Binary set systems and totally balanced hypergraphs

Célia Châtel, François Brucker, Pascal Préa

Binary set systems and totally balanced hypergraphs[*][†]

Célia Châtel[‡] **François Brucker**[§] **Pascal Préa**[¶]
{celia.chatel, francois.brucker, pascal.prea}@lis-lab.fr

Abstract

A hypergraph H is *(i) Totally balanced* if it does not contain a special cycle, *(ii) Binary* if it is closed under intersection and every hyperedge has at most two predecessors (for inclusion order). We show in this paper that a hypergraph H is totally balanced if and only if it can be embedded into a binary hypergraph H'; H' is said to be a *binary extension* of H. We give an efficient algorithm which, given a totally balanced hypergraph H, produces a minimal binary extension $\widehat{H}$ of H; in addition, if H is a hierarchy or an interval hypergraph, then so is $\widehat{H}$.

keyword: hypergraphs, totally balanced hypergraphs, binary hypergraphs, clustering models.

1 Introduction

One of the aims of classification is to sort a data set V into clusters, which is equivalent to produce a hypergraph with vertex set V. In order to be interpretable, the produced clusters must share some common properties. These relationships are classically either structural or inherited. The first kind of relations implies the use of *clustering models* as the produced clusters belong to a particular class of hypergraph; the second kind uses *split models* since the clusters are produced by iteratively splitting them in two, beginning with the whole data set and ending with the singletons which cannot be further split.

The most used model for both aims is certainly the *hierarchical model*. It ensures that the intersection between two clusters is empty if one cluster is not included into the other one and thus applies as a structural constraint between the clusters. As a *hierarchical tree*, it also applies for a split model, for instance in phylogeny or in decision making.

The aim of phylogeny is to build phylogenetic trees. In fact, one wants to determine how a set of actual species evolved from a common ancestor and a cluster is interpreted as the ancestor of its elements. Each evolution event (the creation of a new species from an older one) corresponds to the split of a cluster into two new clusters. This procedure goes from the set of all species (their common ancestor) to singletons (the actual species). As a decision making process, binary hierarchical trees are called decision trees. It is a split model as each internal node corresponds to a decision relatively to a question/test and has two subtrees, one corresponding to the answer *yes* and one to the answer *no*. This model is widely used, for instance in machine learning.

However the hierarchical model does not take into account overlapping (*i.e.* when the intersection of two clusters may be neither empty nor one of them), which is sometimes needed in clustering problems. Several clustering models admitting overlapping have been designed like *interval hypergraphs* (clusters are intervals of a given order), used for instance in archaeology (see Robinson [13]) or *weak hierarchies* (clusters are generated by two elements only) used, among others, for biological problems (as originally stated in

[*]To appear in *Discrete Applied Mathematics*.
[†]This work was supported in part by ANR project DISTANCIA (ANR-17-CE40-0015).
[‡]Aix-Marseille Université, CNRS, Université de Toulon, LIS, Marseille, France.
[§]Aix-Marseille Université, CNRS, Université de Toulon, LIS and École Centrale Marseille, Marseille, France.
[¶]Aix-Marseille Université, CNRS, Université de Toulon, LIS and École Centrale Marseille, Marseille, France.

Bandelt and Dress [2]). To the best of our knowledge, the general split model where the splits can overlap has not been studied. We show in this paper that this general split model corresponds to a subset of *totally balanced hypergraphs*, the *binary hypergraph*, and that any totally balanced hypergraph can be extended into a hypergraph of this model.

Totally balanced hypergraphs, initially defined by Lovász [11], are a hypergraph structure which corresponds to the notion of tree for graphs (see Lehel [9]). They are used as a clustering model, since they generalize both the hierarchical and the interval hypergraphs models, are a subset of weak-hierarchies and admit a convenient graphical representation (Brucker and Préa [5]) but also in various applications like linear programming, phylogenetic problems (see Spinrad [14] for instance) or more recently in concurrent processes (see Dien [7]). We show in this paper that they can also be seen as a general split model minimizing the number of predecessors of each cluster.

The paper is organized as follows. After defining the structures used in this paper and recalling some known properties for totally balanced hypergraph (Section 2), we will show that totally balanced hypergraphs are exactly the hypergraphs which admit a binary extension (Section 3). We will then give an efficient algorithm which can produce any binary hypergraph (Section 4). It will be modified in Section 5 in order to give a binary extension for a given totally balanced hypergraph. Section 6 shows that binary hypergraphs minimize a criterion counting the predecessors of all the clusters. This section also shows that the algorithm of Section 5 runs efficiently as it builds a binary extension with the minimum number of clusters. Moreover, as shown in Section 7, this algorithm is stable for hierarchies and interval hypergraphs. Finally, Section 8 concludes the paper.

2 Basic definitions and classical results

In this part we define the main structures we will use throughout this paper and we recall two classical results on totally balanced hypergraphs that will be used in the following sections.

A *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 (we write $\subset$ for $\subseteq$). Throughout this paper, we only consider hypergraphs such that $V \in E$, $\emptyset \notin E$ 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 $u \subset v$ nor $v \subset u$. For $e_1, e_2 \in E$, we say that e_1 is a *predecessor* of e_2 (equivalently e_2 is a *successor* of e_1) 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 $e_1 \subsetneq e_3 \subsetneq e_2$.

A *special cycle* is a sequence $(x_0, e_0, x_1, e_1, \dots, x_{k-1}, e_{k-1})$ with $k \geq 3$, $x_i \in V$ and $e_i \in E$ for all $i \in \{0, \dots, k-1\}$ 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 a hypergraph with no special cycle.

A hypergraph $H = (V, E)$ is a *hypertree* if there exists a tree T with the same vertex set such that every hyperedge of H is the set of vertices of a subtree of T. We say then that T is a *support tree* of H. The *subhypergraph* of $H = (V, E)$ *induced* by a set $A \subset V$ 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.*

A direct consequence of Theorem 1 is that if $H = (V, E)$ is a totally balanced hypergraph so are all its subhypergraphs and all hypergraphs $H' = (V, E')$ with $E' \subset E$. Theorem 2 goes further on the link between trees and totally balanced hypergraphs as it gives a construction of maximum totally balanced hypergraphs by a sequence of trees. For ease of use it has been stated in the formalism of the rest of the paper.

Theorem 2 (Lehel [8, 9]). *Let $\mathcal{T} = (T_0, S_0), \dots, (T_{n-1}, S_{n-1})$ with $T_i = (V_i, E_i)$ and $S_i : V_i \longmapsto 2^{V_0}$ be a sequence of trees and maps such that $S_0(u) = \{u\}$ for any $u \in V_0$ and for every $0 \leq i < n$:*

1. $|S_i(u)| = i + 1$ for any $u \in V_i$

2. For any $u \in V_i$ there exists a unique $xy \in E_{i-1}$ such that $S_{i-1}(x) \cup S_{i-1}(y) = S_i(u)$

3. $uv \in E_i \implies S_i(u) = S_{i-1}(x) \cup S_{i-1}(z), S_i(v) = S_{i-1}(y) \cup S_{i-1}(z)$ for some $x, y, z \in V_{i-1}$

The hypergraph $H_\mathcal{T} = (V_0, E_\mathcal{T})$ with $E_\mathcal{T} = \bigcup_{0 \leq i < n} \{S_i(v) : v \in V_i\}$ is totally balanced. Moreover, a hypergraph $H = (V, E)$ is totally balanced if and only if there exists a tree sequence $\mathcal{T}$ such that $E \subset E_\mathcal{T}$.

Theorem 2 shows that any totally balanced hypergraph $H = (V, E)$ (with $|V| = n$ vertices) can be extended into a totally balanced hypergraph $H_\mathcal{T} = (V, E_\mathcal{T})$ with the same vertices and $|E_\mathcal{T}| = \binom{n+1}{2}$ hyperedges, which is the maximum possible for a given totally balanced hypergraph with n vertices (see for instance [1]). Also note that the totally balanced hypergraph given by Theorem 2 is closed. A 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 a hypergraph $H = (V, E)$ is the smallest closed hypergraph such that $E \subset \overline{E}$. Note that, by Theorem 1, H is totally balanced if and only if $\overline{H}$ is totally balanced.

Generally speaking, the closure of a given hypergraph can be costly in terms of number of clusters. It is not the case for totally balanced hypergraphs because they are weak hierarchies (Brucker and Gely [4]). *Weak-hierarchies* (Bandelt and Dress [2]) are defined as hypergraphs for which the intersection of three hyperedges is always the intersection of two of them. Weak hierarchies have numerous interesting properties for clustering (see for instance Diatta and Fichet [6] 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.

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 the unique smallest (regarding the inclusion order) element $e \in E$ such that $\forall i \leq p, e_i \subset e$. If $e_i = \{x_i\}$, we will write $\sup(x_1, \ldots, x_p)$. Note that if a hypergraph is not closed, one cannot define a supremum as there may exist several smallest elements containing $e_1, \ldots, e_p$; this is one of the reasons for which closed hypergraphs are widely used in clustering.

A hypergraph $H = (V, E)$ is said to be *binary* if it is closed and each hyperedge has at most two predecessors. Since the only hyperedges of a closed hypergraph which have strictly less than two predecessors are singletons (which have 0 predecessors since $\emptyset \notin E$), binary hypergraphs are closed hypergraphs such that any non singleton hyperedge has exactly two predecessors. Finally, a hypergraph $H = (V, E)$ is *binarizable* if there exists a binary hypergraph $H' = (V, E')$ with $E \subset E'$. We then say that H' is a *binary extension* of H.

3 Totally balanced hypergraphs and binary extensions

This section will show (Theorem 3) that binary hypergraphs are totally balanced and that every totally balanced hypergraph admits a binary extension.

Proposition 1. *Let H be a binary hypergraph, then it is totally balanced and closed.*

Proof. First note that if H is a closed hypergraph and $(x_0, e_0, x_1, e_1, \ldots, x_{k-1}, e_{k-1})$ 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 (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 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*. We now prove that a binary hypergraph cannot have a simple cycle by induction on the size of the 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 (thus closed), $e \in E$ and has at most two predecessors 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 $\sup(x_1, x_2) \subset e'$ and $\sup(x_2, x_0) \subset e'$. 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 $e := \sup(x_0, \ldots, x_k)$. Exactly two hyperedges e' and e'' are predecessors of e. Let $X' := \{x_i : x_i \in e'\}$ and $X'' := \{x_i : x_i \in e''\}$. Since $X' \cap X'' \neq \emptyset$, $X' \cup X'' = \{x_0, \ldots, x_k\}$ 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. $\square$

Corollary 1. *Let H be a hypergraph. If H is binarizable then H is totally balanced.*

Proof. Let H be a binarizable hypergraph and H' one of its binary extension. Since every special cycle of H is a special cycle of H', by Proposition 1, H cannot have a special cycle. $\square$

Proposition 2. *If $H_{\mathcal{T}} = (V_0, E_{\mathcal{T}})$ is a hypergraph as defined in Theorem 2, then $H_{\mathcal{T}}$ is a binary hypergraph.*

Proof. Let $e \in E_{\mathcal{T}}$, $|e| = r \geq 2$. Then $e \in V_{r-1}$, hence, by (1) and (2) of Theorem 2, there exist $x_1, x_2 \in V_0$, $x_1 \neq x_2$ such that $e_i = e \setminus \{x_i\} \in V_{r-2}$ for $i = 1, 2$. Considering the Hasse diagram of the poset of the edges of $H_{\mathcal{T}}$ defined by set inclusion, we conclude that e is a successor of both e_1 and e_2. If there was a third predecessor $e_3 \in E_{\mathcal{T}}$ of e, then $e_3 \parallel e_i$ for $i = 1, 2$. So $x_1, x_2 \in e_3$; furthermore, there exists $x_3 \in e \setminus e_3$. Then $(x_1, e_3, x_2, e_1, x_3, e_2)$ is a special cycle of $H_{\mathcal{T}}$, contradicting the claim in Theorem 2 that $H_{\mathcal{T}}$ is totally balanced. $\square$

By Theorem 2 and Proposition 2, we have:

Corollary 2. *Let H be a hypergraph with n vertices and $\binom{n+1}{2}$ hyperedges. Then H is totally balanced if and only if H is binary.*

Corollary 3. *Let H be a hypergraph. If H is totally balanced then H is binarizable.*

By Corollary 1 and Corollary 3, one can state the following:

Theorem 3. *A hypergraph is totally balanced if and only if it is binarizable.*

From Theorem 2 one can derive an algorithm which constructs a binary extension $H' = (V, E')$ of a given totally balanced hypergraph $H = (V, E)$. This extension maximizes the number of hyperedges: $|E| \leq |E'| = \frac{n(n+1)}{2}$ where $n = |V|$.

In the following Sections 4 and 5, we will give another algorithm which constructs a binary extension $\widehat{H}$ of H. This binary extension has the minimum the number of hyperedges among all the binary extensions of H (Theorem 8). Our construction gives an alternative proof of Corollary 3 and has the following properties (see Sections 6 and 7):

- If H is a hierarchy, then $\widehat{H}$ is a hierarchy (Theorem 10).
- If H is an interval hypergraph, then $\widehat{H}$ is an interval hypergraph (Theorem 9).

4 An algorithm to construct binary hypergraphs

We propose in this section a (non deterministic) procedure which constructs binary hypergraphs (Theorem 5). Section 5 will show that one can in fact construct any binary hypergraph and, more precisely, a binary extension of any totally balanced hypergraph.

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 $\overline{\Delta(x)} := \Delta(x) \cup \Delta^+(x) \cup \Delta^-(x)$.

The procedure constructs a binary hypergraph by generating a sequence of mixed trees. Actually, the algorithm starts with all singletons as hyperedges and, at each step, creates a new hyperedge. All hyperedges that are considered at a step are associated with nodes of a mixed tree. This mixed tree gives information on the inclusion relationship between these hyperedges (see Lemma 1) and indicates which hyperedge is to be created. The algorithm is made of three parts:

- Algorithm 1 (Basic-Tree-Construction) returns a mixed tree T_{i+1} and a map S_{i+1} constructed from a mixed tree T_i and a map S_i. The map S_i (resp. S_{i+1}) associates with each vertex of T_i (resp. T_{i+1}) a hyperedge of the final resulting hypergraph.
- Algorithm 2 (Tree-Sequence-Construction) puts Algorithm 1 into a loop to construct a sequence of 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 2 ends, the sequence of mixed trees $\mathcal{T} = ((T_0, S_0), \dots, (T_p, S_p))$ is merged into the hypergraph $H = (V_0, E)$ with $E = \bigcup_{0 \le i \le p} \{S_i(v) : v \in V_i\}$, which is binary.

Given a mixed tree $T = (V, E, \overrightarrow{E})$, a *path* of T is a sequence of vertices $x_1, \dots, x_k$ 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. By a little abuse of language, we will say that a subset S of V is *connected* if it induces a connected subgraph of T.

Algorithm 1: Basic-Tree-Construction

```
Input: A consistent mixed tree $T = (V, E, \overrightarrow{E})$ with a map $S : V \longmapsto 2^X$ where $X$ is a finite set
Output: A consistent mixed tree $T' = (V', E', \overrightarrow{E'})$ with a map $S' : V' \longmapsto 2^X$
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          Choose $\Delta'(z) \subset \Delta(z)$
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             Let $T_\Delta = (\Delta^+(z), E_\Delta)$ be a tree on vertex set $\Delta^+(z)$
13             $E' \leftarrow E' \cup E_\Delta$
14             $V' \leftarrow V' \setminus \{z\}$
15             $\overrightarrow{E'} \leftarrow \overrightarrow{E'} \setminus \{zu : u \in \Delta^+(z)\}$
16     return $T' = (V', E', \overrightarrow{E'}), S'$
```

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. In order to work, Algorithm 1 needs a consistent mixed tree as input. A mixed tree is said to be *consistent* if:

- For every vertex x, $\Delta^+(x) \neq \emptyset \implies \Delta(x) \neq \emptyset$,

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 $S_i$ is a map $V_i \longmapsto 2^{V_0}$.
1 begin
2     $\forall x \in V_0, S_0(x) \leftarrow \{x\}$
3     $\mathcal{T} \leftarrow ((T_0, S_0))$
4     $(T, S) \leftarrow (T_0, S_0)$
5     while $T$ has more than 1 vertex do
6         $(T, S) \leftarrow$ Basic-Tree-Construction$(T, S)$
7         Append $(T, S)$ to $\mathcal{T}$
8     return $\mathcal{T}$
```

- There does not exist x, y, z such that xy and yz are in $\overrightarrow{E}$.

We can now prove (Claim 2 which uses Claim 1) that given a consistent mixed tree as input, Algorithm 1 is correct.

Claim 1. *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 two 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 two 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 2 or more vertices and 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 2. *Algorithm 1 is correct: with a consistent mixed tree (having more than one vertex) as entry, it returns a consistent mixed tree.*

Proof. Claim 1 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 (Theorem 4) 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 (Theorem 5). These proofs will need some lemmas. Lemma 1 which is the keystone of these proofs and some technical lemmas: Lemmas 2, 3 and 4. Figure 2 shows a run of Algorithm 2 and Figure 3 the resulting binary hypergraph.

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 maps obtained by Algorithm 2 (*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) $uv \in E_i \implies S_i(u) \parallel S_i(v)$.

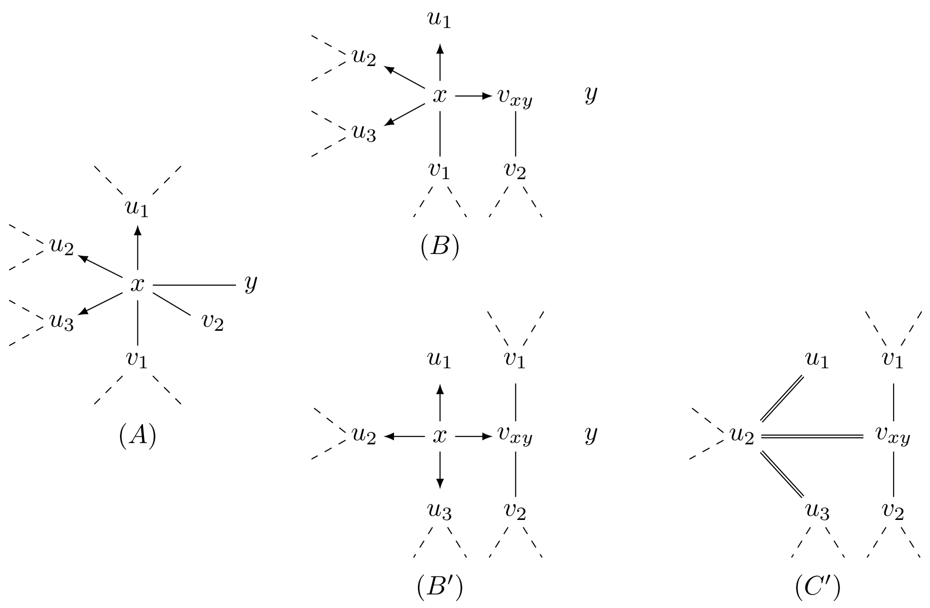

Figure 1: Two runs of Algorithm 1 on the same graph, with the same edge xy chosen. In both cases, as x is the only neighbor of y, y will be suppressed at Line 14 of the **for** loop.
In the first run ($A \to B$), at Line 9, we choose $\Delta'(x) = \{v_2\}$ and so, at Line 10, v_2 becomes a neighbor of v_{xy} (B). In the second run ($A \to B' \to C'$), at Line 9, we choose $\Delta'(x) = \Delta(x)$. 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.

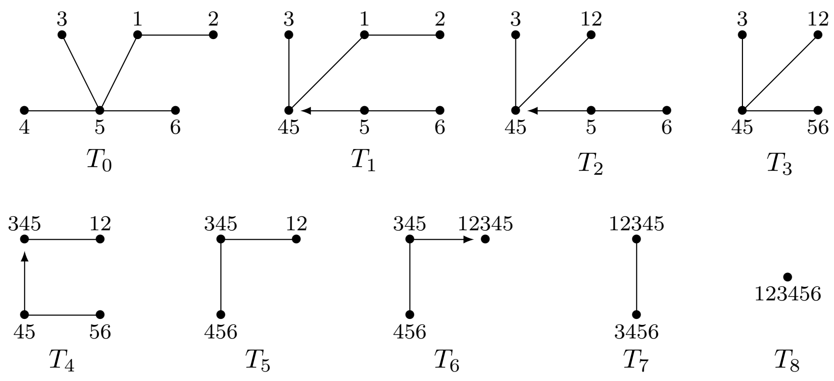

Figure 2: A sequence of mixed trees obtained by Algorithm 2

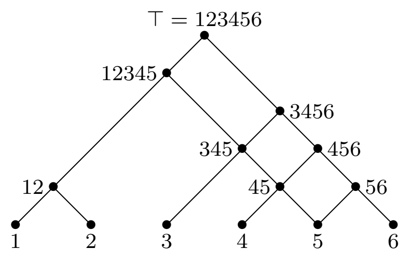

Figure 3: The binary hypergraph obtained by the sequence of mixed trees of Figure 2 represented as a lattice.

Proof. The proof will be by induction. Properties (i), (ii) and (iii) are trivially true for T_0. Suppose Properties (i), (ii) and (iii) are verified for T_i. Let $V_{i+1} \backslash V_i = \{v_{xy}\}$ be the vertex created in T_{i+1} by contracting $xy \in E_i$. We have $S_{i+1}(v_{xy}) = S_i(x) \cup S_i(y)$.

(i) Let $\alpha \in V_0$. If $\alpha \notin S_i(x)$ and $\alpha \notin S_i(y)$, then $X^{\alpha}_{i+1} = X^{\alpha}_i$ 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 $\alpha \in S_i(y)$, then $\alpha \in S_{i+1}(v_{xy})$. Since the only edge changes from T_i to T_{i+1} are edges xu or yu which become $v_{xy}u$, 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 $v = u$ 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 3. *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 maps obtained by Algorithm 2. 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 child of u_k (which is a vertex of T_{k+j}) for all $0 \leq k < p$. If $u_k \neq u_{k+1}$, by Lemma 1-ii, $S_{k+1}(u_k) \subset S_{k+1}(u_{k+1})$ and thus $S_{j+k}(u_k) \subset S_{j+k+1}(u_{k+1})$ 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 maps obtained by Algorithm 2. 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 3.

By Lemma 1-i and by construction, v_{xy} is the only vertex u of V_{i+1} such that $\alpha, \beta \in S_{i+1}(u)$. 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 maps obtained by Algorithm 2. 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 w be a descendant of v_{xy} in T_{j+1}. Since neither z nor t can be a descendant of v_{xy} (because $S_j(v_z) \subsetneq S_{j+1}(v_{zt})$ and $S_j(v_t) \subsetneq S_{j+1}(v_{zt})$), $w \neq v_{zt}$ and w is not a descendant of v_{zt}. For $\alpha \in S_j(z) \setminus S_j(t)$ and $\beta \in S_j(t) \setminus S_j(z)$, we have $\alpha, \beta \in S_{j+1}(w) \supset S_{i+1}(v_{xy}) = S_{j+1}(v_{zt})$, a contradiction with Lemma 2. □

Theorem 4. *Algorithm 2 (*Tree-Sequence-Construction*) stops. Let $\mathcal{T}$ be the final sequence. The last tree of $\mathcal{T}$ is $T_p = (\{u\}, \emptyset, \emptyset)$ with $S_p(u) = V_0$.*

Proof. From Claim 2, if T_p has more than one vertex, 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}, so Algorithm 2 stops. By Claim 1, at last step, T_p has only one vertex, *i.e.* $T_p = (\{u\}, \emptyset, \emptyset)$. □

Note that Algorithm 2 stops, even if we suppress Line 4 from Algorithm 1, *i.e.* without the maps S_i.

Lemma 4. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and maps obtained by Algorithm 2. For $0 \leq i \leq p$, let $(x_0, x_1, \ldots, x_k)$ be a path of T_i. We have:*

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

Proof. Follows immediately from Lemma 1-i. □

One can now prove the main result of this part, Theorem 5:

Theorem 5. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and maps obtained by Algorithm 2. 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 e be a hyperedge which is not a singleton, there exist $i \in \{0, \ldots, p-1\}$, $x, y \in V_i$ such that $V_{i+1} \setminus V_i = \{v_{xy}\}$ and $e = S_{i+1}(v_{xy})$. We will show that the only predecessors of e are $S_i(x)$ and $S_i(y)$. Let $e' \subsetneq e$ be a hyperedge, there exist $j \in \{0, \ldots, p\}$ and $z \in V_j$ such that $e' = S_j(z)$.

If $j \leq i$, let $t \in V_i$ be a descendant of z. By Lemma 4, we can suppose with no loss of generality that $S_i(y) \cap S_i(t) \subset S_i(x) \cap S_i(t)$. So we have $S_i(y) \cap S_j(z) \subset S_i(x) \cap S_j(z)$. As $S_j(z) \subsetneq S_{i+1}(v_{xy}) = S_i(x) \cup S_i(y)$, we have $S_j(z) \subset S_i(x)$.

Suppose now that $j > i$. If neither $S_j(z) \subset S_i(x)$ nor $S_j(z) \subset S_i(y)$, there exist $\alpha, \beta \in S_j(z)$ such that $\alpha \in S_i(x) \setminus S_i(y)$ and $\beta \in S_i(y) \setminus S_i(x)$. By Lemma 2, z is a descendant of v_{xy}, and thus $e \subset e'$, a contradiction. □

Corollary 4. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and maps obtained by Algorithm 2. Every hypergraph $H = (V_0, E)$ with $E \subset \bigcup_{0 \leq i \leq p} \{S_i(v) : v \in V_i\}$ is totally balanced.*

The following Proposition shows the complexity of Algorithm 2.

Proposition 3. *Algorithm 2 runs in $\mathcal{O}(n^3)$, where $n = |V_0|$.*

Proof. Let X be a set and $\mathcal{T}(X)$ the set of all the consistent mixed trees admitting a map $S : X \longmapsto 2^X$ 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 a set X with $|X| = n + 1$. 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 $z \in V\backslash\{x\}$, we have $S(z) \cap S(x)\backslash S(y) = \emptyset$. The set $X' = \bigcup_{z\in V\backslash\{x\}} S(z)$ is then strictly included in X, thus $|X'| \leq n$.

We denote by 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 we denote by 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 $S : X' \longmapsto 2^{X'}$. Since $|X'| < |X|$, we have that $V \leq 2 + 2 \cdot |X'| \leq 2 \cdot |X|$ which concludes the proof by induction.

Moreover, Algorithm 2 adds a new hyperedge of the hypergraph at each step so there are at most $\mathcal{O}(n^2)$ calls of Algorithm 1 (a totally balanced hypergraph has at most $\binom{n+1}{2}$ hyperedges). As Algorithm 1 is linear in the size of the input which is always $\mathcal{O}(n)$, Algorithm 2 runs in $\mathcal{O}(n^3)$. □

Note that Algorithm 2 is really efficient since it is linear in the size of the resulting hypergraph. The next section will adapt this algorithm in order to produce a binary extension from a given totally balanced hypergraph.

5 An algorithm to construct a binary extension of a totally balanced hypergraph

In this section, we will show that, given a totally balanced hypergraph H, it is possible to obtain a binary extension of H as the result of slightly modified versions of Algorithms 1 and 2, namely Algorithm 3 and Algorithm 4. The differences lie 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 4, 11 and 15). Moreover, edges are taken such that the vertices are in a homogeneous subset of T.

If $T = (V, E, \overrightarrow{E})$ is a consistent mixed tree, a subset A of V is said to be a *homogeneous* subset of T if it is connected and for all $x \in A$, $\Delta^-(x) = \emptyset$ and $\Delta(x) \subset A$. Note that if $|V| \geq 2$, then $|A| \geq 2$ (because a consistent mixed tree is connected and for any vertex x, if $\Delta^+(x) \neq \emptyset$ then $\Delta(x) \neq \emptyset$). Thus the method for choosing the edge in Algorithm 3 is a particular case of the method of Algorithm 1.

Figure 4 shows a run of Algorithm 4. The resulting binary hypergraph is the one of Figure 3. In order to show that Algorithm 4 stops, we have to prove that Algorithm 3 is correct, *i.e.* that:

1. One can always find a homogeneous subset of T (Claim 4).
2. One can always find a support tree of $H|_A$ at Line 15. It is clear by Theorem 1.
3. For all $\alpha \in A$, there exists a unique $u \in \Delta^+(z)$ such that $\alpha \in S(u)$ at Line 16. 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 $\alpha \in S(u)$.

Claim 4. *Every consistent mixed tree $T = (V, E, \overrightarrow{E})$ with $|V| \geq 2$ admits a homogeneous subset.*

Proof. The proof will be by induction on $|V|$. If $|V| = 2$, T is made of one non-oriented edge and V is homogeneous. Suppose now that, for $k > 2$, the property is true for $k' < k$, and let $T = (V, E, \overrightarrow{E})$ be a mixed tree with k vertices. If $\overrightarrow{E} = \emptyset$, V is homogeneous. So we can suppose that for some $x \in V$, $\Delta^+(x) \neq \emptyset$. As T is consistent, $\Delta^-(x) = \emptyset$ and $\Delta(x) \neq \emptyset$. Let then T' be the subgraph of T obtained by removing $\Delta^+(x)$

Algorithm 3: Basic-Tree-Construction-H

```
Input: A consistent mixed tree $T = (V, E, \overrightarrow{E})$ with a map $S : V \longmapsto 2^X$ where $X$ is a finite set and
       a closed totally balanced hypergraph $H = (X, E)$.
Output: A consistent tree $T' = (V', E', \overrightarrow{E'})$ with a map $S' : V' \longmapsto 2^X$
1  begin
2      $A \leftarrow$ a homogeneous subset of $T$
3      $\widetilde{E} \leftarrow \{xy \in E : x, y \in A\}$
4      Choose $xy \in \widetilde{E}$ such that $\sup_E(S(x), S(y))$ is minimum for inclusion order
5      $V' \leftarrow V \cup \{v_{xy}\}$
6      $S'(v_{xy}) \leftarrow S(x) \cup S(y)$ ; $S'(u) \leftarrow S(u)\ \forall u \in V$
7      $E' \leftarrow E \setminus \{xy\}$
8      $\overrightarrow{E'} \leftarrow \overrightarrow{E}$
9      for $z \in \{x, y\}$ do
10         $\overrightarrow{E'} \leftarrow \overrightarrow{E'} \cup \{zv_{xy}\}$
11         $\Delta'(z) \leftarrow \{t : zt \in E, S'(v_{xy}) \subset \sup_E(S(z), S(t))\}$
12         $E' \leftarrow E' \cup \{v_{xy}u : u \in \Delta'(z)\} \setminus \{zu : u \in \Delta'(z)\}$
13         if $\Delta'(z) = \Delta(z)$ then
14             $A \leftarrow \bigcup_{t \in \Delta^+(z)} S(t) \setminus S(z)$
15             $T_A = (A, V_A) \leftarrow$ a support tree of $H|_A$
16             $s(\alpha) \leftarrow$ the (unique) element $u$ of $\Delta^+(z)$ such that $\alpha \in S(u)$, $\forall \alpha \in A$
17             $E_\Delta = \{s(\alpha)s(\beta) : \alpha\beta \in V_A\}$
18             $E' \leftarrow E' \cup E_\Delta$
19             $V' \leftarrow V' \setminus \{z\}$
20             $\overrightarrow{E'} \leftarrow \overrightarrow{E'} \setminus \{zu : u \in \Delta^+(z)\}$
21     return $T' = (V', E', \overrightarrow{E'}), S'$
```

Algorithm 4: Tree-Sequence-Construction-H

```
Input: A totally balanced hypergraph $H = (V_0, E)$ 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 $S_i$ is a map $V_i \longmapsto 2^{V_0}$.
1  begin
2      $\overrightarrow{E_0} \leftarrow \emptyset$
3      $\forall x \in V_0, S_0(x) \leftarrow \{x\}$
4      $\mathcal{T} \leftarrow ((T_0, S_0))$
5      $(T, S) \leftarrow (T_0, S_0)$
6      while $T$ has more than 1 vertex do
7          $(T, S) \leftarrow$ Basic-Tree-Construction-H$(T, S, \overline{H})$
8          Append $(T, S)$ to $\mathcal{T}$
9      return $\mathcal{T}$
```

and T'' the connected component of T' containing x. T'' is a consistent mixed tree with strictly less than k vertices. In addition, T'' has at least 2 vertices (it contains x and $\Delta(x)$ which is not empty). So T'' admits a homogeneous subset A, which is also a homogeneous subset for T. □

We now prove that Algorithm 4 constructs a binary extension $\widehat{H}$ of the totally balanced hypergraph H.

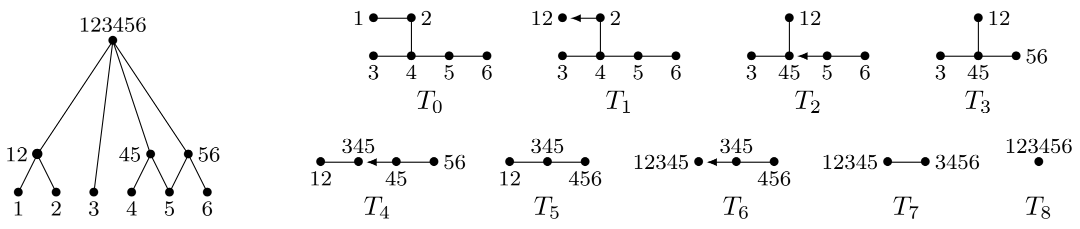

Figure 4: A totally balanced hypergraph, subhypergraph of the binary one of Figure 3, and a sequence of mixed trees which constructs this hypergraph.

This result is Theorem 6, which uses Lemma 5; it is the converse of Corollary 1.

Lemma 5. *Let $H = (V, E)$ be a closed totally balanced hypergraph and $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$, with $\forall i \in \{0, \ldots, p\}$, $T_i = (V_i, E_i, \overrightarrow{E_i})$ and $S_i : V_i \longmapsto 2^V$, be a sequence of mixed trees and maps obtained by Algorithm 4. Then $\forall e \in E, 0 \leq i \leq, p$, $\Psi_i^e := \{v \in V_i : S_i(v) \subset e\}$ induces a subtree of T_i and $\bigcup_{v \in \Psi_i^e} S_i(v)$ is either empty or equal to e.*

Proof. We prove the property by induction on i. Since (V_0, E_0) is a support tree of H, by Theorem 1 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 $e \in E$, Ψ_i^e induces a subtree of T_i and $\bigcup_{v \in \Psi_i^e} S_i(v) = e$.

Several cases can occur:

$S_i(x) \not\subset e$ and $S_i(y) \not\subset e$
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.

$S_i(x) \subset e$ and $S_i(y) \subset e$
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 $S_i(x) \subset S_i(u)$, Line 15 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 $\bigcup_{v \in \Psi_{i+1}^e} S_i(v) = \bigcup_{v \in \Psi_i^e} S_i(v) \cup S_{i+1}(v_{xy}) = e \cup S_{i+1}(v_{xy}) = e$.

$S_i(x) \subset e$ and $S_i(y) \not\subset e$ (symmetrically, $S_i(x) \not\subset e$ and $S_i(y) \subset e$).
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 $S_i(t) \subset e$, $sup_E(S_i(x), S_i(t)) \subset e$ 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.* $\forall t \in \Delta(x), S_i(t) \not\subset e$) and $\Delta_i^+(x) \neq \emptyset$, Lines 15–18 ensure that Ψ_{i+1}^e is connected; in addition, $\Psi_{i+1}^e = \Psi_i^e \setminus \{x\}$. As $S_i(x) \subset S_i(t)$ for $t \in \Delta^+(x)$, $\bigcup_{t \in \Psi_{i+1}^e} S_{i+1}(t) = \bigcup_{t \in \Psi_i^e} S_i(t) = e$. If $x \notin V_{i+1}$ and $\Delta_i^+(x) = \emptyset$, Ψ_{i+1}^e is empty. □

The notation Ψ_i^e of Lemma 5 will be often used in the rest of the paper.

Theorem 6. *$\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and maps obtained by Algorithm 4 for a totally balanced hypergraph H. The hypergraph $\widehat{H} = (V_0, \widehat{E})$ with $\widehat{E} = \bigcup_{0 \leq i \leq p} \{S_i(v) : v \in V_i\}$ is a binary extension of H.*

Proof. As Algorithm 4 is just an adaptation of Algorithm 2, by Theorem 5, the hypergraph $\widehat{H}$ is binary. Let $e \in E$, if $e = V$, then $V_p = \{e\}$ and $e \in \widehat{E}$; 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 $V_i \setminus V_{i-1} = v_{xy}$, $S_i(x) \subset e$, $S_i(y) \not\subset e$, $x \notin V_i$ and $\Delta^+(x) = \emptyset$. In this case, $\Psi_{i-1}^e = x$ and so $e = S_i(x) \in \widehat{E}$. □

6 Minimal binary extensions

For every hypergraph $H = (V, E)$ with closure $\overline{H} = (V, \overline{E})$, we define $\mathcal{P}(H)$ as the quantity:

$$\mathcal{P}(H) := \sum_{e \in \overline{E}, |e|>1} (|\{v \in \overline{E} : v \prec e\}| - 2)$$

This Section will show that $\mathcal{P}(H)$ is small for totally balanced hypergraphs and minimal for the binary ones. Claim 5 gives a lower and an upper bound of this number and Lemma 6 gives an upper bound for totally balanced hypergraphs. These results are resumed in Theorem 7. The main purpose of $\mathcal{P}(H)$ is nevertheless to show that Algorithm 4 constructs a binary extension with the minimum number of added clusters.

Claim 5. *Let $H = (V, E)$ be a hypergraph with $|V| = n$. We have $0 \leq \mathcal{P}(H) \leq (n-4) \cdot 2^{n-1} + n + 2$. The lower bound is obtained for binary hypergraphs and the upper bound for hypercubes $(V, 2^V \backslash \{\emptyset\})$.*

Proof. Since each non singleton hyperedge has at least two predecessors, it is clear that $\mathcal{P}(H)$ is positive and is equal to 0 for binary hypergraphs.

Suppose now that H is not a hypercube. Let $a \in 2^V$ be a smallest non empty element not in $\overline{E}$ and let $H' := (V, \overline{E} \cup \{a\})$. There exists only one element $a' \in \overline{E}$ such that $a \prec a'$ (a' is the intersection of all hyperedges x of H such that $a \subset x$). Let b be a predecessor of a (in H'). In H, b is a predecessor of a'; and if b is a predecessor of another hyperedge c, then b is also a predecessor of c in H'.

The hypergraph $H' = (V, \overline{E} \cup \{a\})$ is closed (for every hyperedge x, if $x \cap a \neq a$, then $x \cap a$ is smaller than a), so $\mathcal{P}(H') = \mathcal{P}(H) + 1$ and we can add iteratively all the missing hyperedges. Thus hypercubes realize the upper bound.

To conclude, note that for a hypercube H with n vertices, $\mathcal{P}(H) = \sum_{2 \leq k \leq n} (k-2) \cdot \binom{n}{k}$. Since $\sum_{0 \leq k \leq n} k \cdot \binom{n}{k} = n \cdot 2^{n-1}$, we have $\mathcal{P}(H) = (n-4) \cdot 2^{n-1} + n + 2$. □

Lemma 6. *If $H = (V, E)$ is a totally balanced hypergraph, then $\mathcal{P}(H) \leq |V| - 2$.*

Proof. With no loss of generality, we can suppose that H is a closed totally balanced hypergraph. Closed totally balanced hypergraphs are in bijection with so-called dismantlable lattices [4]. A lattice $L = (E, \leq)$ is *dismantlable* [12] if there exists a sequence $(E_i, 0 \leq i \leq n)$ with $E_0 = \emptyset$, $E_n = E$ and $E_{i-1} = E_i \setminus e_i$ where e_i is a doubly irreducible element of the lattice $L_i = (E_i, \leq)$. An element is *doubly irreducible* in a finite lattice if it has exactly one predecessor and one successor. Moreover, if e is a doubly irreducible element of a lattice $L = (E, \leq)$, then $L' = (E \setminus \{e\}, \leq)$ remains a lattice.

So, if $H = (V, E)$, with $|E| = m$, is a closed totally balanced hypergraph, $(E \cup \{\emptyset\}, \subset)$ is a dismantlable lattice, then there exists an order $e_1, \ldots, e_{m-1}$ on the elements of $E \backslash \{V\}$ such that, for any $1 \leq i < m$, the set $S_i = \{V, \emptyset\} \cup \{e_1, \ldots, e_i\}$ is closed and e_i admits only one predecessor p_i and only one successor s_i in S_i. Let $P_i := \sum_{e \in S_i} |\{v \in S_i : v \prec e\}|$, it is then clear that:

- $|S_{i+1}| = |S_i| + 1$
- $P_{i+1} = P_i + 1$ if $p_{i+1} \prec_{S_i} s_{i+1}$
- $P_{i+1} = P_i + 2$ if $p_{i+1} \not\prec_{S_i} s_{i+1}$ (we nevertheless have $p_{i+1} \subsetneq s_{i+1}$)

We can now prove by induction that $2 \cdot |S_i| \geq P_i + 4$. Since $2 \cdot |S_1| = 6$ and $P_1 = 2$ the property is true for $i = 1$. Suppose it true for $i \geq 1$. We have $2 \cdot |S_{i+1}| = 2 \cdot |S_i| + 2 \geq P_i + 4 + 2 \geq P_{i+1} + 4$, which concludes the proof by induction. So, we have $2 \cdot |S_{m-1}| \geq P_{m-1} + 4$.

Since $|E| = |S_{m-1}| - 1$ (we remove the empty set) and $P_{m-1} = \mathcal{P}(H) + 2 \cdot (|E| - |V|) + |V|$ (the predecessors of the singletons are the emptyset), we have $2 \cdot (|E| + 1) \geq \mathcal{P}(H) + 2 \cdot (|E| - |V|) + |V| + 4$, hence the result. □

Note that this upper bound for totally balanced hypergraphs is reached for $H = (V, E)$ with $E = \{V\} \cup \{\{x\} : x \in V\}$. All these properties on $\mathcal{P}(H)$ are summarized in the following:

Theorem 7. *Let $H = (V, E)$ be a hypergraph with $|V| = n$, we have $0 \leq \mathcal{P}(H) \leq (n-4) \cdot 2^{n-1} + n + 2$ and:*

- *$\mathcal{P}(H) = 0$ if and only if H is a binary hypergraph,*
- *$\mathcal{P}(H) = (n-4) \cdot 2^{n-1} + n + 2$ if and only if $E = 2^V \setminus \{\emptyset\}$.*
- *$\mathcal{P}(H) \leq n - 2$ if H is a totally balanced hypergraph,*
- *$\mathcal{P}(H) = n - 2$ if $E = \{V\} \cup \{\{x\} : x \in V\}$.*

Theorem 7 shows that the $\mathcal{P}(H)$ is very low for totally balanced hypergraphs, it is bound by the number of vertices even though the number of hyperedges can be $\binom{|V|+1}{2}$. We will now show that Algorithm 4 only adds a minimal number of clusters. To do this, we will prove that Algorithm 4 iteratively constructs closed totally balanced hypergraphs $\widehat{H_i}$ (Lemma 7) and that $\mathcal{P}(\widehat{H_i})$ decreases one by one (Lemma 10).

Lemma 7. *Let $H = (V, E)$ be a closed totally balanced hypergraph and $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ with $T_i = (V_i, E_i, \overrightarrow{E_i})$ be a sequence of mixed trees and maps built by Algorithm 4. For $0 \leq i \leq p$, let $\widehat{H_i} = (V, \widehat{E_i})$ be the hypergraph defined by $\widehat{E_i} = E \cup (\bigcup_{0 \leq j \leq i} \{S_j(v) : v \in V_j\})$. Then for every $0 \leq i \leq p$, $\widehat{H_i}$ is a closed totally balanced hypergraph.*

Proof. Let $\widehat{H}$ be the binary extension of H built from $\mathcal{T}$. Since $\widehat{H_i}$ is a subhypergraph of $\widehat{H}$, by Theorem 1, $\widehat{H_i}$ is totally balanced.

We now show by induction on i that $\widehat{H_i}$ is closed. Since H is closed and $\widehat{H_0} = H$ the property is true for $i = 0$. Suppose it true for i and set $v_{xy} := V_{i+1} \setminus V_i$, *i.e.* $\widehat{E_{i+1}} \setminus \widehat{E_i} = \{e\}$, with $e = \{S_i(x) \cup S_i(y)\}$. Let e' be a hyperedge of $\widehat{H_{i+1}}$.

If there exist $\alpha \in S_i(x) \setminus S_i(y)$ and $\beta \in S_i(y) \setminus S_i(x)$ with $\alpha, \beta \in e'$ then, by Lemma 2, $e \subset e'$ (it is possible that $e' \neq e$ if $e' \in E$). So, we can suppose with no loss of generality that $e' \cap S_i(y) \setminus S_i(x) = \emptyset$. In this case, $e' \cap e = e' \cap S_i(x) \in \widehat{E_i} \subset \widehat{E_{i+1}}$. □

The sequence $(\widehat{H_0}, \ldots, \widehat{H_p})$ of hypergraphs defined in Lemma 7 will be used several times in the following. In order to prove Lemma 10, we need to measure the impact of the creation of each cluster in Algorithm 4. This is done by Lemma 8 (which needs Claim 6), and Lemma 9.

Claim 6. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ with $T_i = (V_i, E_i, \overrightarrow{E_i})$ be a sequence of consistent mixed trees and maps built by Algorithm 4 from a closed totally balanced hypergraph $H = (V, E)$. If $xy \in \overrightarrow{E_i}$ and $xz \in E_i$ for some $0 \leq i < p$, then there exists a hyperedge $e \in E$ such that $y \notin \Psi_i^e$ and $z, x \in \Psi_i^e$.*

Proof. If $V_i \setminus V_{i-1} = \{y\}$, *i.e.* if the arc xy is created at Step i, then, by Line 11 and 12 of Algorithm 3, $S_i(y) \not\subset \sup_E(S_i(x), S_i(z))$; so $e = \sup_E(S_i(x), S_i(z))$ is a hyperedge of H such that $y \notin \Psi_i^e$ and $z, x \in \Psi_i^e$.

Suppose now that $y \in V_{i-1}$, *i.e.* that the arc xy is created at Step $j < i$. If $z \in V_{i-1}$, $j < i - 1$ and Step i is identical to Step $i - 1$. If $z = V_i \setminus V_{i-1}$, $z = v_{z_1 z_2}$ and, in T_{i-1}, x is a neighbor of z_1. Neither $z_1 \in \Delta^-(x)$ (since, in T_{i-1}, $\Delta^-(x) \neq \emptyset$) nor $z_1 \in \Delta^+(x)$ (otherwise, the edge $z_1 z_2$ would not have been chosen at Step $i - 1$); so, in T_{i-1} $x \in \Delta(z_1)$ and, in T_i, $x \in \Delta(z)$. By Line 11 and 12 of Algorithm 3, $S_i(z) \subset \sup_E(S_{i-1}(z_1), S_{i-1}(x))$, and thus $\sup_E(S_i(z), S_i(x)) \subset \sup_E(S_{i-1}(z_1), S_{i-1}(x))$.

So, there exists $z^* \in V_j$ such that $z^* \in \Delta(x)$ for T_i and $\sup_E(S_i(z), S_i(x)) \subset \sup_E(S_j(z^*), S_j(x))$. As $S_j(y) \not\subset \sup_E(S_j(z^*), S_j(x))$ (at Step j, we are in the first case of the proof), $e = \sup_E(S_i(x), S_i(z))$ is a hyperedge of H such that $y \notin \Psi_i^e$ and $z, x \in \Psi_i^e$. □

Lemma 8. *Let $\mathcal{T} = ((T_0, S_0), \ldots, (T_p, S_p))$ be a sequence of mixed trees and maps obtained by Algorithm 4 for a closed totally balanced hypergraph $H = (V, E)$. For $0 \leq i \leq p$, let A be the homogenous subset of T and xy be the edge chosen at Step i. If we denote $\sup_E(S(x), S(y))$ by e, then $2 \leq |\Psi_i^e|$ and $\Psi_i^e \subset A$.*

Proof. Since $x, y \in \Psi_i^e$, it is clear that $2 \leq |\Psi_i^e|$. Suppose that $\Psi_i^e \not\subset A$. Then there exist $u \in \Psi_i^e \cap A$, $v \in \Delta^+(u) \cap \Psi_i^e$ ($v \notin A$) and $w \in \Psi_i^e \cap A \cap \Delta(u)$. By Claim 6 there exists a hyperedge e' such that $u, w \in \Psi_i^{e'}$ and $v \notin \Psi_i^{e'}$. The hyperedge $e'' = e \cap e'$ is such that $u, w \in \Psi_i^{e''}$; thus $|\Psi_i^{e''} \cap A| \geq 2$, and $\sup_E(S(u), S(w)) \subset e'' \subsetneq e$, which is in contradiction with the minimality of e. □

Lemma 9. *Let $H = (V, E)$ be a closed totally balanced hypergraph , $\mathcal{T} = ((T_0, S_0), \dots, (T_p, S_p))$ with $T_i = (V_i, E_i, \overrightarrow{E_i})$ be a sequence of mixed trees and maps built by Algorithm 4 and $xy \in E_i$ be the edge chosen at Step i at Line 4 of Algorithm 3. If we denote $\sup_E(S_i(x), S_i(y))$ by e, then $\{S_i(u) : u \in \Psi_i^e\}$ is the set of all predecessors of e in $\widehat{E_i}$ (with $\widehat{H_i} = (V, \widehat{E_i})$ as defined in Lemma 7).*

Proof. Let e' be a predecessor of e in $\widehat{H_i}$. We first show that $e' = S_i(u)$ for a vertex $u \in V_i$ (and thus for $u \in \Psi_i^e$). Suppose that $e' = S_j(u)$ for a vertex $u \in V_j \setminus V_i$, with $j < i$. We can suppose, with no loss of generality, that $u \notin V_{j+1}$; so $V_{j+1} \setminus V_j = \{v_{uu'}\}$. The set Ψ_j^e contains u and one of its neighbors ($S_j(u) \subsetneq e$ and it is connected by Lemma 5). If $u' \in \Psi_j^e$, then $S_{j+1}(v_{uu'}) \subsetneq e$; and if there exists $u'' \in \Delta^+(u) \cap \Psi_j^e$, then $S_j(u'') \subsetneq e$. In both cases, $S_j(u)$ is not a predecessor of e. Otherwise, there exists $u'' \in \Delta(u) \cap \Psi_j^e$. Since $u \notin V_{j+1}$, by Line 11 of Algorithm 3, $S_{j+1}(v_{uu'}) \subset \sup_E(S_j(u), S_j(u''))$. As $S_j(u) \subsetneq S_{j+1}(v_{uu'})$ and $\sup_E(S_j(u), S_j(u'') \subset e$, $S_j(u)$ is not a predecessor of e. A contradiction: there exists $u \in V_i$ such that $e' = S_i(u)$.
Suppose now that $e' \in E \setminus (\bigcup_{0 \le j \le i} \{S_j(v) : v \in V_j\})$. We have $\Psi_i^{e'} \subsetneq \Psi_i^e \subset A$; so at Line 4 of Algorithm 3, we would have chosen xy such that $\sup_E(S(x), S(y)) = e'$ instead of e. Thus for any predecessor e' of e in $\widehat{H_i}$, there exists $u \in V_i$ such that$e' = S_i(u)$: Ψ_i^e is exactly the set of all the predecessors of e in $\widehat{H_i}$.

We now show by contradiction that for all $u, v \in \Psi_i^e$, $S_i(u) \not\subset S_i(v)$, and so, for all $u \in \Psi_i^e$, $S_i(u)$ is a predecessor of e. Let $u = u_0u_1 \dots u_k = v$ be the path between u and v in T_i. By Lemma 4, $S_i(u) \cap S_i(v) \subset S_i(u_1) \cap S_i(v) \subset S_i(u_1)$. If $S_i(u) \subset S_i(v)$, $S_i(u) \cap S_i(v) = S_i(u)$; so, as $S_i(u) \neq S_i(u_1)$, $S_i(u) \subsetneq S_i(u_1)$ and thus, by Lemma 1, $uu_1 \in \overrightarrow{E_i}$. The path u is included in Ψ_i^e which is, by Lemma 8, included in a homogeneous set, a contradiction. □

Lemma 10. *With the notations of Lemma 7, for $i \in \{0, \dots, p-1\}$, if $\widehat{H_{i+1}} \neq \widehat{H_i}$, then $\mathcal{P}(\widehat{H_{i+1}}) = \mathcal{P}(\widehat{H_i}) - 1$.*

Proof. If $\widehat{H_{i+1}}$ is different from $\widehat{H_i}$ then $\widehat{E_{i+1}} \setminus \widehat{E_i} = \{S_i(x) \cup S_i(y)\}$, where $xy \in E_i$ is the edge taken at Step i by Algorithm 3. This new hyperedge has two predecessors ($S_i(x)$ and $S_i(y)$). The hyperedge $\sup_E(S_i(x), S_i(y))$ has one predecessor less in $\widehat{H_{i+1}}$ than in $\widehat{H_i}$. The number of predecessors of the other hyperedges is unchanged. □

The following theorem shows that the binary extension built by algorithm 4 always adds the minimum number of clusters.

Theorem 8. *Let $H = (V, E)$ be a totally balanced hypergraph and $\widehat{H} = (V, \widehat{E})$ be one of its binary extension built by Algorithm 4. If $H' = (V', E')$ is another binary extension of H, then $|E'| \geq |\widehat{E}|$. In particular, if H is binary, then $\widehat{H} = H$.*

Proof. First note that any binary extension of H contains $\overline{E}$; thus one can consider with no loss of generality that $H = (V, E)$ is a closed totally balanced hypergraph. Let e be a hyperedge of E with $k \geq 2$ predecessors $p_1, \dots, p_k$ in E and let $G = (V_G, E_G)$ be the directed graph defined by V_G is the set of hyperedges of E' such that $u \in V_G$ if $p_i \subset u \subset e$ for some $1 \leq i \leq k$, and $xy \in E_G$ if $y \prec_{V_G} x$.

This graph admits a directed minimal spanning tree rooted in e with $p_1, \dots, p_k$ as leaves. Moreover, each inner vertex x of this directed tree is such that $\Delta^+(x) \leq 2$. Since a binary tree with k leaves has at least $k - 1$ inner vertices, V_G admits more than $k - 2$ vertices different from e or p_i ($1 \leq i \leq k$). Since these vertices are in $E' \backslash E$, we conclude that any binary extension H' of H has more edges than $\mathcal{P}(H)$ new hyperedges. By Lemma 10, the binary extension $\widehat{H}$ built by Algorithm 4 has exactly $\mathcal{P}(H)$ new hyperedges, so $|E'| \geq |\widehat{E}|$. □

Finally, in order to find a binary extension of some totally balanced hypergraph $H = (V, E)$, the closure is the most costly operation since it can add $\mathcal{O}(|V|^2)$ clusters. The binarization itself only adds at most $|V| - 2 = \mathcal{O}(|V|)$ clusters for a closed totally balanced hypergraph. But since in classification we usually work with closed models (hierarchy or interval clusters are usually closed for instance) the binarization is a very light operation.

7 Stability properties of Algorithm 4

This section shows that Algorithm 4 is stable for two popular clustering models, interval hypergraphs and hierarchies, both subsets of totally balanced hypergraphs.

A hypergraph $H = (V, E)$ is *hierarchical* or a *hierarchy* if $\forall e, e' \in E, e \cap e' \in \{e, e', \emptyset\}$ and H is an *interval hypergraph* if there exist a linear order σ on V such that, when V is sorted along σ, every hyperedge is an interval of V. The permutation σ is said to be *compatible*. Clearly a hierarchy is an interval hypergraph. Moreover, an interval hypergraph is totally balanced: if, when sorted along a compatible order σ, $V = (v_1, v_2, \ldots, v_n)$, then the (non oriented) path $v_1 - v_2 - \ldots - v_n$ is a support tree that we denote by P^σ.

Theorem 9. *Let H be an interval hypergraph admitting a permutation σ as compatible order and let $\widehat{H}$ be a binary extension of H built by Algorithm 4, starting with P^σ as T_0. Then $\widehat{H}$ is an interval hypergraph and admits σ as a compatible order.*

Proof. Since every connected subset of V_0 is an interval for σ, the proof derives immediately from Lemma 5 because for all $e \in E_{\widehat{H}}$, Ψ_0^e is a connected subtree of T_0 thus an interval. $\square$

Note that if we start Algorithm 4 with a support tree which is not a path, then the binary extension may not be an interval hypergraph (see Figure 5).

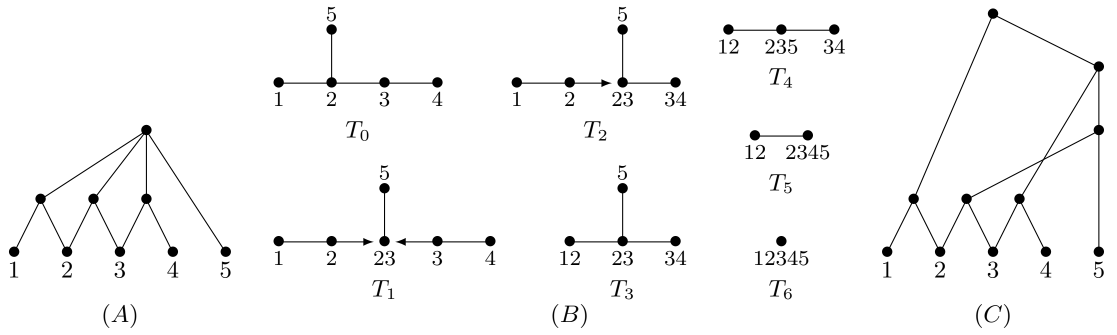

Figure 5: (A): An interval hypergraph ($\{1, 2\}$, $\{2, 3\}$ and $\{3, 4\}$, are intervals of $(1, 2, 3, 4, 5)$). (B): a run of Algorithm 4 which does not start with a path. (C): the resulting binary extension, which is not an interval hypergraph (there is no linear order of $\{1, 2, 3, 4, 5\}$ admitting $\{1, 2\}$, $\{2, 3\}$, $\{3, 4\}$ and $\{2, 3, 5\}$ as intervals).

Lemma 11. *Let $\mathcal{T} = ((T_0, S_0), \ldots (T_p, S_p))$ with $T_i = (V_i, E_i, \overrightarrow{E_i})$ be the sequence of consistent mixed trees and maps built by Algorithm 2 (or Algorithm 4) and $H = (V, E)$ be the resulting hypergraph. H is a hierarchy if and only if $\overrightarrow{E_i} = \emptyset$ for all $0 \leq i \leq k$.*

Proof. We first prove that if there exists $0 \leq i < p$ such that $\overrightarrow{E_i} \neq \emptyset$, then H is not a hierarchy. Let xy be an arc created at Step i. In T_i, $\Delta(x) \neq \emptyset$. Let xz be the first edge with x as an extremity to be selected at Line 2 of Algorithm 1 or Line 4 Algorithm 3. By Lemma 1, $S_i(x) \subset S_i(y)$ and there exist $\alpha \in S_i(y) \setminus S_i(x)$ and $\beta \in S_i(z) \setminus S_i(x)$; in addition, $\alpha \notin S_i(z)$ and $\beta \notin S_i(y)$. Both $S_i(x) \cup S_i(z)$ and $S_i(y)$ are in E but $e = (S_i(x) \cup S_i(z)) \cap S_i y)$ is neither $\emptyset$ ($S_i(x) \subset e$) nor $S_i(x) \cup S_i(z)$ ($\beta \notin e$) nor $S_i(y)$ ($\alpha \notin e$).

Conversely, suppose $\overrightarrow{E_i} = \emptyset$ for all $0 \leq i \leq p$. A trivial induction on i shows that $P_i = \{S_i(x) : x \in V_i\}$ is a partition of V_0, and that $P_{i+1} = P_i \cup \{S_i(x) \cup S_i(y)\} \backslash \{S_i(x), S_i(y)\}$ for an edge xy of V_i. It is then clear that $\bigcup_{0 \leq i \leq p} P_i = E$ is a hierarchy. $\square$

Theorem 10. *Let $H = (V, E)$ be a hierarchical hypergraph and let $\widehat{H}$ be a binary extension of H built by Algorithm 4. Then $\widehat{H}$ is hierarchical.*

Proof. Let $\mathcal{T} = ((T_0, S_0), \ldots (T_p, S_p))$ with $T_i = (V_i, E_i, \overrightarrow{E_i})$ be the sequence of consistent mixed trees and maps built by Algorithm 4 and $\widehat{H} = (V, \widehat{E})$ be the resulting hypergraph. We will show by induction on i that $\overrightarrow{E_i} = \emptyset$ fo all $i \in \{0, \ldots, p\}$. Clearly, $\overrightarrow{E_0} = \emptyset$.

Suppose that $\overrightarrow{E_i} = \emptyset$ for all $0 \leq i \leq i_0$. For $i = i_0 + 1$, let $xy \in E_{i_0}$ be the chosen edge and let $xz \in E_{i_0}$ be another undirected edge. Since H is a hierarchy, $\sup_E(S_{i_0}(x), S_{i_0}(y)) \cap \sup_E(S_{i_0}(x), S_{i_0}(z)) \in \{\sup_E(S_{i_0}(x), S_{i_0}(y)), \sup_E(S_{i_0}(x), S_{i_0}(z)), \emptyset\}$, thus by minimality of $\sup_E(S_{i_0}(x), S_{i_0}(y))$ and the fact that the intersection is not empty we have that $\sup_E(S_{i_0}(x), S_{i_0}(y)) \subset \sup_E(S_{i_0}(x), S_{i_0}(z))$ and xz become the undirected edge $v_{xy}z$ in T_{i_0+1}: all the undirected edges of x are now undirected edges of v_{xy}, x is deleted and no directed edges are created. Thus $\overrightarrow{E_{i_0+1}} = \emptyset$ and Lemma 11 allows us to conclude. $\square$

8 Conclusion

We show in this paper that the only clustering model that admits binary extensions is the totally balanced hypergraph one, making them useful both as a clustering model and a split model. Totally balanced hypergraphs were already known to have numerous good properties (see for instance Lehel, McMorris and Powers [10]). We have shown that they can in addition replace the (too) simple hierarchical model when the overlapping between clusters is desirable.

We also exhibit an algorithm that can binarize any given totally balanced hypergraph by adding a minimal number of clusters and is stable for the hierarchies and the interval hypergraphs. We now work on using these algorithms for practical cases. For instance, we plan to use Algorithm 4 to 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.

Acknowledgments

We are indebted to an anonymous referee for Proposition 2 (and the proof of Corollary 3) and for many other valuable remarks and advice.

References

[1] R.P. Anstee & M. Farber (1983), Hypergraphs with no special cycles, *Combinatorica* **3**, 141–146.

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

[3] P. Bertrand (2000), Set Systems and Dissimilarities, *European Journal of Combinatorics* **21**, 727–743.

[4] F. Brucker & A. Gély (2011), Crown-free Lattices and Their Related Graphs, *Order* **28**, 443–454.

[5] F. Brucker & P. Préa (2015), Totally Balanced Formal Concept Representations, in *Proceedings of ICFCA 2015*, J. Baixeries C. Sacarea& M. Ojeda-Aciego Eds, Springer, 169–182.

[6] J. Diatta & B. Fichet (1994), From Asprejan hierarchies and Bandelt-Dress weak hierarchies to quasi-hierarchies, in *New Approaches in classification and data analysis*, E. Diday, Y. Lechevallier, M. Schader & P. Bertrand Eds, Springer, 111–118.

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

[8] J. Lehel (1983), Helly hypergraphs and abstract interval structures, *Ars Combinatoria* **16-A**, 239–253.

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

[10] J. Lehel, F.R. McMorris & R.C. Powers (1998), Consensus methods for pyramids and other hypergraphs, in *Data Science, Classification and Related Methods*, C. Hayashi, N. Ohsumi, K. Yajima, Y. Tanaka, H.H. Bock& Y. Baba Eds, Springer, 187–190.

[11] L. Lovász (1968), Graphs and set systems, in *Beiträge zur Graphentheorie*, H. Walther, H. Sachs & H-J. Voss Eds, Teubner, 99–106.

[12] I. Rival (1974), Lattices with doubly irreducible elements, *Canadian Mathematical Bulletin* **17-1**, 91–95,

[13] W.S. Robinson (1951), A Method for Chronologically Ordering Archeological Deposits, *American Antiquity* **16**, 293–301.

[14] J. Spinrad (2003), *Efficient Graph Representations*, American Mathematical Society.