ELSEVIER

Discrete Applied Mathematics 79 (1997) 75–89

DISCRETE APPLIED MATHEMATICS

Transversal partitioning in balanced hypergraphs

Elias Dahlhaus[a,*], Jan Kratochvíl[b,1], Paul D. Manuel[c], Mirka Miller[c]

[a] *Basser Department of Computer Science, University of Sydney, NSW, Australia 2006*

[b] *Department of Applied Mathematics, Charles University, Malostranské nám. 25, 118 00 Prague, Czech Republic*

[c] *Department of Computer Science, University of Newcastle, Newcastle, Australia 2308*

Received 5 September 1995; received in revised form 20 October 1996; accepted 5 March 1997

Abstract

A transversal of a hypergraph is a set of vertices meeting all the hyperedges. A k-fold transversal Ω of a hypergraph is a set of vertices such that every hyperedge has at least k elements of Ω. In this paper, we prove that a k-fold transversal of a balanced hypergraph can be expressed as a union of k pairwise disjoint transversals and such partition can be obtained in polynomial time. We give an NC algorithm to partition a k-fold transversal of a totally balanced hypergraph into k pairwise disjoint transversals. As a corollary, we deduce that the domatic partition problem is in polynomial class for chordal graphs with no induced odd trampoline and is in NC-class for strongly chordal graphs.

Keywords: Balanced hypergraphs; Totally balanced hypergraphs; Transversal; k-fold transversal

1. Introduction

A 0–1 matrix is *balanced* if it does not contain as a submatrix an edge–vertex incidence matrix of an odd cycle. A 0–1 matrix is *totally balanced* if it does not contain as a submatrix an edge-vertex incidence matrix of any cycle. A *hypergraph* H is an ordered pair $(V,\mathscr{E})$ where V is a set of vertices and $\mathscr{E}$ is a family of subsets of V. The members of $\mathscr{E}$ are called hyperedges of H. Let $V=\{v_1,v_2,\dots,v_n\}$ and $\mathscr{E}=\{E_1,E_2,\dots,E_m\}$. The *sub-hypergraph* of $H(V,\mathscr{E})$ induced by a subset $S\subset V$, is an ordered pair $(S,\mathscr{E}_S)$ where $\mathscr{E}_S=\{E_i\cap S \mid E_i\in\mathscr{E}$ and $E_i\cap S\neq\emptyset\}$. The *partial hypergraph* of $H(V,\mathscr{E})$ generated by a subset $\mathscr{F}\subset\mathscr{E}$ is an ordered pair $(S',\mathscr{F})$ where S' is the set of vertices of hyperedges of $\mathscr{F}$. Let $A(H)$ denote the hyperedge–vertex incidence matrix of a hypergraph H. A hypergraph H is *balanced* (respectively *totally balanced*) if $A(H)$ is balanced (respectively totally balanced). The original definition of balanced hypergraph, due to Berge, is: a hypergraph is said to be balanced if every odd cycle

* Corresponding author, present address: Department of Computer Science, University of Cologne, Pohligstraße 1, Cologne, Germany.

[1] Supported in part by Czech research grants GAUK 194 and GAČR 0194/1996.

PII S0166-218X(97)00034-6

$(a_1, E_1, a_2, E_2, \ldots, E_{2p+1}, a_1)$ has an edge E_i which contains at least three of its vertices a_j. A graph is said to be *chordal* if it contains no induced cycle of length 4 or greater. A chordal graph is said to be *strongly chordal* if every cycle on six or more vertices contains a chord joining two vertices with an odd distance in the cycle. A *trampoline* is a chordal graph which has a Hamiltonian cycle $x_1, y_1, x_2, y_2, \ldots, x_r, y_r, x_1$ where y_i is of degree 2 for $1 \leqslant i \leqslant r$ and $\{x_1, x_2, \ldots, x_r\}$ is a clique. A trampoline is an *odd* (*even*) *trampoline* if r is odd (even).

Combinatorial graph problems in general, and partition problems in particular, have been studied widely. The well studied partition problems are coloring (partitioning vertex set by independent sets), clique covering (partitioning vertex set by cliques), transversal partitioning [2], and domatic partition (partitioning vertex set by dominating sets). A partition $V_1, V_2, \ldots, V_d$ of V is a *domatic partition* of a graph G if each V_i is a dominating set of G. The *domatic partition problem* is to find a domatic partition of maximum size. Such problems have an application in the optimum location of facilities in a network [18] and in communication networks [11]. The domatic partition problem is shown to be NP-complete for circular arc graphs [4], chordal graphs, and bipartite graphs [11]. It is polynomially solvable for interval graphs [18], strongly chordal graphs [11, 15] and proper circular arc graphs [4]. Moreover, this problem is proved to be in NC-class for interval graphs [19] . The *domatic number* of a graph G is the size of a maximum domatic partition of G and is denoted by $dom(G)$. A graph G is *domatically full* if $dom(G) = \delta(G) + 1$ where $\delta(G)$ is the minimum degree of a vertex in G. A *transversal* of a hypergraph is a set of vertices meeting all the hyperedges. Berge [2] has proved the following partition theorem pertaining to transversals in balanced hypergraphs.

Theorem 1.1 (Berge [2]). *In a balanced hypergraph H, the maximum number of pairwise disjoint transversals equals the minimum cardinality of a hyperedge.*

Farber [8] has used this result to show that strongly chordal graphs are domatically full. Kaplan and Shamir [11] have used Farber's result to find a maximum domatic partition of a strongly chordal graph in polynomial time. Brouwer et al. [5] have shown that the class of graphs whose neighborhood hypergraphs are balanced is identical to the class of chordal graphs with no induced odd trampoline. Thus it follows from the Berge's result that

Theorem 1.2 (Brouwer et al. [5]). *Chordal graphs with no induced odd trampoline are domatically full.*

Since strongly chordal graphs are trampoline-free chordal graphs [8], chordal graphs with no induced odd trampoline form a superclass of strongly chordal graphs. Following Kaplan and Shamir's work, two questions arise:

(i) Is the domatic partition problem polynomially solvable for chordal graphs with no induced odd trampoline?

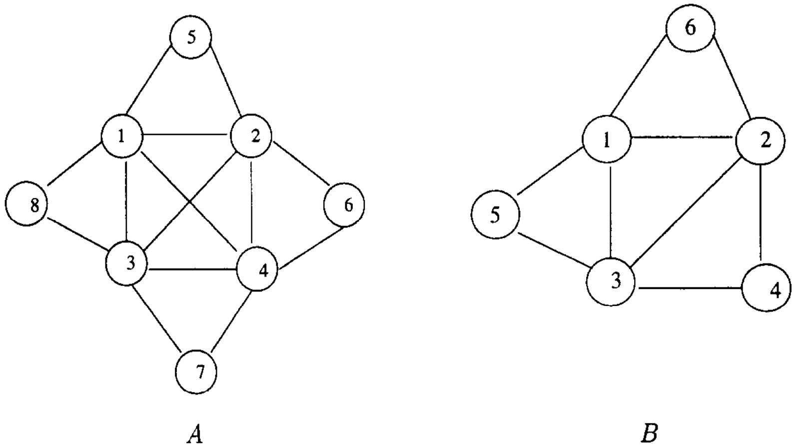

Fig. 1. Even trampoline (A) and odd trampoline (B).

(ii) Does the domatic partition problem on strongly chordal graphs belong to NC-class?

A set S of vertices of a graph G is called *k-fold dominating* if every vertex in S has at least $k-1$ neighbors in S and every vertex in $V \backslash S$ has at least k neighbors in S. It is easy to verify that $dom(G) \leqslant \delta + 1$ [18] and the vertex set V is a $(\delta+1)$-fold dominating set of G where δ is the minimum degree of a vertex in G. Thus the problem of partitioning a k-fold dominating set into k pairwise disjoint dominating sets is one step ahead of the domatic partition problem. Obviously, the union of k disjoint dominating sets in a graph is a k-fold dominating set but the converse is not always true. That is, the partition of a k-fold dominating set into k pairwise disjoint dominating sets is not always possible. In Fig. 1(A) (an even trampoline), $\{1,2,3,4\}$ is a 2-fold dominating set which can be partitioned into two disjoint dominating sets $\{1,4\}$ and $\{2,3\}$. In Fig. 1(B) (an odd trampoline), $\{1,2,3\}$ is a 2-fold dominating set which cannot be partitioned into two disjoint dominating sets. This leads to the following question which extends the domatic partition problem:

> Can a k-fold dominating set be partitioned into k pairwise disjoint dominating sets for all k, $1 \leqslant k \leqslant \delta(G)+1$?

In fact, we can show that this problem is NP-hard on chordal graphs even for $k=2$.

Theorem 1.3. *The problem of partitioning a 2-fold dominating set of a chordal graph into two disjoint dominating sets is* NP-*hard.*

Proof. We reduce the problem of bicolorability of 3-uniform hypergraphs [9] to our problem. If $H=(V,\mathscr{E})$ is a 3-uniform hypergraph (i.e., every edge $e \in \mathscr{E}$ is a 3-element subset of V), we construct a graph $G(V',E')$ such that the vertex set $V' = V \cup \mathscr{E}$ and the edge set $E' = \{(u,v) \mid u,v \in V\} \cup \{(w,e) \mid w \in e \in \mathscr{E}\}$. Now $G(V',E')$ is a chordal graph and V is a 2-fold dominating set of $G(V',E')$. It is easy to verify that the 2-fold dominating set V is partitioned into two disjoint dominating sets of $G(V',E')$ if and only if H is bicolored. Thus the problem of partitioning a 2-fold dominating set of a chordal graph into 2 disjoint dominating sets is NP-hard. □

Berge's concept of transversal partition includes the domatic partition problem. As the hyperedges of a hypergraph are the closed neighborhoods of a graph, the transversal partition reduces to the domatic partition [8]. A *k-fold transversal* Ω of a hypergraph is a set of vertices such that every hyperedge has at least k elements of Ω. The above question can be pushed one step further as follows:

> Can a k-fold transversal of a hypergraph be partitioned into k pairwise disjoint transversals for all $k \leqslant \gamma$, where γ is the minimum cardinality of a hyperedge?

The main result of this paper is to answer this question which extends Berge's Theorem 1.1:

Theorem 1.4. *In a balanced hypergraph H, let γ denote the minimum cardinality of a hyperedge. Then for any $k \leqslant \gamma$, a k-fold transversal can be expressed as a union of k pairwise disjoint transversals.*

It is easy to note that k-fold transversal exists only for $k \leqslant \gamma$. Berge's proof [2] can be used directly to prove Theorem 1.4. Berge, however, does not indicate how to find a transversal partition from a k-fold transversal in a balanced hypergraph or in the special case of totally balanced hypergraphs. Claiming this, Kaplan and Shamir [11] have given an algorithmic proof for Theorem 1.4 in a totally balanced hypergraph. We modify Berge's arguments to give an algorithmic proof of Theorem 1.4 for balanced hypergraphs. We give an NC algorithm to partition a k-fold transversal of a totally balanced hypergraph into k pairwise disjoint transversals. As a corollary, we deduce that the domatic partition problem is in NC-class for strongly chordal graphs.

2. Sequential algorithm to partition a k-fold transversal in a balanced hypergraph

Berge [2] deals with the partition of a k-fold transversal of a balanced hypergraph as follows:

(i) He proves that a balanced hypergraph is bicolorable.

(ii) Using (i), he gives a constructive proof to partition a k-fold transversal into k pairwise disjoint transversals.

Since Berge's proof to bicolor a balanced hypergraph is not constructive, we give a polynomial-time algorithm to bicolor a balanced hypergraph in the following subsection. The problem of bicoloring is to label the vertices of a hypergraph with two colors such that in each hyperedge both colors appear. The following algorithm for bicoloring a balanced hypergraph is a slight modification of Berge's original analytical proof [2].

Lemma 2.1. *A balanced hypergraph can be bicolored in polynomial time.*

Proof. We first compute an enumeration $(v_1, v_2, \ldots, v_n)$ of V and then bicolor the vertices based on this enumeration order. For every $i = 1, 2, \ldots, n$, let H_i denote the sub-hypergraph of H induced by the vertices $\{v_1, v_2, \ldots, v_i\}$ and G_i denote the partial graph of H_i generated by the edges of H_i with exactly two elements. A vertex v of a graph $G(V, E)$ is called an *articulation vertex* if the sub-graph of G induced by $V \setminus \{v\}$ is disconnected.

Now we compute an enumeration $(v_1, v_2, \ldots, v_n)$ of V such that v_i is a non-articulation vertex of G_i. We design the enumeration order from v_n to v_1 as follows: after enumerating $v_n, \ldots, v_{i+2}, v_{i+1}$, construct a spanning forest F_i of G_i. It is known that a leaf of a spanning forest of a graph is a non-articulation vertex of the graph [1]. Hence pick any leaf of F_i and call it v_i.

Now we bicolor $H(V, \mathscr{E})$ using induction based on the enumeration order $(v_1, v_2, \ldots, v_n)$. Assuming that $v_1, v_2, \ldots, v_{i-1}$ admits a bicoloring (S_1, S_2) in H_{i-1}, we color v_i in H_i as follows.

Case 1: If v_i does not belong to any edge of size 2 in H_i, then add v_i to S_1. Then $(S_1 \cup \{v_i\}, S_2)$ is a bicoloring of H_i.

Case 2: If v_i belongs to any edge of size 2 in H_i, consider the vertices of $\{v_1, v_2, \ldots, v_{i-1}\}$ which are adjacent to v_i in G_i. Since H is a balanced hypergraph, G_i is balanced and hence bipartite [2]. Since G_i is bipartite and v_i is not an articulation point of G_i, the vertices of $\{v_1, v_2, \ldots, v_{i-1}\}$ which are adjacent to v_i in G_i, have the same color in H_{i-1}; let S_1 be the class of vertices having this color. Then $(S_1, S_2 \cup \{v_i\})$ is a bicoloring of H_i, because every edge of H_i with two elements is bicolored, since it is an edge of G_i and every edge of H_i with more than two elements is also bicolored, since its intersection with $\{v_1, v_2, \ldots, v_{i-1}\}$ is bicolored.

Thus H_i is bicolored and hence by induction H is bicolored. It takes $O(n+m)$ time to compute a spanning forest [1]. The algorithm computes spanning forest as many times as the number of vertices. Thus bicoloring needs $O(n(n+m))$ steps. □

Combining Berge's proof [2] of Theorem 1.1 and the above bicoloring algorithm, a k-fold transversal Ω of a balanced hypergraph can be partitioned into k pairwise disjoint transversals in $p(n, m)$ time where $p(n, m)$ is polynomial in n and m, and m is the number of the hyperedges. Using the fact that the edges of a balanced hypergraph satisfy Helly's property [3, p. 452], Prisner [16] has shown that balanced hypergraphs have $O(n^2)$ hyperedges. Thus we have

Theorem 2.2. *A k-fold transversal of a balanced hypergraph can be partitioned in polynomial time.*

Corollary 2.3. *The domatic partition problem can be solved for chordal graphs with no induced odd trampoline in polynomial time.*

3. Parallel algorithm to partition a k-fold transversal in a totally balanced hypergraph

In this section, let $H(V,\mathscr{E})$ denote a totally balanced hypergraph. Consider the hyperedge–vertex incidence matrix $A(H)=(a(i,j))$ with m rows representing the hyperedges $E_1,E_2,\dots,E_m$ of H and n columns representing the vertices $v_1,v_2,\dots,v_n$ of H with

$$a(i,j)=\begin{cases}1 & \text{if } v_i\in E_j,\\ 0 & \text{if } v_i\notin E_j.\end{cases}$$

From here on, the matrix $A(H)$ will be denoted as A. A 0–1 matrix is called *Γ-free* if it does not contain the submatrix

$$\Gamma=\begin{pmatrix}1 & 1\\ 1 & 0\end{pmatrix}.$$

Let Ω be a k-fold transversal of a totally balanced hypergraph H. We shall use the following steps to partition a k-fold transversal Ω of a totally balanced hypergraph into k pairwise disjoint transversals $\Omega_1,\Omega_2,\dots,\Omega_k$:

(1) *Input the hyperedge-vertex incidence matrix A in the Γ-free form.*
(2) *Prune matrix A and call the resulting matrix $\bar{A}$.*
(3) *Assign colors to the non-zero entries of $\bar{A}$ such that*
  (i) *the non-zero entries of each column are assigned the same color and*
  (ii) *all the k colors appear in each row.*
(4) *Assign a column with color α to Ω_α.*
(5) *Output transversal partition $\Omega_1,\Omega_2,\dots,\Omega_k$ of Ω.*

Lubiw [14] described a polynomial algorithm to obtain a Γ-free ordering of a 0–1 matrix. In [7], an NC-algorithm to obtain a Γ-free ordering has been described. Latter algorithm is quite complicated. Here we assume A is Γ-free. An example of Γ-free matrix A is given in Fig. 2.

3.1. Pruning matrix A

As a first step of the algorithm, we remove the unnecessary columns of A. When a transversal Ω is partitioned into k transversals, the vertices of $V\backslash\Omega$ are not needed. Thus we remove the columns of the matrix A corresponding to the vertices of $V\backslash\Omega$. The remaining matrix of A is still Γ-free and we again call it A with m rows and n columns.

Let E_i denote the non-zero entries of row i of A. That is, $E_i=\{(i,j)\mid a(i,j)=1,\ 1\leqslant j\leqslant n\}$. Also let R_i denote the k rightmost non-zero entries of row i and $L_i=E_i\backslash R_i$. Thus $E_i=L_i\cup R_i$ for every row i of A (see Fig. 3). In Fig. 2, for $k=3$, $L_2=\{a(2,2),\ a(2,3)\}$ and $R_2=\{a(2,4),a(2,8),a(2,10)\}$.

We replace the non-zero entries of L_i by zeros in each row i of A. That is, in each row i of A we keep only the k rightmost non-zero entries R_i of A and remove the rest.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
E_1	1	1	1	0	0	0	0	0	0	0	0
E_2	0	1	1	1	0	0	0	1	0	1	0
E_3	0	0	1	1	1	0	0	1	0	1	0
E_4	0	0	0	0	0	1	1	0	0	0	1
E_5	0	0	0	0	0	0	1	0	0	1	1
E_6	0	0	0	0	0	0	0	1	0	1	1
E_7	0	0	0	0	0	1	1	0	0	1	1
E_8	0	0	0	0	0	1	1	1	1	1	1

Fig. 2. An example of Γ-free matrix A.

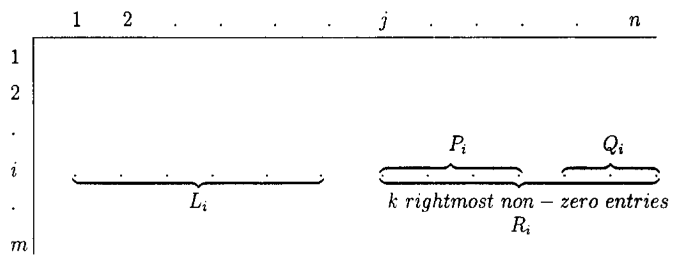

Fig. 3. Matrix A.

The resultant matrix of A is denoted by $\bar{A}=(\bar{a}(i,j))$. Now the non-zero entries of $\bar{A}$ are the members of $\bigcup_{i=1}^{m} R_i$. In $\bar{A}$, there may be a column without any non-zero entry. This column may be colored arbitrarily at the end. So, now we assume that $\bar{A}$ has no such columns.

For $k=3$, Fig. 4 represents the pruned matrix $\bar{A}$ of A in Fig. 2. Using the facts that A is Γ-free and only the leftmost non-zero entries of a row of A are replaced to construct $\bar{A}$, it is easy to observe that

Observation 3.1. *$\bar{A}$ is Γ-free.*

Corresponding to $\bar{A}$, define

$$bottom\_row(j) = \max\{i \mid \bar{a}(i,j)=1\}.$$

The $bottom\_row(j)$ of $\bar{A}$ is the row containing the bottommost non-zero entry of column j in $\bar{A}$. In matrix $\bar{A}$ of Fig. 4, $bottom\_row(3)=1$, $bottom\_row(8)=6$ and $bottom\_row(10)=8$.

Corresponding to each row i of $\bar{A}$, define

$$P_i = \{(i,j)\in R_i \mid bottom\_row(j)=i\} \quad \text{and} \quad Q_i = R_i \backslash P_i.$$

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}	v_{11}
E_1	1	1	1	0	0	0	0	0	0	0	0
E_2	0	0	0	1	0	0	0	1	0	1	0
E_3	0	0	0	0	1	0	0	1	0	1	0
E_4	0	0	0	0	0	1	1	0	0	0	1
E_5	0	0	0	0	0	0	1	0	0	1	1
E_6	0	0	0	0	0	0	0	1	0	1	1
E_7	0	0	0	0	0	0	1	0	0	1	1
E_8	0	0	0	0	0	0	0	0	1	1	1

Fig. 4. An example of $\bar{A}=(\bar{a}(i,j))$.

In row 1 of matrix $\bar{A}$ of Fig. 4, $P_1=\{(1,1),(1,2),(1,3)\}$ and $Q_1=\emptyset$. In row 2, $P_2=\{(2,4)\}$ and $Q_2=\{(2,8),(2,10)\}$. In row 5, $P_5=\emptyset$ and $Q_5=\{(5,7),(5,10),(5,11)\}$. We observe the following from the fact that $\bar{A}$ is Γ-free:

Observation 3.2. *The members of Q_i are the rightmost* 1*'s of row i of $\bar{A}$. That is, for any $(i,j_1)\in P_i$ and $(i,j_2)\in Q_i$ of row i, $j_1<j_2$ (refer to Fig.* 3).

Corresponding to each row i of $\bar{A}$, define

$$\lambda col(i)=\begin{cases}\min\{j\mid(i,j)\in Q_i\} & \text{if } Q_i \text{ is non-empty,}\\ NIL & \text{if } Q_i \text{ is empty.}\end{cases}$$

Corresponding to each row i of $\bar{A}$, define

$$\mu row(i)=\begin{cases}bottom\_row(\lambda col(i)) & \text{if } \lambda col(i)\neq NIL,\\ m & \text{if } \lambda col(i)=NIL.\end{cases}$$

Corresponding to row 1 of $\bar{A}$ in Fig. 4, $\lambda col(1)=NIL$ and $\mu row(1)=8$. For row 2, $\lambda col(2)=8$ and $\mu row(2)=6$. For row 5, $\lambda col(5)=7$ and $\mu row(5)=7$.

Corresponding to each row i of $\bar{A}$, define

$$\bar{Q}_{\mu row(i)}=\{(\mu row(i),j)\mid(i,j)\in Q_i\} \quad\text{and}\quad \bar{P}_{\mu row(i)}=R_{\mu row(i)}\backslash\bar{Q}_{\mu row(i)}.$$

Corresponding to row 1 of $\bar{A}$ in Fig. 4, $\bar{Q}_{\mu row(1)}=\emptyset$ and $\bar{P}_{\mu row(1)}=\{(8,9),(8,10),(8,11)\}$. For row 2, $\bar{Q}_{\mu row(2)}=\{(6,8),(6,10)\}$ and $\bar{P}_{\mu row(2)}=\{(6,11)\}$. For row 5, $\bar{Q}_{\mu row(5)}=\{(7,7),(7,10),(7,11)\}$ and $\bar{P}_{\mu row(5)}=\emptyset$.

Observation 3.3. *The entries of $\bar{Q}_{\mu row(i)}$ are non-zero and $\bar{Q}_{\mu row(i)}\subseteq R_{\mu row(i)}$.*

Proof. The assertion follows from the fact that $\bar{A}$ is Γ-free and $\bar{a}(\mu row(i),\lambda col(i))=1$. $\square$

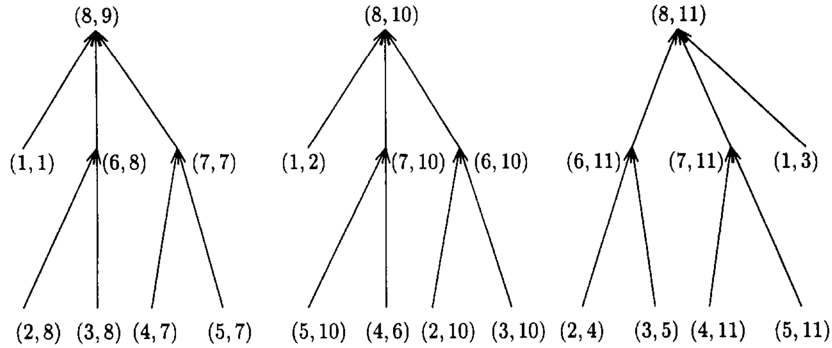

Fig. 5. The forest F of matrix $\bar{A}$.

3.2. Constructing a forest

$\bar{P}_{\mu row(i)}$ and $\bar{Q}_{\mu row(i)}$ are as defined above. Note that the entries of $\bar{Q}_{\mu row(i)}$ need not be the rightmost 1's of row $\mu row(i)$ as those of Q_i in row i. It is true that P_i and $\bar{P}_{\mu row(i)}$ are of the same cardinality for any row i where $|P_i| = |\bar{P}_{\mu row(i)}| = k - |Q_i|$. Let $R = \bigcup_{i=1}^{m} R_i$. Now we construct a forest with vertex set R, the non-zero entries of $\bar{A}$. We define a *Parent* function on $R_i = P_i \cup Q_i$ for every row $i = 1, 2, \ldots, m$ as follows:

(i) On Q_i, *Parent*: $Q_i \to \bar{Q}_{\mu row(i)}$ is a bijective map such that $Parent(\bar{a}(i,j)) = \bar{a}(\mu row(i), j)$ for every $\bar{a}(i,j) \in Q_i$.

(ii) On P_i, *Parent*: $P_i \to \bar{P}_{\mu row(i)}$ is a bijective map. (The *Parent* function on P_i is any bijective map onto $\bar{P}_{\mu row(i)}$).

Let F denote the set of directed edges $(\overrightarrow{u, v})$ such that v is a parent of u of F.

On row 1 of $\bar{A}$ in Fig. 4, the *Parent* function is defined as $Parent(\bar{a}(1,1)) = \bar{a}(8,9)$, $Parent(\bar{a}(1,2)) = \bar{a}(8,10)$ and $Parent(\bar{a}(1,3)) = \bar{a}(8,11)$. On row 2, the *Parent* function is defined as $Parent(\bar{a}(2,4)) = \bar{a}(6,11)$, $Parent(\bar{a}(2,8)) = \bar{a}(6,8)$ and $Parent(\bar{a}(2,10)) = \bar{a}(6,10)$. On row 5, the *Parent* function is defined as $Parent(\bar{a}(5,7)) = \bar{a}(7,7)$, $Parent(\bar{a}(5,10)) = \bar{a}(7,10)$ and $Parent(\bar{a}(5,11)) = \bar{a}(7,11)$ (see Fig. 5).

Observation 3.4. *F is a forest consisting of a set of rooted directed trees with the vertex set R. Moreover, F has exactly k directed trees whose roots are the non-zero entries of the bottommost row (that is, row m) of $\bar{A}$.*

Proof. The *Parent* function is well defined since it is a bijective map on R_i for every $i = 1, 2, \ldots, m$. The *Parent* function is strictly increasing with respect to the order of the rows. Therefore, there is no cycle in F and hence induces a forest. Moreover, the roots of the directed trees of the forest are in row m of $\bar{A}$ because the *Parent* function is strictly increasing with respect to the order of the rows. Since there are k non-zero entries in row m, F has exactly k directed trees. □

Fig. 5 is the forest F constructed from the matrix $\bar{A}$ of Fig. 4. In the figure, $\bar{a}(i,j)$ is represented by (i,j).

Algorithm 3.5. *Here is the parallel algorithm following the construction of the forest:*
- *Input the hyperedge–vertex incidence matrix A in the Γ-free form.*
- *Prune A to get the matrix $\bar{A}$.*
- *Define the Parent function on R, the non-zero entries of $\bar{A}$, and construct forest F.*
- *Color the members of R as follows: First assign distinct color to each root of the directed tree of F. Let u be a member of R. Then u will be a vertex of some tree of the forest. Let r be the root of the tree which contains u. The entry u of R is assigned the color of its root r.*
- *Partition the columns according to the colors of the columns. That is, Ω_α is the set of columns with color α.*
- *Output transversal partition $\Omega_1, \Omega_2, \ldots, \Omega_k$ of Ω.*

Proof of Correctness. It is enough to prove the following:
(i) The non-zero entries of each column $\bar{A}$ are assigned the same color.
(ii) All the k colors appear in each row of $\bar{A}$.

(i) It is enough to show that the non-zero entries of a column are in the same tree of forest F. For any column j, consider the set $\Pi_j = \{\bar{a}(i,j) = 1 \mid i \neq \mathit{bottom\_row}(j)\}$ which contains all the non-zero entries of column j except $\bar{a}(\mathit{bottom\_row}(j), j)$. For any element $\bar{a}(i,j)$ of $\Pi_j, \bar{a}(i,j)$ is in Q_i of row i since $i \neq \mathit{bottom\_row}(j)$. Thus, by the definition of the *Parent* function on Q_i, the parent of $\bar{a}(i,j)$ of Π_j is in column j itself. Moreover, since the *Parent* function is strictly increasing with respect to the order of the rows, $\bar{a}(\mathit{bottom\_row}(j), j)$ is an ancestor of Π_j. Hence all the non-zero entries of column j are in the same tree.

(ii) To prove that all the k colors appear in each row of $\bar{A}$, we use induction on the rows from the bottom. By induction hypothesis, we assume that all the k colors appear in row l for $l > i$. We know that $\mu row(i) > i$ for any row i. All the k colors appear in row $\mu row(i)$ by induction hypothesis. Since the *Parent* function is a bijective map from R_i onto $R_{\mu row(i)}$, all the k colors appear in row i. □

3.3. Complexity analysis

To prune matrix A to get $\bar{A}$, it is enough to find the k rightmost non-zero entries of row i in A. This can be done by parallel prefix computation [13] in $O(\log n)$ time with $O(n^2)$ processors. (Each row is processed independently and needs n processors).

The major part of the algorithm is to design forest F, that is, to define the *Parent* function. For every column j, $\mathit{bottom\_row}(j)$ which is $\max\{i \mid \bar{a}(i,j) = 1\}$, can be found in $O(\log n)$ time with $O(n^2)$ processors using parallel maximum computation [10]. Now the set R_i of row i can be partitioned into P_i and Q_i in constant time with $O(n^2)$ processors, since $Q_i = \{\bar{a}(i,j) \in R_i \mid \mathit{bottom\_row}(j) \neq i\}$. Knowing the *bottom_row* function,

$\lambda col(i)$ of row i of $\bar{A}$ which is $\min\{j \mid \bar{a}(i,j) \in Q_i\}$, can be obtained in $O(\log n)$ time with $O(n^2)$ processors using parallel minimum computation [10]. Since *bottom_row* function is known, $\mu row(i)$ of row i is known since $\mu row(i) = bottom\_row(\lambda col(i))$. Now for each row i, P_i, Q_i and $\mu row(i)$ are known. The function $Parent : Q_i \to \bar{Q}_{\mu row(i)}$ can be evaluated in constant time. The function $Parent : P_i \to \bar{P}_{\mu row(i)}$ can be evaluated in logarithmic time with a work load of $O(k)$. Note that the function *Parent* function on P_i may be any bijective map onto $\bar{P}_{\mu row(i)}$. By parallel prefix computation, we can determine a canonical enumeration of $\bar{P}_{\mu row(i)}$, i.e. the jth element of $\bar{P}_{\mu row(i)}$ gets the number j. In detail, we assign the jth element of R_i with $x_j = 1$ if it is in $\bar{P}_{\mu row(i)}$ and with $x_j = 0$ otherwise, and the jth element of R_i gets index l if the sum of $a_{j'}$, $j' < j$ is l. If the jth element of R_i is in $\bar{P}_{\mu row(i)}$ then it is the lth element of $\bar{P}_{\mu row(i)}$. For each i, this can be done in $O(\log n)$ time with $O(k/\log n)$ processors [13].

The last and important stage is to color the vertices of forest F. For every vertex of the rooted directed tree of F, its root can be found in $O(\log n)$ time with $O(n^2)$ processors. This can be done by *pointer jumping technique* described in [10]. Thus Algorithm 3.5 runs $O(\log n)$ time using $O(n^2)$ processors. □

Theorem 3.6. *A k-fold transversal of a totally balanced hypergraph can be partitioned into k transversals in* $O(\log n)$ *time with* $O(n^2)$ *processors (provided that the incidence matrix is given in the Γ-free form).*

It is proved [14] that the neighborhood matrix of a strongly chordal graph is Γ-free if the vertices are arranged by strong elimination ordering. Since the computation of strong elimination ordering of strongly chordal graphs is in NC-class [7], the Γ-free neighborhood matrix can be realized in polylogarithmic time with polynomial number of processors. Thus we can state that

Theorem 3.7. *A k-fold transversal of a totally balanced hypergraph can be partitioned into k transversals in polylogarithmic time with polynomial number of processors.*

Corollary 3.8. *The domatic partition problem is in* NC-*class for strongly chordal graphs.*

We would like to remark that the parallel algorithm to partition a k-fold transversal of a totally balanced matrix into k transversals can be transformed into an optimal parallel algorithm with $O(n + m)/\log n)$ processors working in $O(\log n)$ time. Here m is the number of non-zero entries in the matrix. Instead of implementing the matrix A as an array, we realize it by a doubly linked list and apply list ranking instead of parallel prefix [6].

4. More on domination

We call a graph *k-unfolding* if every k-fold dominating set can be partitioned into k sets, each of them dominating the original graph. We say that a graph is *unfolding* if it is k-unfolding for every k. The following two problems are of particular interest.

k-fold domination
Instance: A graph G and an integer r.
Question: Does G contain a k-fold dominating set of size at most r?

k-unfold domination
Instance: A graph G and a k-fold dominating set D in G.
Question: Can D be partitioned into k dominating sets? If yes, find a partition.

Both these problems are NP-hard and we have proved in Theorem 1.3 that the 2-unfold domination problem is NP-hard even when restricted to chordal graphs. Now, we will relate the unfolding property to domatic fullness of graphs and the concept of balanced matrices.

Observation 4.1. *Unfolding implies domatically full, but domatically full does not imply unfolding.*

Proof. The 3-trampoline is domatically full but not unfolding. □

In the sequel, $N=(n(i,j))_{i,j=1}^{n}$ will denote the neighborhood matrix of a graph $G(V,E)$ where $V=\{v_1,v_2,\dots,v_n\}$ and

$$n(i,j)=\begin{cases}1 & \text{if } i=j \text{ or } (v_i,v_j)\in E,\\ 0 & \text{otherwise.}\end{cases}$$

Observation 4.2. *A graph G is k-unfolding if and only if for every* 0–1 *vector $\bar{x}$,*

$$N\bar{x}\geqslant k\mathbf{1}$$

implies the existence of k 0–1 *vectors $\bar{y}_1,\bar{y}_2,\dots,\bar{y}_k$ such that*

$$\bar{x}=\sum_{i=1}^{k}\bar{y}_i \quad \textit{and} \quad N\bar{y}_i\geqslant \mathbf{1}, \quad i=1,2,\dots,k.$$

One can show that the graph of the three-dimensional cube is unfolding. On the other hand, it contains a cycle of length 4 as an induced subgraph. By Observation 4.1, since the cycle of length 4 is not domatically full, it is not unfolding. This shows that being unfolding is not a hereditary property. This leads us to the following definition: Call a graph *hereditarily unfolding* if every induced subgraph of G is unfolding. Observation 4.2 leads to the following generalization of the concept of unfolding graphs (Z_0^+ denotes the set of non-negative integers): Define a graph G *weight unfolding* if for every k and every vector $\bar{x} \in (Z_0^+)^n$,

$$N\bar{x} \geqslant k\mathbf{1}$$

implies the existence of k 0–1 vectors $\bar{y}_1, \bar{y}_2, \ldots, \bar{y}_k$ such that

$$\bar{x} = \sum_{i=1}^{k} \bar{y}_i \quad \text{and} \quad N\bar{y}_i \geqslant \mathbf{1}, \quad i = 1, 2, \ldots, k.$$

Observation 4.3. *Weight unfolding implies unfolding.*

Problem 4.4. *Does unfolding imply weight unfolding?*

Similarly as we defined hereditarily unfolding graphs, we define a graph to be *hereditarily weight unfolding* if each of its induced subgraphs is weight unfolding. Of course, hereditarily weight unfolding implies hereditarily unfolding. Even if the answer to Problem 4.4 is negative, we can still ask

Problem 4.5. *Does hereditary unfolding imply hereditarily weight unfolding?*

We should note that these questions apply to graphs with induced cycles, since for chordal graphs the questions are settled by the result of Brouwer et al. [5] (reading that the neighborhood matrix of a graph is balanced if and only if it is an odd trampoline-free chordal graph). Hence

Theorem 4.6. *If G is chordal, then G is hereditarily weight unfolding iff it is hereditarily unfolding iff N is balanced iff G is odd trampoline-free. In particular, interval graphs and strongly chordal graphs are hereditarily weight unfolding.*

We will continue the investigation of properties related to unfolding of dominating sets.

Observation 4.7. *Hereditarily weight unfolding does not imply that the neighborhood matrix is balanced.*

Proof. The cycle of length 6 can be shown to be hereditarily weight unfolding, but its neighborhood matrix is not balanced. □

Proposition 4.8. *The only cycles that are unfolding are C_3, C_6 and C_9. These particular cycles are hereditarily weight unfolding.*

Proof. First one proves that if a cycle is 3-unfolding then its length is divisible by three. Note that the only 3-fold dominating set of a cycle is the whole cycle. Moreover, every dominating set in a cycle of length n has size at least $\lceil \frac{n}{3} \rceil$. Therefore the vertex set of C_n can be partitioned into 3 dominating sets only if (and also if) n is divisible by 3.

Next we show that no cycle of length $3m$, $m \geqslant 4$ is 2-unfolding. Consider a 2-fold dominating set $D = \{1, 2, 3, 5, 6, 7, 9, 10, 11\} \cup \{3i+1, 3i+2 : i \geqslant 4, i < m\}$. Assume D can be divided into two dominating sets U_1 and U_2. Assume $1 \in U_1$. Then $2 \in U_2$ (otherwise 1 that has 2 and $3m \notin D$ as its neighbors cannot be a neighbor of an element of U_2). For the same reason, $3 \in U_1$. Then $5 \in U_2$, because otherwise 4 is not a neighbor of an element in U_2. Similarly, $6 \in U_1$, $7 \in U_2$, $9 \in U_1$, $10 \in U_2$, and $11 \in U_1$. If $m = 4$ then 12 is not a neighbor of U_2, a contradiction. One can show by induction that $3i+1 \in U_2$ and $3i+2 \in U_1$ for $i = 4, 5, \ldots, m-1$. Therefore also in general, $3m$ is not in the neighborhood of U_2.

Finally, one can easily show by case analysis that cycles of length 3, 6, and 9 are 2-unfolding (note that we have shown in the first part of the proof that cycles of length of $3m$ are 3-unfolding). Note that every proper subgraph of a cycle is a disjoint union of paths, and hence a strongly chordal graph. Therefore, the cycles C_3, C_6 and C_9 are hereditarily unfolding. For the proof of weight unfoldingness we refer to the technical report [12]. □

Problem 4.9. *Characterize the classes of unfolding (weight unfolding, hereditarily unfolding, hereditarily weight unfolding) graphs.*

5. Conclusion

We have proved that a k-fold transversal of a balanced hypergraph can be expressed as a union of k pairwise disjoint transversals and the partition of a balanced hypergraph can be obtained in polynomial time. We have given an NC algorithm to partition a k-fold transversal of a totally balanced hypergraph into k pairwise disjoint transversals. As a corollary, we have derived that the domatic partition problem can be solved for chordal graphs with no induced odd trampoline in polynomial time and is in NC-class for strongly chordal graphs. Finally we discussed the problem how to characterize graphs with the property that every k-fold dominating set can be partitioned into k dominating sets. We did not get a final answer.

References

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, Data Structures and Algorithms, Addison-Wesley, New York, 1983.

[2] C. Berge, Balanced matrices, Math. Programming 2 (1972) 19–31.

[3] C. Berge, Graphs and Hypergraphs, North-Holland, Amsterdam, 1973.

[4] M.A. Bonuccelli, Dominating sets and domatic number of circular arc graphs, Discrete Appl. Math. 12 (1985) 203–213.

[5] A.E. Brouwer, P. Duchet, A.Schrijver, Graphs whose neighborhoods have no special cycles, Discrete Math. 47 (1983) 177–182.

[6] R. Cole, U. Vishkin, Deterministic coin tossing with applications to optimal parallel list ranking, Inform. and Control 70 (1986) 32–53.
[7] E. Dahlhaus, M. Karpinski, Fast parallel computation of perfect and strongly perfect elimination schemes,Technical Report 8513-CS, Institut fur informatik, Universitat Bonn, November 1987.
[8] M. Farber, Domination, independent domination, and duality in strongly chordal graphs, Discrete Appl. Math. 7 (1984) 115–130.
[9] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman, New York, 1979.
[10] J. Jaja, An Introduction to Parallel Algorithms, Addison-Wesley, Reading, MA, 1992.
[11] H. Kaplan, R. Shamir, The domatic number problem on some perfect graph families, Inform. Process. Lett. 49 (1994) 51–56.
[12] J. Kratochvíl, P. Manuel, M. Miller, A. Proskurowski, Unfolding dominating sets in graphs, Technical Report.
[13] R. Ladner, M. Fischer, Parallel prefix computation, J. ACM 27 (1980) 831–838.
[14] A. Lubiw, Doubly lexical orderings of matrices, SIAM J. Comput. 16 (1987) 854–879.
[15] S.L. Peng, M.S. Chang, A simple linear time algorithm for the domatic partition problem on strongly chordal graphs, Inform. Process. Lett. 43 (1992) 297–300.
[16] E. Prisner, Hereditarily cliqeue hellymgraphs, J. Combin. Math. Combin. Comput. 14 (1993) 216–220.
[17] A.S. Rao, C.P. Rangan, Linear algorithm for domatic number problem on interval graphs, Inform. Process. Lett. 33 (1989) 29–33.
[18] M. Yu, C. Yang, An optimal parallel algorithm for the domatic partition problem on an interval graph given its sorted model, Inform. Process. Lett. 44 (1992) 15–22.