea R = 8, 7, 6, 5,1 su reducción. En este ejemplo, excepto por el primer nodo que será eliminado, mostramos sólo estados asociados con k-cliques que tienen conjuntos de descendientes no vacíos, a saber, $S(\{2, 3, 4\})$, $S(\{3, 4, 5\})$, $S(\{3, 4, 6\})$, S({3,5,6}). En este caso, el algoritmo *MinK-MIS,* dado en la figura 4, determina la existencia de un Min 1-MIS con cardinalidad a lo más 3.

Nodo que se borrará: 8.

Los siguientes cuatro estados se crean antes de encontrar el estado final asociado con el k-clique {3, 4, 6}, ellos corresponden a los 3-cliques en K' cuyos estados asociados son vacíos en el momento del borrado del nodo 8.

Algoritmo *MinK-MIS(H,R)*

ENTRADA: Un k-árbol parcial H, una reducción $R = r_0, \ldots, r_n$ de T, y un entero positivo λ.

SALIDA: Indicación acerca de la existencia de un Min 1-MIS de cardinalidad a lo más λ en H.

for $r = r_0, \ldots, r_n$ do

$M \leftarrow \{1, \ldots, k+1\}$

$K' \leftarrow \{u_1, \ldots, u_{k+1}\} = \{r\} \cup K(r)$

for each $u \in K'$ such that $S(K^u) = \emptyset$ do

$S(K^u) \leftarrow \{[I, Adj(I), K^u \setminus I \cup Adj(I)]$, *para todo conjunto independiente* $I \subseteq K^u\}$ y el valor f asociado con un índice $[I, Adj(I), K^u \setminus I \cup Adj(I)]$ es $|I|$.

endfor

else

for cada seleccción de índices $(IS_i)_{i \in I}$ donde:

$IS_i = [B_i, G_i, W_i] \in S(K^{u_i})$, para toda $u_i \in K'$ y

$\cup_{i \in M} B_i$ es un conjunto independiente y

para toda i y j en M *el conjunto* $B_i \cap G_j$ es vacío y

para algún $i \in M$, $r \in B_i$ o $r \in G_i$ do

Sea $B = \cup_{i \in M} B_i \cap K(r)$ y

$G = (\cup_{i \in M} G_i \cup Adj(\cup_{i \in M} B_i)) \cap K(r) \setminus \cup_{i \in M} B_i$ y

$W = K(r) \setminus \cup_{i \in M} G_i \cup \cup_{i \in M} B_i \cup Adj(\cup_{i \in M} B_i)$

add $[B, G, W]$ a *NewS, con f igual a*

$$\left(\sum_{i \in M} f(IS_i)\right) - \sum_{i=0}^{M-1} \sum_{j=i+1}^{M} |B_i \cap B_j|$$

endfor

$S(K(r)) \leftarrow NewS$

endfor

if existe un conjunto independiente $IS = [B, G, \emptyset]$ en $S(K_0)$ tal que $f(IS) < \lambda$ then

Output('Existe un conjunto Min 1-MIS de cardinalidad a lo más λ en H')

else

Output('No existe un conjunto Min 1-MIS de cardinalidad a lo más λ en H')

endif

end

FIGURA 4. ALGORITMO PARA EL PROBLEMA DEL MÍNIMO 1-MIS PARA *K*-ÁRBOLES PARCIALES

K = {3, 4, 6}.

S({3, 4, 6}) = { [Æ, Æ, {3, 4, 6}], [{3}, Æ, {4, 6}], [{4}, Æ, {3, 6}], [{6}, Æ , {3, 4}], [{3, 4}, Æ, {6}], [{3, 6}, Æ , {4}], [{4, 6}, Æ , {3}], [{3, 4, 6}, Æ , Æ] } cada índice con valor de *f* igual a 0, 1, 1, 1, 2, 2, 2, y 3 respectivamente.

K = {3, 4, 8}

S({3, 4, 8}) = { [Æ, Æ, {3, 4, 8}], [{3}, {8}, {4}], [{4}, {8}, {3}], [{8}, {3, 4}, Æ], [{3, 4}, {8}, Æ] } cada índice con valor de *f* igual a 0, 1, 1, y 2 respectivamente.

K = {3, 8, 6}

S({3, 8, 6}) = {[Æ, Æ, {3, 6, 8}], [{3}, Æ, {6, 8}], [{8}, {3, 6}, Æ], [{6}, {8}, {3}], [{3, 6}, {8}, Æ] } cada índice con valor de *f* igual a 0, 1, 1, 1 y 2 respectivamente.

K = {4, 6, 8}

S({4,6, 8}) = { [Æ, Æ, {4,6, 8}], [{4}, {8}, {6}], [{6}, {8}, {4}], [{8}, {4, 6},Æ], [{4,6}, {8}, Æ] } cada índice con valor de *f* igual a 0, 1, 1, 1 y 2 respectivamente.

El valor final del estado asociado con *K(8)* = {3, 4, 6} es:

S({3, 4, 6}) = { [Æ, {3, 6}, {4}], [{3, 4}, Æ , {6}], [{3,6}, Æ , {4}], [{4,6}, Æ , {3}], [{4}, {3, 6},Æ], [{3}, Æ, {4,6}], [{6}, Æ , {3, 4}], [{3, 4,6}, Æ, Æ] } cada índice con valor de *f* igual a 1, 2, 2, 2, 2, 1,1 y 3 respectivamente.

Después de este punto se muestran sólo los estados asociados con *k*-cliques que tienen un conjunto no vacío de descendientes.

Nodo que se borrará: 7

K(7) = {3, 5, 6}

S({3, 5, 6}) = { [Æ, {3, 6}, {5}], [{5}, {3, 6}, Æ], [{3}, {5}, {6}], [{6}, {5}, {3}], [{3, 6}, {5}, Æ] } cada índice con valor de *f* igual a 1, 2, 1, 1 y 2 respectivamente.

Nodo que se borrará: 6

K(6) = {3, 4, 5}.

S({3, 4, 5}) = { [Æ , {2, 3}, Æ], [{3}, {2}, {4}], [{3, 4}, {2}, Æ], [{4}, Æ, {2, 3}], [{4}, {3}, {2}], [Æ, Æ, {2, 3, 4}], [{2}, {3, 4}, Æ], [{2, 3}, Æ , {4}], [{2, 4}, {3}, Æ] } cada índice con valor de *f* igual a 3, 2, 3, 2, 3, 1, 4, 3 y 3 respectivamente.

Nodo que se borrará: 5

K(5) = {2, 3, 4}

S({2, 3, 4}) = { [Æ, {2, 3, 4}, Æ], [{3}, {2}, {4}], [{3, 4}, {2}, Æ], [{4}, Æ, {2, 3}], [{4}, {3}, {2}], [Æ, Æ, {2, 3, 4}], [{2}, {3, 4}, Æ], [{2, 3}, Æ , {4}], [{2, 4}, {3}, Æ] } cada índice con valor de *f* igual a 3, 2, 3, 2, 3, 1, 4, 3 y 3 respectivamente.

Nodo que se borrará: 1

K(1) = {2, 3, 4}

S({2, 3, 4}) = { [Æ, {2, 3, 4}, Æ], [{3}, {2, 4}, Æ], [Æ, {2, 4}, {3}], [{2, 4}, {3}, Æ], [{2}, {3, 4}, Æ], [{2}, {3}, {4}], [{2, 3}, {4}, Æ], [{2, 4}, {3}, Æ], [{2}, {3}, {1}] } cada índice con valor de *f* igual a 4, 3, 2, 3, 4, 2, 3, 3 y 3 respectivamente.

Después del borrado del nodo 1, el estado asociado con el *k*-clique {2, 3, 4} incluye el índice [{2, 4}, {3}, Æ] con un valor de f de 3. Esto significa que existe un Min 1-MIS con la cardinalidad requerida.

Conclusiones

Hemos mostrado que el problema del Min 1-MIS restringido a *k*-árboles parciales se puede resolver en tiempo lineal, cuando *k* es cualquier constante positiva y la instancia consiste de un *k*-árbol parcial, su reducción y un entero positivo l. E1 algoritmo decide si existe un Min 1-MIS de cardinalidad a lo más l. E1 Min 1-MIS se puede construir manteniendo la misma complejidad de tiempo, representando los conjuntos independientes intermedios con listas encadenadas.

No se sabe si el problema del Min a-MIS se puede resolver en tiempo lineal o aún polinomial cuando a es parte de la instancia del problema o cuando a es una constante positiva mayor que 1 y las instancias se mantienen restringidas a *k*-árboles parciales.

No hemos intentado reducir la magnitud de las constantes que aparecen en la cota para la complejidad de tiempo del algoritmo *MinK-MIS*. Estas constantes son exponenciales en *k*, pero puesto que *k* no depende de la entrada, ellas no afectan la complejidad. El hecho que estas constantes son exponenciales en *k* limita la aplicabilidad del algoritmo, aunque tenga compejidad de tiempo lineal. La complejidad de tiempo del algoritmo para árboles no incluye este tipo de constantes. Así, su aplicabilidad no está restringida en el mismo sentido que lo está el algoritmo para *k*-árboles parciales.

La técnica utilizada para resolver el problema del MIN 1-MIS restringido a grafos con treewidth acotado, *frecuentemente* produce algoritmos eficientes para problemas cuyas instancias se restringen a *k*-árboles parciales. Sin embargo, es una pregunta no resuelta la caracterización de los problemas a los que se les puede aplicar esta técnica T

Bibliografía

[1] ARNBORG, Stefan; CORNEIL, Derek G.; PROSKUROWSKI, Andrzej. *Complexity of finding embeddings in a k-tree*. SIAM J. of Alg. and Disc. Math., *8(2):277-284, 1987.*

[2] ARNBORG, Stefan; PROSKUROWSKI, Andrzej. *Linear time algorithms for NP-hard problems restricted to partial k-trees.* To appear Discr. Applied Math.

[3] ARNBORG, Stefan; PROSKUROWSKI, Andrzej. *Algorithms on graphs with bounded decomposability. Congressus Num.,* (53):161-170, 1986.

[4] *Bodlaender*, Hans L. *Improved self-reduction algorithms for graphs with bounded treewidth*. Technical Report RUU-CS-88-29, Computer Science Dept., University of Utrecht, 1988.

[5] BOLLOBÁS, B.; COCKAYNE, Ernest J.; MYNHARDT C. M., *Generalized minimal domination parameters for paths.* Technical Report DM-406-IR, Department of Mathematics, University of Victoria, 1985.

[6] *Cockayne,* Ernest J.; MACGILLIVRAY, G.; MYNHARDT, C. M. *Generalized maximal independence parameters for paths and cycles.* Technical Report DM-410-IR, Department of Mathematics, University of Victoria, 1986.

[7] COOK, Stephen A. *An observation on time-storage trade-off.* Journal of Computer and Systems Science, 9(3):308-316, 1974.

[8] *Cormen*, Thomas H.; LEISERSON, Charles E.; RIVEST, Ronald L. *Introduction to Algorithms.* MIT Press, 1991.

[9] DADOUN, Nou; KIRKPATRICK, David G. *A parallel algorithm for finding maximal independent sets in planar graphs.* Technical Report 87-15, University of British Columbia, 1987.

[10] ELLIS, John A.; MATA-MONTERO, Manrique; MULLER, Hausi. *Serial and parallel algorithms for (k,2)-partite graphs.* Journal of Parallel and Distributed Processing, 1994.

[11] GAREY, Michael R.; JOHNSON, David S. *Computers and Intractability: A Guide to the Theory of NP-completeness. W.* H. Freeman and Co., San Francisco, 1979.

[12] KARP, Richard M.; WIGDERSON, Avi. *A fast parallel algorithm for the maximal independent set problem.* J. of the ACM, pages 762-773, 1985.

[13] LUBY, Michael. *A simple parallel algorithm for the maximal independent set problem.* In Proc. of the 17th ACM Symposium on the Theory of Computing, pages 1-10. ACM, 1985.

[14] MATA-MONTERO, Erick. *Resilience of partial k-tree networks with node and edge failures.* Technical Report CIS-TR-89-15, University of Oregon, Eugene, 1989.

[15] MATA-MONTERO, Manrique; ELLIS, John A. *Treewidth bounded graphs and the longest (s_t)-path problem.* Technical report # 9304, Department of Comp. Science, Memorial University, 1993.

[16] MATA-MONTERO, Manrique; ELLIS, John A. *A linear time algorithm for the longest (s-t)-path problem restricted to partial k-trees. SA JC,* 1994.

[17] NAOR, J.; NAOR, M.; SCHAFFER, A. *Fast parallel algorithms for chordal graphs.* Proc. of the 19th Annual ACM Symp. on the Theory of Computing, 1987.

[18] *Sagan*, Bruce E. *A note on independent sets in a tree.* SIAM J. Disc. Math., 1(1):105-108, 1988.

[19] *Van Leewuwen*, J. (editor). *Algorithms and Complexity,* volume A of The Handbook of Theoretical Computer Science. MIT Press/Elsevier, 1990.