Binary set systems and totally balanced hypergraphs

François Brucker, Pascal Préa, Célia Châtel

Binary set systems and totally balanced hypergraphs

Célia Châtel[a], François Brucker[a,b], Pascal Préa[a,b]

[a] *Aix Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France*
[b] *École Centrale Marseille, France*

Abstract

We show in this paper that totally balanced hypergraphs (hypergraphs which do not contain special cycles) are exactly hypergraphs that are embeddable into closed hypergraphs for which each vertex admits at most two sons (so called binary hypergraphs). To prove this result we exhibit an efficient algorithm which can produce any binary hypergraph and adapt it to binarize a given totally balanced hypergraph. This result gives, like Lehel[9], a constructive characterization of totally balanced hypergraph.

Keywords: hypergraphs, totally balanced hypergraphs, binary hypergraphs

1. Introduction

Totally balanced hypergraphs, initially defined by Lovasz[10], are an hypergraph structure which corresponds to the notion of tree for graphs (see Lehel [9]), thus occurring in various applications like linear programming, phylogenetic problems (see Spinrad [11] for instance) or more recently in concurrent processes (see Dien [6]). This structure has many nice properties, since they are equivalent to other well known combinatorial models like Γ-free 0/1-matrices (Antsee and Farber [1]), strongly chordal graphs (Farber [7]) or dismantlable lattices (Brucker and Gély [4]).

Finally, as hypergraphs, totally balanced hypergraphs admit a relatively small number of clusters, at most the square of the number of vertices, and admit a convenient graphical representation (Brucker and Préa [5]). So they can be beneficially used as a clustering model. In clustering we are interested in the relationships between objects. These relationships are likely to yield two properties: *homogeneity* (which states what is common between objets for elements in the same cluster) and *separability* (what make two objects different for elements in different clusters). For instance for the binary trees model which is often used as a clustering model:

- *homogeneity*: the minimal subtree containing the 2 objets
- *separability*: the split of the tree/cluster into 2 subtrees (the 2 sons).

The above notions of homogeneity and separability can be used for hypergraphs providing they have the following three properties:

- The hypergraph contains all the objets (the objects are part of the structure),
- The hypergraph is closed under intersection (so there is a minimal cluster containing any given subset of objects),
- Each vertex has at most 2 sons.

Email addresses: celia.chatel@lis-lab.fr (Célia Châtel), francois.brucker@lis-lab.fr (François Brucker), pascal.prea@lis-lab.fr (Pascal Préa)

This work was supported in part by ANR project DISTANCIA (ANR-17-CE40-0015).

Note that the above three properties do not assume the clusters to be disjoint, which allows overlapping cluster models. We will show in this paper that the most general hypergraph model admitting these tree properties is totally balanced hypergraphs. In order to prove this result we will follow a path similar to the one used by Lehel[9] for its characterization of totally balanced hypergraphs. We will exhibit an algorithm which construct a so called *binary* hypergraph and show that one can use any given totally balanced hypergraph as a guide to construct it.

The paper is organized as follows. After defining the structure used in this paper and recalling some known properties for totally balanced hypergraph (Section 2), we will give an efficient algorithm which can produce a binary hypergraph (Section 3). This algorithm is then modified in order to binarize a given totally balanced hypergraph, which will prove the main result (Section 4). We will finally conclude (Section 5).

2. Basic definitions and first result

This part will define the main structures we will use throughout this paper and prove Corollary 1, the first implication of the mapping between binary and totally balanced hypergraphs.

An *hypergraph* is a couple $H = (V, E)$ where V is a finite set whose elements are *vertices* and $E \subset 2^V$ is the *hyperedge* set. Throughout this paper, we only consider hypergraphs such that $V \in E$ and $\forall x \in V, \{x\} \in E$. As a clustering model, these hypergraphs are called *set systems* [3]. Hypergraphs and set systems will be equivalent here. We write $u \parallel v$ if neither $u \subseteq v$ nor $v \subseteq u$. For $e_1, e_2 \in E$, we say that e_2 *covers* e_1 (or that e_1 is *covered* by e_2) and write $e_1 \prec_E e_2$ (or $e_1 \prec e_2$ if there is no confusion) if $e_1 \subsetneq e_2$ and there exists no $e_3 \in E$ such that $e_1 \subsetneq e_3$ and $e_3 \subsetneq e_2$.

We will moreover often use *closed* hypergraph. An hypergraph is said to be *closed under intersection* (or *closed* for short) if $\forall u, v \in E, u \cap v \neq \emptyset \implies u \cap v \in E$. The *closure* $\overline{H} = (V, \overline{E})$ of an hypergraph $H = (V, E)$ is the smallest closed hypergraph such that $E \subseteq \overline{E}$. The main interest of closed structures in clustering is that every set of hyperedges admits a supremum, where the *supremum* of $e_1, \ldots, e_p \in E$, written $\sup_E(e_1, \ldots, e_p)$ (or $\sup(e_1, \ldots, e_p)$ if there is no confusion) is, if it exists, the smallest (regarding the inclusion order) element $e \in E$ such that $\forall i \leq p, e_i \subseteq e$. If $e_i = \{x_i\}$, we will write $\sup(x_1, \ldots, x_p)$.

Generally speaking, the closure of a given hypergraph can be costly. It is not the case for totally balanced hypergraphs because they are weak-hierachies (Brucker and Gely [4]). *Weak-hierarchies* (Bandelt and Dress [2]) are defined as hypergraphs for which the intersection of 3 edges is always the intersection of two of them.

Weak hierarchies have numerous interesting properties for clustering (see for instance Diatta and Fichet [8] for an extensive study of them); as they only admit a small number of clusters (the square of the number of elements), their closure can be computed by only intersecting clusters pairwise. In addition, for a closed weak-hierarchy, each cluster is the supremum of two elements.

A *special cycle* is a sequence $(x_0, e_0, x_1, e_1, \ldots, x_{k-1}, e_{k-1})$ with $k \geq 3$, $x_i \in V$ and $e_i \in E$ for all $i \in \{0, \ldots, k-1\}$ and such that $x_i \in e_j$ if and only if $i = j$ or $i = j+1 \mod k$. A *totally balanced hypergraph* is an hypergraph with no special cycle.

We say that a hypergraph $H = (V, E)$ is a *hypertree* if there exists a tree $T = (V, E')$ such that every hyperedge of H is the set of vertices of a connected subtree of T. We name T a *support tree* of H.

The *subhypergraph* of $H = (V, E)$ *induced* by a set $A \subseteq V$ is the hypergraph $H|_A = (A, \{e \cap A : e \in E\})$. The following theorem gives a characterization of totally balanced hypergraphs by their induced subhypergraphs.

Theorem 1 (Lehel [9])**.** *A hypergraph H is totally balanced if and only if every subhypergraph of H is a hypertree.*

Note that if H is a totally balanced hypergraph so are all its subhypergraphs and restrictions. Moreover:

Property 1. *Let $H = (V, E)$ be a hypergraph. H is totally balanced iff $\overline{H}$ is totally balanced.*

Proof. Let H be a hypergraph with a special cycle $C = (x_0, e_0, x_1, \ldots, x_{k-1}, e_{k-1})$. For all $i \leq k-1$, $e_i \in \overline{E}$ hence C is a special cycle in $\overline{H}$. So $\overline{H}$ totally balanced implies H totally balanced.

Conversely, let $H = (V, E)$ be a totally balanced hypergraph and $\overline{H} = (V, \overline{E})$ its closure. If $\overline{H}$ is not totally balanced, it contains a special cycle $C = (x_0, e_0, x_1, \ldots, x_{k-1}, e_{k-1})$. At least one of the e_i is in $\overline{E} \setminus E$ (otherwise, C would be a special cycle of H). We suppose, with no loss of generality, that C has the smallest number (among all special cycles of $\overline{H}$) of hyperedges in $\overline{E} \setminus E$ and that $e_0 \in \overline{E} \setminus E$; so $e_0 = e_0^1 \cap \ldots \cap e_0^p$, with $p > 1$ and $e_0^i \in E$ for $i \leq p$. At least one of the e_0^i (say e_0^1) does not contain x_{k-1} (otherwise $x_{k-1} \in e_0$, which is impossible). Let k' be the greatest index such that $x_{k'} \in e_0^1$; we have $1 \leq k' < k-1$. The cycle $(x_0, e_0^1, x_{k'}, e_{k'}, \ldots, x_{k-1}, e_{k-1})$ is a special cycle of $\overline{H}$ containing less hyperedges in $\overline{E} \setminus E$ than C, which is a contradiction. □

We will call a special cycle of the form $(x_0, \sup(x_0, x_1), x_1, \sup(x_1, x_2), \ldots x_{k-1}, \sup(x_{k-1}, x_0))$ a *simple cycle*.

Claim 1. *Let H be a closed hypergraph. If $(x_0, e_0, x_1, e_1, \ldots, x_{k-1}, e_{k-1})$ is a special cycle of H, then $(x_0, \sup(x_0, x_1), x_1, \sup(x_1, x_2), \ldots x_{k-1}, \sup(x_{k-1}, x_0))$ is a special cycle of H.*

Proof. By definition, $x_i, x_{i+1} \in \sup(x_i, x_{i+1})$. As $\sup(x_i, x_{i+1}) \subset e_i$, if $j \neq i, i+1$, $x_j \notin \sup(x_i, x_{i+1})$. □

We can now introduce the link between *binary* hypergraphs and totally balanced hypergraphs. A hypergraph $H = (V, E)$ is said to be *binary* if it is closed and:

$$\forall u \in E, |\{v \in E : v \prec u\}| \leq 2$$

A hypergraph $H = (V, E)$ is *binarizable* if there exists a binary hypergraph $H' = (V, E')$ with $E \subseteq E'$.

Property 2. *Let $H = (V, E)$ be a binary hypergraph then it is totally balanced.*

Proof. The proof will be by induction on the size of the simple cycle.

Suppose that $H = (V, E)$ is a binary hypergraph with a simple 3-cycle $(x_0, \sup(x_0, x_1), x_1, \sup(x_1, x_2), x_2, \sup(x_2, x_0))$ and let $e = \sup(x_0, x_1, x_2)$. As H is binary (and closed), $e \in E$ and covers at most two hyperedges e' and e''. As $e = \sup(\sup(x_0, x_1), \sup(x_1, x_2), \sup(x_2, x_0))$, we can suppose, with no loss of generality, that $\sup(x_1, x_2) \subseteq e'$ and $\sup(x_2, x_0) \subseteq e'$. So $x_0 \in e'$ and thus $x_0, x_1, x_2 \in e' \subsetneq e = \sup(x_0, x_1, x_2)$, which is a contradiction.

Suppose now that every hypergraph with a simple k-cycle ($k \geq 3$) is not binary. Let $H = (V, E)$ be a binary hypergraph with a simple $(k+1)$-cycle $(x_0, \sup(x_0, x_1), \ldots, x_k, \sup(x_k, x_0))$ and let $e = \sup(x_0, \ldots, x_k)$. At most two hyperedges e' and e'' are covered by e. Let $X' = \{x_i : x_i \in e'\}$ and $X'' = \{x_i : x_i \in e''\}$. Since $X' \cup X'' = \{x_0, \ldots, x_k\}$ and $X', X'' \neq \{x_0, \ldots, x_k\}$, we can suppose, with no loss of generality, that $|X''| > 2$ and that $x_0 \notin X''$. Let $u = \min\{i \in \{0, \ldots k\} : x_i \in e''\}$ and $v = \max\{i \in \{0, \ldots k\} : x_i \in e''\}$. Since $|v - u| > 1$, the cycle $(x_0, \sup(x_0, x_1), x_1, \ldots, x_u, e'', x_v, \ldots, x_k, \sup(x_k, x_0))$ is a special cycle of length $\leq k$. By the induction hypothesis, H is not binary, a contradiction. □

Corollary 1. *Let $H = (V, E)$ be a hypergraph. If H is binarizable then H is totally balanced.*

Proof. Let H be a binarizable hypergraph and H' a binary hypergraph containing H. Since every special cycle of H is a special cycle of H', by Property 2, H can not have a special cycle. □

In order to prove the other implication (Section 4) we will first introduce an algorithm which produces binary hypergraphs.

3. An algorithm to construct binary hypergraphs

We propose in this section a (non deterministic) procedure which constructs binary hypergraphs by generating a sequence of mixed trees. This section proves that this procedure construct a binary hypergraph (Theorem 3) and Section 4 will show that one can in fact construct any binary hypergraph.

This procedure is made of three parts :

- Algorithm 1 (Basic-Tree-Construction) return a mixed tree T_{i+1} and a S_{i+1} constructed from a mixed tree T_i and a map S_i.
- Algorithm 2 (Tree-Sequence-Construction) puts Algorithm 1 into a loop to construct a sequence of consistent mixed trees. It begins with a given mixed tree $T_0 = (V_0, E_0, \emptyset)$ and $S_0(x) = \{x\}$ for $x \in V_0$.
- when Algorithm 1 ends, the sequence of mixed-trees $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ is merged into the hypergraph $H = (V_0, E)$ with $E = \bigcup_{0 \leq i \leq p} \{S_i(v) : v \in V_i\}$, which is binary.

Before entering into the details of the procedure, let us define a *mixed tree*. A *mixed graph* is a triplet $G = (V, E, \overrightarrow{E})$ such that $G_1 = (V, E)$ is an undirected graph and $G_2 = (V, \overrightarrow{E})$ is a directed graph. A *mixed tree* is a mixed graph such that the undirected underlying graph obtained by replacing all directed edges of the graph by undirected edges is a tree. We will denote $xy \in E$ (resp. $xy \in \overrightarrow{E}$) if $\{x, y\}$ (resp. (x, y)) is an undirected (resp. directed) edge of $G = (V, E, \overrightarrow{E})$. For $x \in V$, we define $\Delta(x) = \{y \in V : xy \in E\}$, $\Delta^+(x) = \{y \in V : xy \in \overrightarrow{E}\}$, $\Delta^-(x) = \{y \in V, yx \in \overrightarrow{E}\}$ and $\overline{\Delta(x)} = \Delta(x) \cup \Delta^+(x) \cup \Delta^-(x)$.
A mixed tree is said to be *consistent* if :

- for every vertex x, $\Delta^+(x) \neq \emptyset \implies \Delta(x) \neq \emptyset$,
- there does not exist x, y, z such that xy and yz are in $\overrightarrow{E}$

Given a mixed tree $T = (V, E, \overrightarrow{E})$, a *path* of T is a sequence of vertices $x_1, \ldots, x_k$ such that for all $i < k$, $x_{i+1} \in \overline{\Delta(x)}$, i.e x_i and x_{i+1} are neighbors in the undirected underlying graph. Similarly, a subgraph of a mixed tree is *connected* if it is connected in the undirected underlying graph.

Algorithm 1: Basic-Tree-Construction

Input: A consistent mixed tree $T = (V, E, \overrightarrow{E})$ with a map S from V to 2^X where X is a finite set
Output: A consistent tree $T' = (V', E', \overrightarrow{E'})$ with a map S' from V' to 2^X

```
1  begin
2      Choose xy ∈ E such that Δ⁻(x) = Δ⁻(y) = ∅
3      V' ← V ∪ {v_xy}
4      S'(v_xy) ← S(x) ∪ S(y) ; S'(u) ← S(u) ∀u ∈ V
5      E' ← E \ {xy}
6      E⃗' ← E⃗
7      for z ∈ {x, y} do
8          E⃗' ← E⃗' ∪ {z v_xy}
9          Choose Δ'(z) ⊆ Δ(z)
10         E' ← E ∪ {v_xy u : u ∈ Δ'(z)} \ {zu : u ∈ Δ'(z)}
11         if Δ'(z) = Δ(z) then
12             Let T_Δ = (Δ⁺(z), E_Δ) be a tree on vertex set Δ⁺(z)
13             E' ← E' ∪ E_Δ
14             V' ← V' \ {z}
15             E⃗' ← E⃗' \ {zu : u ∈ Δ⁺(z)}
16     return T' = (V', E', E⃗'), S'
```

Algorithm 1 is not deterministic. Depending on the choices made at lines 2, 9 or 12, we get a different mixed tree T' thus, *in fine*, a different hypergraph. Figure 1 shows two different runs of Algorithm 1 for the same initial mixed-tree.

We will now prove (Claim 3 which uses Claim 2) that Algorithm 1 is correct.

Algorithm 2: TREE-SEQUENCE-CONSTRUCTION

Input: A (consistent) mixed tree $T_0 = (V_0, E_0, \emptyset)$.

Output: A sequence $\mathcal{T} = ((T_0, S_0), (T_1, S_1), \ldots, (T_p, S_p))$, where, $\forall i \leq p, T_i$ is a consistent mixed tree $(V_i, E_i, \overrightarrow{E_i})$ and S_i a map from V_i to 2^{V_0}.

```
1 begin
2     ∀x ∈ V_0, S_0(x) ← {x}
3     T ← ((T_0, S_0))
4     (T, S) ← (T_0, S_0)
5     while |V(T)| > 1 do
6         (T, S) ← Basic-Tree-Construction(T, S)
7         Append (T, S) to T
8     return T
```

Claim 2. *A consistent mixed tree with more than one vertex contains an edge satisfying the condition of Line 2 of Algorithm 1.*

Proof. We will prove it by induction on the number of vertices $|V|$. Since a consistent mixed tree with 2 vertices contains one undirected vertex, the property is true for $|V| = 2$. Suppose that the property is true for $2 \leq |V| \leq k$ and consider a consistent mixed tree $T = (V, E, \overrightarrow{E})$ with $k+1$ vertices. Since a consistent mixed tree with 2 or more vertices contains at least one undirected edge, let $xy \in E$. If this undirected edge does not satisfy the condition of Line 2 of Algorithm 1, one can consider without loss of generality that there exists $x'x \in \overrightarrow{E}$. Deleting this edge leads to 2 consistent mixed trees, one containing x' and the other containing x. Since x' cannot be a leaf of the undirected underlying tree of T, the consistent mixed tree containing x' has more than 2 vertices thus satisfy the induction hypothesis: there exists an edge satisfying the condition of Line 2 of Algorithm 1 in this mixed tree. This edge clearly also satisfies the condition for T. □

Claim 3. *Algorithm 1 is correct: with a consistent mixed tree (having more than one vertex) as entry, it returns a consistent mixed tree.*

Proof. Claim 2 shows that one can always find an edge xy satisfying conditions of Line 2. It suffices now to show that T' is also a consistent mixed tree. The underlying graph of T' is clearly a tree. Moreover, the only oriented edge creation is at Line 8. In this case $\Delta(z) \setminus \Delta'(z) \neq \emptyset$ thus $\Delta(z) \neq \emptyset$ for T'. □

Since algorithm 1 is correct, one can now prove that Algorithm 2 stops and that the final sequence $\mathcal{T}$ is such that $H = (V_0, \bigcup_{0 \leq i \leq p} \{S_i(v) : v \in V_i\})$ is a binary hypergraph. Figure 2 shows a run of Algorithm 2 and Figure 3 the resulting binary hypergraph.

The proof (Theorem 2 and Theorem 3) will need some lemmas. Lemma 1 which is the keystone of the proof and two technical lemmas, Lemma 2 and Lemma 3.

Lemma 1. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p), \ldots)$, with $T_i = (V_i, E_i, \overrightarrow{E_i})$ for all i, be a sequence of mixed trees and sets obtained by Algorithm* TREE-SEQUENCE-CONSTRUCTION*. For all i:*

(i) $\forall \alpha \in V_0$, $X_i^\alpha = \{v \in V_i : \alpha \in S_i(v)\}$ *is a connected part of T_i;*

(ii) $uv \in \overrightarrow{E_i} \implies S_i(u) \subsetneq S_i(v)$*;*

(iii) $uv \in E_i \implies S_i(u) \parallel S_i(v)$.

Proof. The proof will be by induction. The property is trivially true for T_0.

Suppose Properties (i), (ii) and (iii) are verified for T_i. Let $V_{i+1} \setminus V_i = \{v_{xy}\}$ be the vertex created in T_{i+1} by contracting $xy \in E_i$. We have $S_{i+1}(v_{xy}) = S_i(x) \cup S_i(y)$.

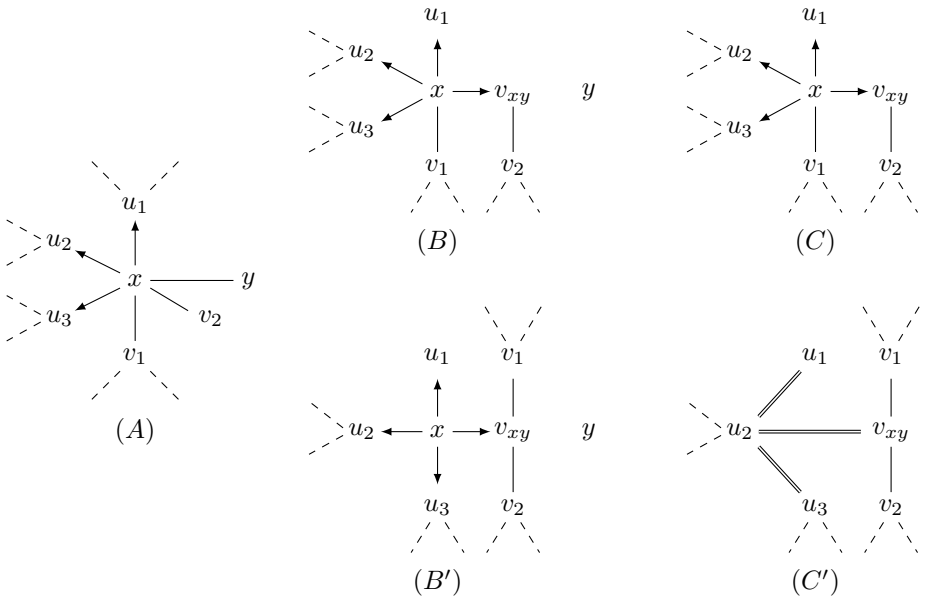

Figure 1: Two runs of Algorithm 1 on the same graph, with the same edge xy chosen. In both cases, as x is the only neighbor of y, y will be suppressed at Line 14.
In the first run ($A \to B \to C$), at Line 9, we choose $\Delta'(x) = \{v_2\}$ and so, at Line 10, v_2 becomes a neighbor of v_{xy} (B). Vertex y is deleted to give (C). In the second run ($A \to B' \to C'$), at Line 9, we chose $\Delta'(x) = \Delta(x)$. So, at Line 12, we create a tree T_Δ on $\{u_1, u_2, u_3, v_{xy}\}$ whose edges are drawn with a double line in (C'), and vertex x is suppressed at Line 14.

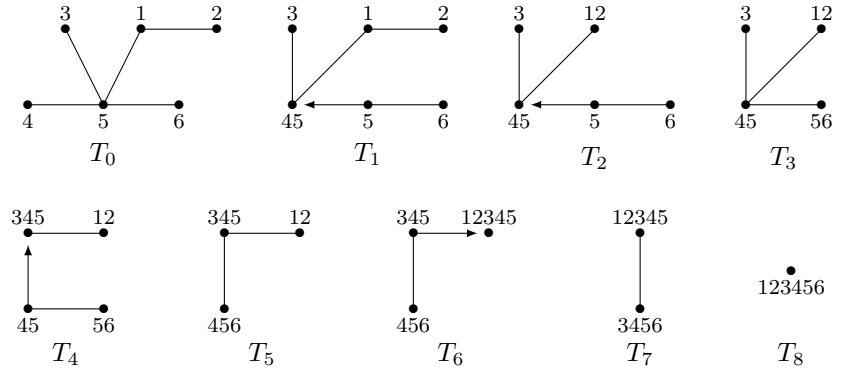

Figure 2: A sequence of mixed trees obtained by Algorithm 2

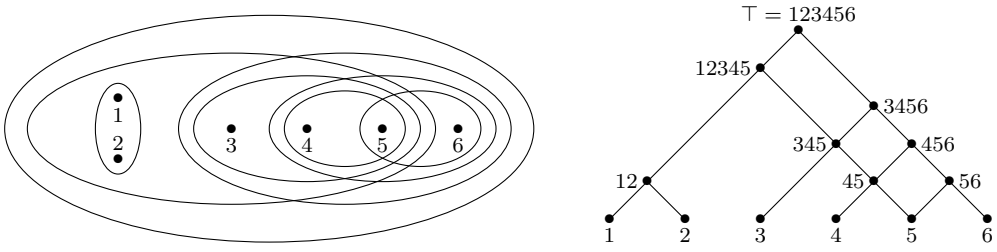

Figure 3: The binary hypergraph obtained by the sequence of mixed trees of Figure 2, represented as sets (left) and as a lattice (right).

(i) Let $\alpha \in V_0$. If $\alpha \notin S_i(x)$ and $\alpha \notin S_i(y)$, $X^\alpha_{i+1} = X^\alpha_i$ and the edges inside X^α_i are not changed by the construction of T_{i+1}. So X^α_{i+1} is a connected part of T_{i+1}.
If $\alpha \in S_i(x)$ or $\alpha \in S_i(y)$, $\alpha \in S_{i+1}(v_{xy})$. Since the only edge changes from T_i to T_{i+1} are edges xz or yz which become $v_{xy}z$, X^α_{i+1} is connected.

(ii) The only oriented edges that can be created when constructing T_{i+1} from T_i are xv_{xy} and yv_{xy}, thus Property (ii) is true for (T_{i+1}, S_{i+1}).

(iii) The only non-oriented edges that can be created when constructing T_{i+1} from T_i are:

- uv_{xy} with $xu \in E_i$ (symmetrically, $yu \in E_i$) at Line 10. In this case, $\exists \alpha \in S_i(u) \setminus S_i(x)$; by Property (i), $\alpha \notin S_i(y)$ and thus $S_i(u) \not\subset S_{i+1}(v_{xy})$. Since $xy \in E_i$, $\exists \beta \in S_i(y) \setminus S_i(x)$; by Property (i), $\beta \notin S_i(u)$ and thus $S_{i+1}(v_{xy}) \not\subset S_i(u)$.
- Edges of E_A (Lines 12 and 13). By Property (ii), for each element u of $\Delta^+(z)$, $S_i(u)$ contains an element α_u not in $S_i(z)$. So does $S_{i+1}(v_{xy})$. By Property (i), for all $u \in V_A$, $\alpha_u \notin S_i(u')$ for all $u' \in V_A, u' \neq u$. So, for each edge $uu' \in E_A$, $S_i(u) \parallel S_i(u')$.

So the three properties are verified for T_{i+1}. □

For $i \geq 0$, let u be a vertex of T_i and v be a vertex of T_{i+1}. We say that v is a *child* of u if either u remains unchanged between T_i and T_{i+1} and $v = u$ or $v = v_{uy}$ (see Line 3 of Algorithm 1). A *descendant* of a vertex u is either u or a child of a descendant of u.

Claim 4. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p), \ldots)$, with $T_i = (V_i, E_i, \overrightarrow{E_i})$ for all i, be a sequence of mixed trees and sets obtained by Algorithm* Tree-Sequence-Construction. *If a vertex v of T_i is a descendant of a vertex u of T_j $(j < i)$, then $S_j(u) \subsetneq S_i(v)$ if $u \neq v$.*

Proof. If v is a descendant of u there exists a chain $u_0, \ldots, u_p$ with $u_0 = u$ and $u_p = v$ such that u_{k+1} is a vertex of T_{j+k+1} and is the son of u_k (which is a vertex of T_{k+j}) for all $0 \leq k < p$. Thus $S_{j+k}(u_k) \subseteq S_{j+k+1}(u_{k+1})$ for all $0 \leq k < p$. □

Lemma 2. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p), \ldots)$ be a sequence of mixed trees and sets obtained by Algorithm* Tree-Sequence-Construction. *For $i \geq 0$, let $V_{i+1} \setminus V_i = \{v_{xy}\}$ with $\alpha \in S_i(x) \setminus S_i(y)$ and $\beta \in S_i(y) \setminus S_i(x)$. For $j > i$, a vertex u of V_j is such that $S_j(u)$ contains both α and β if and only if u is a descendant of v_{xy}.*

Proof. The "if" part follows directly from Claim 4.

By Lemma 1-i and by construction, v_{xy} is the only vertex u of V_{i+1} such that $\alpha, \beta \in S_u$. Let $j > i$ be the smallest integer such that there exists $u \in V_j$ which is not a descendant of v_{xy} and $\alpha, \beta \in S_j(u)$. The vertex u does not exist in T_{j-1}, so $u = v_{zt}$ with $\alpha \in S_{j-1}(z) \setminus S_{j-1}(t)$ and $\beta \in S_{j-1}(t) \setminus S_{j-1}(z)$. Let $w \in V_{j-1}$ be a descendant of v_{xy}. By Lemma 1-i, there exist in T_{j-1} a path from z to w which does not contain t and a path from t to w which does not contain z. As zt is an edge of T_{j-1} and T_{j-1} is a tree, this is a contradiction. □

Lemma 3. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p), \ldots)$ be a sequence of mixed trees and sets obtained by Algorithm* Tree-Sequence-Construction. *For $0 \leq i < j$ with $V_{i+1} \setminus V_i = \{v_{xy}\}$ and $V_{j+1} \setminus V_j = \{v_{zt}\}$, we have $S_{i+1}(v_{xy}) \neq S_{j+1}(v_{zt})$.*

Proof. We suppose that the property is false and take $j > i$ with $S_{i+1}(v_{xy}) = S_{j+1}(v_{zt})$. Let t be a descendant of v_{xy} in T_{j+1}. Since neither u nor v can be a descendant of v_{xy} (because $S_j(v_u) \subsetneq S_{j+1}(v_{uv})$ and $S_j(v_v) \subsetneq S_{j+1}(v_{uv})$), $t \neq v_{uv}$ and t is not a descendant of v_{uv}. Since there exist $\alpha \in S_i(x) \setminus S_i(y)$ and $\beta \in S_i(y) \setminus S_i(x)$ and $\alpha, \beta \in t$ there is a contradiction with Lemma 2 □

Theorem 2. *Algorithm* Tree-Sequence-Construction *stops. Let $\mathcal{T}$ be the final sequence. The last tree of $\mathcal{T}$ is $T = (\{u\}, \emptyset, \emptyset)$ with $S_u = V_0$.*

Proof. From Claim 3, if $|V(T)| > 1$, line 6 of Algorithm 2 will always produce a new consistent mixed tree. But Lemma 3 argues that each new set produced is a different set from 2^{V_0}: the Algorithm 2 will stop. At this step $T = (\{u\}, \emptyset, \emptyset)$ □

Remark that Algorithm 2 stops, even if we suppress Line 4 from Algorithm 1, *i.e.* without the maps S_i. One can now prove that the resulting hypergraph is binary.

Lemma 4. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and sets obtained by Algorithm* Tree-Sequence-Construction. *For $0 \leq i \leq p$, let $(x_0, x_1, \ldots, x_k)$ be a path of T_i. We have:*

$$S_i(x_0) \cap S_i(x_k) \subseteq S_i(x_0) \cap S_i(x_{k-1}) \subseteq \ldots \subseteq S_i(x_0) \cap S_i(x_1)$$

Proof. Follows immediately from Lemma 1-i. □

One can now prove the main result of the part, Theorem 3:

Theorem 3. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and sets obtained by Algorithm* Tree-Sequence-Construction. *The hypergraph $H = (V_0, E)$ with $E = \bigcup_{0 \leq i \leq p} \{S_i(v) : v \in V_i\}$ is binary.*

Proof. We first show that H is closed under intersection, and more precisely, we show by induction on i that, $\forall 0 \leq i \leq p$, $\bigcup_{0 \leq j \leq i} \{S_j(v) : v \in V_j\}$ is closed.

This is obviously true for $i = 0$. Suppose now that the property is true for some $i \geq 0$, and let $V_{i+1} \setminus V_i = \{v_{xy}\}$. For $j \leq i$, let z be a vertex of V_j.

By induction hypothesis, $S_j(z) \cap S_i(x)$ and $S_j(z) \cap S_i(y)$ are elements of $\bigcup_{0 \leq j \leq i} \{S_j(v) : v \in V_j\}$. Let z' be a descendant of z in V_i. By Lemma 4, we can suppose that $S_i(z') \cap S_i(x) \subset S_i(z') \cap S_i(y)$. As $S_j(z) \subset S_i(z')$, $S_j(z) \cap S_i(x) \subset S_j(z) \cap S_i(y)$. So $S_j(z) \cap S_{i+1}(v_{xy}) = S_j(z) \cap S_i(y) \in \bigcup_{0 \leq j \leq i} \{S_j(v) : v \in V_j\} \subset \bigcup_{0 \leq j \leq i+1} \{S_j(v) : v \in V_j\}$.

We now show that H is binary. Let $V_{i+1} \setminus V_i = \{v_{xy}\}$ and $z \in V_j$, $j \leq i$ be such that $S_j(z) \subsetneq S_{i+1}(v_{xy})$. Let $t \in V_i$ be a descendant of z. By Lemma 4, we can suppose with no loss of generality that $S_i(y) \cap S_i(t) \subseteq S_i(x) \cap S_i(t)$. So we have $S_i(y) \cap S_j(z) \subseteq S_i(x) \cap S_j(z)$. As $S_j(z) \subsetneq S_{i+1}(v_{xy}) = S_i(x) \cup S_i(y)$, we have $S_j(z) \subsetneq S_i(x)$. So $S_{i+1}(v_{xy})$ covers exactly two other elements of E, namely $S_i(x)$ and $S_i(y)$. □

Corollary 2. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and sets obtained by Algorithm* Tree-Sequence-Construction. *Every hypergraph $H = (V_0, E)$ with $E \subseteq \bigcup_{0 \leq i \leq p} \{S_i(v) : v \in V_i\}$ is totally balanced.*

The following Proposition shows the complexity of Algorithm 2.

Proposition 1. *Algorithm 2 runs in $O(n^3)$, where $n = |V_0|$.*

Proof. Les X be a set and $\mathcal{T}(X)$ the set of all the consistent mixed-trees admitting a map S from X to 2^X satisfying the conditions of Lemma 1. We prove by induction on $|X|$ that the number of vertices of those trees cannot exceed $2 \cdot |X|$.

For $|X| = 1$ the property is trivially true. Suppose it true for $|X| \leq n$ and consider $|X| = n + 1$. Let $T(V, E, \overrightarrow{E}) \in \mathcal{T}(X)$.

Let x be a leaf of T. Two cases may occur : either $xy \in E$ or $yx \in \overrightarrow{E}$. The set $S(x) \backslash S(y)$ is then not empty and for all $z \in V \backslash \{x\}$: $S(z) \cap S(x) \backslash S(y) = \emptyset$. The set $X' = \cup_{z \in V \backslash \{x\}} S(z)$ is then strictly included in X thus $|X'| \leq n$.

Note T' the restriction from T to $V \backslash \{x\}$, if it is a consistent mixed tree. If the restriction from T to $V \backslash \{x\}$ is not a consistent mixed tree, then y is a leaf with $yz \in \overrightarrow{E}$ $(z \neq x)$ and one can note T' the restriction of T to $V \backslash \{x, y\}$ which is consistent (there cannot exist z' such that $zz' \in \overrightarrow{E}$). In both cases, T' is a consistent mixed tree associated with a map S from X' to $2^{X'}$. Since $|X'| < |X|$, we have that $V \leq 2 + 2 \cdot |X'| \leq 2 \cdot |X|$ which concludes the proof by induction.

Moreover, a totally balanced hypergraph have at most $\frac{n^2+n}{2}$ hyperedges (see for instance [5]); since Algorithm 2 adds a new hyperedge of the hypergraph at each step there are at most $\mathcal{O}(n^2)$ calls of Algorithm 1. As Algorithm 1 is linear in the size of the input which is always $\mathcal{O}(n)$, Algorithm 2 runs in $O(n^3)$ □

Remark that Algorithm 2 is really efficient since it is linear in the size of the resulting hypergraph.

4. Equivalence between binarizable and totally balanced hypergraphs

In this section, we will show that, given a totally balanced hypergraph H, it is possible to obtain a binarization of H as the result of slightly modified versions of Algorithms 1 and 2, namely Algorithm 3 and Algorithm 4.

The difference lies in the fact that the random choices of algorithm 1 (lines 2, 9 and 12) are in Algorithm 3 directed by the given closed totally balanced hypergraph H (lines 2, 9 and 13). Figure 4 shows a run of Algorithm 4. The resulting binary hypergraph is the one of Figure 3.

Algorithm 3: BASIC-TREE-CONSTRUCTION-H

Input: A consistent mixed tree $T = (V, E, \overrightarrow{E})$ with a map S from V to 2^X where X is a finite set and a closed totally balanced hypergraph $H = (X, E_H)$.

Output: A consistent tree $T' = (V', E', \overrightarrow{E'})$ with a map S' from V' to 2^X

1 **begin**
2 Choose $xy \in E$ such that $\Delta^-(x) = \Delta^-(y) = \emptyset$
3 $V' \leftarrow V \cup \{v_{xy}\}$
4 $S'(v_{xy}) \leftarrow S(x) \cup S(y)$; $S'(u) \leftarrow S(u)\ \forall u \in V$
5 $E' \leftarrow E \setminus \{xy\}$
6 $\overrightarrow{E'} \leftarrow \overrightarrow{E}$
7 **for** $z \in \{x, y\}$ **do**
8 $\overrightarrow{E'} \leftarrow \overrightarrow{E'} \cup \{zv_{xy}\}$
9 $\Delta'(z) \leftarrow \{t : zt \in E, S(v_{xy}) \subseteq \sup_{E_H}(S(z), S(t))\}$
10 $E' \leftarrow E' \cup \{v_{xy}u : u \in \Delta'(z)\} \setminus \{zu : u \in \Delta'(z)\}$
11 **if** $\Delta'(z) = \Delta(z)$ **then**
12 $A \leftarrow \bigcup_{t \in \Delta^+(z)} S_i(t) \setminus S_i(z)$
13 $T_A = (A, V_A) \leftarrow$ a support tree of $H|_A$
14 $s(\alpha) \leftarrow$ the (unique) element u of $\Delta^+(z)$ such that $\alpha \in S(u)$, $\forall \alpha \in A$
15 $E_\Delta = \{s(\alpha)s(\beta) : \alpha\beta \in V_A\}$
16 $E' \leftarrow E' \cup E_\Delta$
17 $V' \leftarrow V' \setminus \{z\}$
18 $\overrightarrow{E'} \leftarrow \overrightarrow{E'} \setminus \{zu : u \in \Delta^+(z)\}$
19 **return** $T' = (V', E', \overrightarrow{E'}), S'$

In order to show that algorithm 4 stops, we only have to prove that Algorithm 3 is correct, that is that:

1. one can always find a support tree of $H|_A$ at Line 13. It is clear by Theorem 1.
2. for all $\alpha \in A$, there exists a unique $u \in \Delta^+(z)$ such that $\alpha \in S(u)$ at Line 14. This is true because by Lemma 1-*(i)*, (which holds since Algorithms 3 and 4 are only variants of Algorithms 1 and 2), $X^\alpha := \{v \in V : \alpha \in S(v)\}$ is a connected part of T. As $\alpha \notin S(z)$, there is only one neighbor u of z such that $\alpha \in S(u)$.

We now prove that Algorithm 4 constructs a binary hypergraph for which H is a sub-hypergraph (Theorem 4 which uses Lemma 5). This is the contraposition of Corollary 1.

Lemma 5. *Let $H = (V_H, E_H)$ be a closed totally balanced hypergraph and $\mathcal{T} = (T_0, \ldots, T_p)$, with $T_i = (V_i, E_i, \overrightarrow{E_i})$ $\forall i \in \{0, \ldots, p\}$, be a sequence of mixed trees and sets obtained by Algorithm* TREE-SEQUENCE-CONSTRUCTION-H. *Then $\forall e \in E_H, i \in \{0, \ldots, p\}$, $\Psi_i^e := \{v \in V_i : S_i(v) \subseteq e\}$ is a connected part of T_i and $\bigcup_{v \in \Psi_i^e} S_i(v)$ is either empty or equal to e.*

Algorithm 4: TREE-SEQUENCE-CONSTRUCTION-H

Input: A totally balanced hypergraph $H = (V_0, E_H)$ and one of its support tree $T = (V_0, E_0)$
Output: A sequence $\mathcal{T} = ((T_0, S_0), (T_1, S_1), \ldots, (T_p, S_p))$, where, $\forall i \leq p, T_i$ is a consistent mixed tree $(V_i, E_i, \overrightarrow{E_i})$ and S_i a map from V_i to 2^{V_0}.

```
1  begin
2      E→_0 ← ∅
3      ∀x ∈ V_0, S_0(x) ← {x}
4      T ← ((T_0, S_0))
5      (T, S) ← (T_0, S_0)
6      while |V(T)| > 1 do
7          (T, S) ← Basic-Tree-Construction-H(T, S, H̄)
8          Append (T, S) to T
9      return T
```

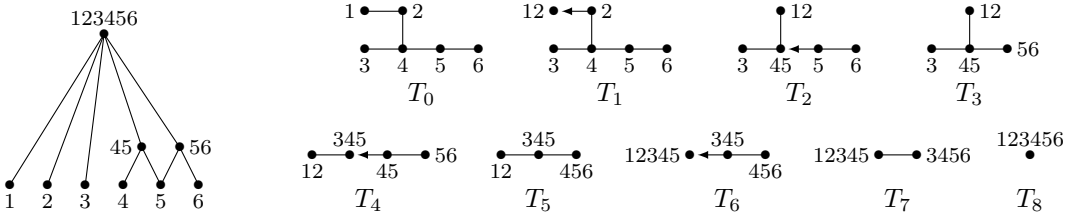

Figure 4: A totally balanced hypergraph, sub-hypergraph of the binary one of Figure 3, and a sequence of mixed trees which constructs this hypergraph.

Proof. We prove the property by induction on i. Since (V_0, E_0) is a support tree of H, the property is true for $i = 0$. We suppose now that, for some i, the property is true for all $i' \leq i$, and we set $v_{xy} := V_{i+1} \setminus V_i$. Let $e \in E_H$, Ψ_i^e is a connected part of T_i and $\bigcup_{v \in \Psi_i^e} S_i(v) = e$.

Several cases can occur:

$S_x \not\subset e$ and $S_y \not\subset e$.
In this case, $\Psi_{i+1}^e = \Psi_i^e$ and induces the same (connected) subgraph in T_{i+1} than in T_i. The two induction properties are thus satisfied.

$S_x \subseteq e$ and $S_y \subseteq e$.
In this case, $v_{xy} \in \Psi_{i+1}^e$. If $x \in V_{i+1}$ (symmetrically $y \in V_{i+1}$), neighbors of x which are in Ψ_i^e are, in T_{i+1}, neighbors of x or v_{xy}, which are both in Ψ_{i+1}^e. If $x \notin V_{i+1}$ (symmetrically $y \notin V_{i+1}$), v_{xy} is neighbor of all vertices in $\Delta(x)$, and so of all such vertices in Ψ_{i+1}^e. In addition, for all vertices u in $\Delta^+(x)$ since $S_x \subset S_u$, Line 13 of Algorithm 3 and the induction properties ensure that v_{xy} and the neighbors of x in Ψ_i^e induce a connected subgraph of T_{i+1}, thus Ψ_{i+1}^e is a connected subgraph of T_{i+1}. Moreover, since $x, y \in \Psi_i^e$, we have that $\bigcup_{v \in \Psi_{i+1}^e} S_i(v) = \bigcup_{v \in \Psi_i^e} S_i(v) \cup S_{i+1}(v_{xy}) = e \cup S_{i+1}(v_{xy}) = e$.

$S_x \subset e$ and $S_y \not\subset e$. (symmetrically, $S_x \not\subset e$ and $S_y \subset e$).
In this case, $v_{xy} \notin \Psi_{i+1}^e$ and Ψ_i^e is a subtree of T_i containing x and not y. In addition, for $t \in \Delta(x)$, if $S(t) \subset e$, $sup_{E_H}(S(x), S(t)) \subset e$ and thus $t \notin \Delta'(x)$. So, if $x \in V_{i+1}$, $\Psi_{i+1}^e = \Psi_i^e$ and induces the same (connected) subgraph in T_{i+1} that in T_i. If $x \notin V_{i+1}$ (*i.e.* $\forall t \in \Delta(x), S(t) \not\subset e$) and $\Delta_i^+(x) \neq \emptyset$, Lines 13–16 ensure that Ψ_{i+1}^e is connected; in addition, $\Psi_{i+1}^e = \Psi_i^e \setminus \{x\}$. As $S(x) \subset S(t)$ for $t \in \Delta^+(x)$, $\bigcup_{t \in \Psi_{i+1}^e} S(t) = \bigcup_{t \in \Psi_i^e} S(t) = e$. If $x \notin V_{i+1}$ and $\Delta_i^+(x) = \emptyset$, Ψ_{i+1}^e is empty. □

Theorem 4. *$\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and sets obtained by Algorithm 4 for a closed totally balanced hypergraph H. The hypergraph $H' = (V_0, E')$ with $E' = \bigcup_{0 \leq i \leq p} \{S_i(v) : v \in V_i\}$ is binary and such that $E \subseteq E'$.*

Proof. As Algorithm 4 is just an adaptation of Algorithm 2, by Theorem 3, the hypergraph H' is binary. Let $e \in E_H$, if $e = V_H$, then $V_p = \{e\}$ and $e \in E'$; otherwise, $\Psi_p^e = \emptyset$ and $\Psi_0^e \neq \emptyset$. By the proof of Lemma 5, the smallest i with $\Psi_i^e = \emptyset$ is such that $V_i \setminus V_{i-1} = v_{xy}$, $S_x \subset e$, $S_y \not\subset e$, $x \notin V_i$ and $\Delta^+(x) = \emptyset$. In this case, $\Psi_{i-1}^e = x$ and so $e = S(x) \in E'$. □

Finally, by Property 2 and Theorem 4, one can state the main result of our paper:

Theorem 5. *A hypergraph is totally balanced if and only if it is binarizable.*

5. Conclusion

We show in this paper that the only clustering model that is binarizable is the totally balanced hypergraph one. We also exhibit an algorithm that can binarize any given totally balanced hypergraph. Actually, the proof yields an algorithmic characterization like the one of Lehel[9], but using another form of vertices saturation (here binarization).

We now work on using these algorithms for practical cases. For instance use algorithm 4 for approximate a given hypergraph into a binary one and study the properties of this approximation. We also want to use Algorithm 2 in order to iteratively produce a binary hypergraph from real data.

References

[1] R.P. Anstee and M. Farber. Characterization of totally balanced matrices. *Journal of Algorithms*, 5:215–230, 1984.

[2] H.-J. Bandelt and W. M. Dress. Weak hierarchies associated with similarity measures – an additive clustering technique. *Bulletin of Mathematical Biology*, 51:133–166, 1989.

[3] P. Bertrand. Set systems and dissimilarities. *European Journal of Combinatorics*, 21:727–743, 2000.

[4] F. Brucker and A. Gély. Crown-free lattices and their related graphs. *Order*, 28:443–454, 2011.

[5] F. Brucker and P. Préa. Totally balanced formal concept representations. In J. Baixeries, C. Sacarea, and M. Ojeda-Aciego, editors, *Proceedings of ICFCA 2015*, pages 169–182. Springer, 2015.

[6] M. Dien. *Concurrent process and combinatorics of increasingly labeled structures: quantitative analysis and random generation algorithms.* PhD thesis, Pierre et Marie Curie University, Paris, France, 2017.

[7] M. Farber. Characterizations of strongly chordal graphs. *Discrete Mathematics*, 43:173–189, 1983.

[8] Diatta J. and Fichet B. From asprejan hierarchies and bandelt-dress weak hierarchies to quasi-hierarchies. In E. Diday, Y. Lechevallier, M. Schader, and P. Bertrand, editors, *New Approaches in classification and data analysis*, pages 111–118. Springer, 1994.

[9] J. Lehel. Characterization of totally balanced hypergraphs. *Discrete Mathematics*, 57:59–65, 1985.

[10] L. Lovász. Graphs and set systems. In H. Walther, H. Sachs, and H-J. Voss, editors, *Beiträge zur Graphentheorie*, volume 57, pages 99–106. Teubner, Leipzig, 1968.

[11] J. Spinrad. *Efficient Graph representations.* American Mathematical Society, 2003.