Theoretical Computer Science 615 (2016) 1–11

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Fast factorization of Cartesian products of (directed) hypergraphs

Marc Hellmuth [a,b,*], Florian Lehner [c,*]

[a] Department of Mathematics and Computer Science, University of Greifswald, Walther-Rathenau-Straße 47, 17487 Greifswald, Germany
[b] Center for Bioinformatics, Saarland University, Building E 2.1, Room 413, P.O. Box 15 11 50, D-66041 Saarbrücken, Germany
[c] Department of Mathematics, University of Hamburg, Bundesstraße 55, 20146 Hamburg, Germany

ARTICLE INFO

Article history:
Received 28 August 2015
Accepted 17 November 2015
Available online 2 December 2015
Communicated by D.-Z. Du

Keywords:
Directed hypergraph
Cartesian product
Prime factor decomposition
Factorization algorithm
2-section

ABSTRACT

Cartesian products of graphs and hypergraphs have been studied since the 1960s. For (un)directed hypergraphs, unique *prime factor decomposition (PFD)* results with respect to the Cartesian product are known. However, there is still a lack of algorithms, that compute the PFD of directed hypergraphs with respect to the Cartesian product.

In this contribution, we focus on the algorithmic aspects for determining the Cartesian prime factors of a finite, connected, directed hypergraph and present a first polynomial time algorithm to compute its PFD. In particular, the algorithm has time complexity $O(|E||V|r^2)$ for hypergraphs $H = (V, E)$, where the rank r is the maximum number of vertices contained in a hyperedge of H. If r is bounded, then this algorithm performs even in $O(|E|\log^2(|V|))$ time. Thus, our method additionally improves also the time complexity of PFD-algorithms designed for undirected hypergraphs that have time complexity $O(|E||V|r^6\Delta^6)$, where Δ is the maximum number of hyperedges a vertex is contained in.

1. Introduction

Products are a common way in mathematics of constructing larger objects from smaller building blocks. For graphs, hypergraphs, and related set systems several types of products have been investigated, see [18,14] for recent overviews.

In this contribution we will focus on the *Cartesian product* of *directed hypergraphs* that are the common generalization of both directed graphs and (undirected) hypergraphs. In particular, we present a fast and conceptually very simple algorithm to find the decomposition of directed hypergraphs into *prime* hypergraphs (its so-called prime factors), where a (hyper)graph is called prime if it cannot be presented as the product of two nontrivial (hyper)graphs, that is, as the product of two (hyper)graphs with at least two vertices.

Graphs and the Cartesian product A graph is a tuple $G = (V, E)$ with non-empty set of vertices V and a set of edges E containing two-element subsets of V. If the edges are ordered pairs, then G is called directed and undirected, otherwise. The Cartesian *graph* product was introduced by Gert Sabidussi [26]. As noted by Szamkołowicz [29] also Shapiro introduced a notion of Cartesian products of graphs in [27]. Sabidussi and independently V.G. Vizing [30] showed that connected *undirected* graphs have a representation as the Cartesian product of prime graphs that is unique up to the order and isomorphisms of the factors. The question whether one can find the prime factorization of connected undirected graphs in polynomial time

* Corresponding authors.
E-mail addresses: mhellmuth@mailbox.org (M. Hellmuth), mail@florian-lehner.net (F. Lehner).

http://dx.doi.org/10.1016/j.tcs.2015.11.038
0304-3975/

was answered about two decades later by Feigenbaum et al. [13] who presented an $O(|V|^{4.5})$ time algorithm. From then on, a couple of factorization algorithms for undirected graphs have been developed [1,11,13,21,31]. The fastest one is due to Imrich and Peterin [21] and runs in linear-time $O(|V|+|E|)$.

For connected *directed* graphs, Feigenbaum showed that the Cartesian product satisfies the unique prime factorization property [12]. Additionally, she provided a polynomial-time algorithm to determine the prime factors which was improved to a linear time approach by Crespelle et al. [9].

Hypergraphs and the Cartesian product Hypergraphs are a natural generalization of graphs in which "edges" may consist of more than two vertices. More precisely, a hypergraph is a tuple $H = (V, E)$ with non-empty set of vertices V and a set of hyperedges E, where each $e \in E$ is an ordered pair of non-empty sets of vertices $e = (t(e), h(e))$. If $t(e) = h(e)$ for all $e \in E$ the hypergraph is called undirected and directed, otherwise. Products of hypergraphs have been investigated by several authors since the 1960s [2,3,5–8,10,16,19,20,22,23,25,28,32]. It was shown by Imrich [19] that connected *undirected* hypergraphs have a unique prime factor decomposition (PFD) w.r.t. to the Cartesian product, up to isomorphism and the order of the factors. A first polynomial-time factorization algorithm for undirected hypergraphs was proposed by Bretto et al. [8].

Unique prime factorization properties for *directed* hypergraphs were derived by Ostermeier et al. [25]. However, up to our knowledge, no algorithm to determine the Cartesian prime factors of a connected directed hypergraph is established, so-far.

Summary of the results In this contribution, we present an algorithm to compute the PFD of connected directed hypergraphs in $O(|V||E|r^2)$ time, where the rank r denotes the maximum number of vertices contained in the hyperedges. In addition, if we assume to have hypergraphs with bounded rank the algorithm runs in $O(|E|\log^2(|V|))$ time. Note, as directed hypergraphs are a natural generalization of undirected hypergraphs, our method generalizes and significantly improves the time-complexity of the method by Bretto et al. [8]. In fact, the algorithm of Bretto et al. has time complexity $O(|V||E|\Delta^6 r^6)$, where Δ is the maximum number of hyperedges a vertex is contained in. Assuming that given hypergraphs have bounded rank r and bounded maximum degree Δ this algorithm runs therefore in $O(|V||E|)$ time.

We shortly outline our method. Given an arbitrary connected directed hypergraph $H = (V, E)$ we first compute its so-called 2-section $[H]_2$, that is, roughly spoken the underlying *undirected* graph of H. This allows us to use the algorithm of Imrich and Peterin [21] in order to compute the PFD of $[H]_2$ w.r.t. the Cartesian *graph* product. As we will show, this provides enough information to compute the Cartesian prime factors of the directed hypergraph H. In distinction from the method of Bretto et al. our algorithm is in a sense conceptually simpler, as *(1)* we do not need the transformation of the hypergraph H into its so-called L2-section and back, where the L2-section is an edge-labeled version of the 2-section $[H]_2$, and *(2)* the test which (collections) of the factors of the 2-section are prime factors of H follows a complete new idea based on increments of fixed vertex-coordinate positions, that allows an easy and efficient check to determine the PFD of H.

2. Preliminaries

2.1. Basic definitions

A *directed hypergraph* $H = (V, E)$ consists of a finite *vertex* set $V(H) := V$ and a set of *directed hyperedges* or *(hyper)arcs* $E(H) := E$. Each arc $e \in E$ is an ordered pair of non-empty sets of vertices $e = (t(e), h(e))$. The sets $t(e) \subseteq V$ and $h(e) \subseteq V$ are called the *tail* and *head* of e, respectively. The set of vertices, that are contained in an arc will be denoted by $V(e) := t(e) \cup h(e)$. If $t(e) = h(e)$ holds for all $e \in E$, we identify e with $V(e)$, and we call $H = (V, E)$ an *undirected hypergraph*. An undirected hypergraph is an *undirected graph* if $|V(e))| = 2$ for all $e \in E$. The elements of E are called simply edges, if we consider an undirected graph. The hypergraph with $|V| = 1$ and $E = \emptyset$ is denoted by K_1 and is called *trivial*.

Throughout this contribution, we only consider hypergraphs without multiple hyperedges and thus, being E a usual set, and without loops, that is, $|V(e)| > 1$ holds for all $e \in E$. However, we allow to have hyperedges being properly contained in other ones, i.e., we might have arcs $e, f \in E$ with $t(e) \subseteq t(f)$ and $h(e) \subseteq h(f)$.

A *partial hypergraph* or *sub-hypergraph* $H' = (V', E')$ of a hypergraph $H = (V, E)$, denoted by $H' \subseteq H$, is a hypergraph such that $V' \subseteq V$ and $E' \subseteq E$. The partial hypergraph $H' = (V', E')$ is *induced (by V')* if $E' = \{e \in E \mid V(e) \subseteq V'\}$. Induced hypergraphs will be denoted by $\langle V' \rangle$.

A *weak path P (joining the vertices $v_0, v_k \in V$)* in a hypergraph $H = (V, E)$ is a sequence $P = (v_0, e_1, v_1, e_2, \ldots, e_k, v_k)$ of distinct vertices and arcs of H, such that $v_0 \in V(e_1)$, $v_k \in V(e_k)$ and $v_j \in V(e_j) \cap V(e_{j+1})$. A hypergraph H is said to be *weakly connected* or simply *connected* for short, if any two vertices of H can be joined by a weak path. A *connected component* of a hypergraph H is a connected sub-hypergraph $H' \subseteq H$ that is maximal w.r.t. inclusion, i.e., there is no other connected sub-hypergraph $H'' \subseteq H$ with $H' \subsetneq H''$. Usually, we identify connected components $H' = (V', E')$ of H simply by their vertex set V', since $\langle V' \rangle = H'$.

A *homomorphism* from $H_1 = (V_1, E_1)$ into $H_2 = (V_2, E_2)$ is a mapping $\phi : V_1 \to V_2$ such that $\phi(e)$ is an arc in H_2 whenever e is an arc in H_1 with the property that $\phi(t(e)) = t(\phi(e))$ and $\phi(h(e)) = h(\phi(e))$. A bijective homomorphism ϕ whose inverse function is also a homomorphism is called an *isomorphism*.

The *rank* of a hypergraph $H = (V, E)$ is $r(H) = \max_{e \in E} |V(e)|$.

The 2-*section* $[H]_2$ of a (directed) hypergraph $H = (V, E)$ is the undirected graph (V, E') with $E' = \{xy := \{x, y\} \subseteq V \mid x \neq y, \exists\, e \in E : x, y \subseteq V(e)\}$. In other words, two vertices are linked by an edge in $[H]_2$ if they belong to the same hyperarc in H. Thus, every arc $e \in E$ of H is a *complete graph* in $[H]_2$, i.e., all pairwise different vertices in $V(e)$ are linked by an edge in $[H]_2$. Complete graphs defined on a vertex set V will be denoted by $K_{|V|}$.

We will also deal with equivalence relations, for which the following notations are needed. For an equivalence relations R we write $\varrho \in R$ to indicate that ϱ is an equivalence class of R. A relation Q is finer than a relation R while the relation R is coarser than Q if $(e, f) \in Q$ implies $(e, f) \in R$, i.e., $Q \subseteq R$. In other words, for each class ϱ of R there is a collection $\{\chi \mid \chi \subseteq \varrho\}$ of Q-classes, whose union equals ϱ. Equivalently, for all $\varphi \in Q$ and $\psi \in R$ we have either $\varphi \subseteq \psi$ or $\varphi \cap \psi = \emptyset$.

Remark 1. *If not stated differently, we assume that the hypergraphs considered in this contribution are connected.*

2.2. The Cartesian product, (pre-)coordinates and (pre-)layers

Let H_1 and H_2 be two hypergraphs. The *Cartesian product* $H = H_1 \Box H_2$ has vertex set $V(H) = V(H_1) \times V(H_2)$, that is the Cartesian set product of the vertex sets of the factors and the arc set

$$E(H) = \big\{(\{x\} \times t(f), \{x\} \times h(f)) \mid x \in V(H_1),\, f \in E(H_2)\big\} \cup \big\{(t(e) \times \{y\}\ ,\ h(e) \times \{y\}) \mid e \in E(H_1),\, y \in V(H_2)\big\}.$$

Thus, the tuple $(\{(x_i, y_i), i \in I\}, \{(x'_j, y'_j), j \in J\})$ with $x_i, x'_j \in V(H_1),\ \ y_i, y'_j \in V(H_2),\ \ i \in I, j \in J$ is an arc in $E(H_1 \Box H_2)$ if and only if either

(i) $(\{x_i, i \in I\}, \{x'_j, j \in J\})$ is an arc in $E(H_1)$ and $y_i = y'_j$ for all $i \in I$, $j \in J$ or

(ii) $(\{y_i, i \in I\}, \{y'_j, j \in J\})$ is an arc in $E(H_2)$ and $x_i = x'_j$ for all $i \in I$, $j \in J$.

The Cartesian product is associative, commutative, and the trivial one-vertex hypergraph K_1 without arcs serves as unit [18,25]. Thus, for arbitrary finitely many factors $\{H_i, i \in I\}$ the product $\Box_{i\in I} H_i$ is well-defined, and each vertex $x \in V(\Box_{i\in I} H_i)$ is properly "coordinatized" by the vector $(x_i)_{i\in I}$ whose entries are the vertices x_i of the factors H_i.

A nontrivial hypergraph H is *prime* with respect to the Cartesian product if it cannot be represented as the Cartesian product of two nontrivial hypergraphs. A *prime factor decomposition (PFD)* of H is a representation as a Cartesian product $H = \Box_{i\in I} H_i$ such that all factors H_i, $i \in I$, are prime and $H_i \not\simeq K_1$. Note, the number k of prime factors of $H = (V, E)$ is bounded by $log(|V|)$, since every Cartesian product of k non-trivial hypergraphs has at least 2^k vertices.

Two important results concerning the Cartesian products of hypergraphs are given now.

Lemma 2.1. *(See [25].) The Cartesian product $H = \Box_{i=1}^{n} H_i$ of directed hypergraphs is connected if and only if all of its factors H_i are connected.*

Theorem 2.2. *(See [25].) Connected (directed) hypergraphs have a unique prime factor decomposition with respect to the Cartesian product.*

We will show, that the PFD of a hypergraph H can be obtained from the PFD of its 2-section $[H]_2$. For this the following lemma is crucial.

Lemma 2.3. *If $\Box_{i\in I} H_i$ is an arbitrary factorization of H it holds that $[H]_2 = \Box_{i\in I}[H_i]_2$.*

Proof. Since the Cartesian product is commutative and associative it suffices to prove the statement for two factors. Assume that $H = H_1 \Box H_2$ and every vertex x has coordinates (x_1, x_2). Thus, there is an isomorphism $\phi : V(H) \to V(H_1 \Box H_2)$ via $x \mapsto (x_1, x_2)$. We show that ϕ is also an isomorphism for the graphs $[H]_2$ and $[H_1]_2 \Box [H_2]_2$.

The edge xy is contained in $E([H]_2)$ if and only if there is an arc $e \in E(H)$ with $x, y \in V(e) = t(e) \cup h(e)$ if and only if (i) $x_1 = y_1$ and $x_2 y_2 \subset V(f) \in E(H_2)$ or (ii) $x_2 = y_2$ and $x_1 y_1 \subset V(f) \in E(H_1)$ if and only if (i) $x_1 = y_1$ and $x_2 y_2 \in E([H_2]_2)$ or (ii) $x_2 = y_2$ and $x_1 y_1 \in E([H_1]_2)$ if and only if the edge $\phi(x)\phi(y)$ is contained in $E([H_1]_2 \Box [H_2]_2)$. □

Now, given the PFD of $H = \Box_{i\in I} H_i$, we can infer that $[H]_2 = \Box_{i\in I}[H_i]_2$. However, the factors $[H_i]_2$ might not be prime w.r.t. the Cartesian *graph* product and hence, $[H]_2$ might have more prime factors. Since the PFD of $[H]_2 = \Box_{j\in J} G_j$ is unique it follows that the 2-section of the prime factors H_i of H is a combination of the prime factors G_j of $[H]_2$, that is, $[H_i]_2 \simeq \Box_{j\in J'} G_j,\ J' \subseteq J$ for all $i \in I$.

Our algorithm will start with the PFD of $[H]_2 = \Box_{j\in J} G_j$ w.r.t. the Cartesian product of undirected graphs and then tries to combine the respective prime factors of $[H]_2$ to reconstruct the prime factors of H. In other words, we need to find suitable subsets $J' \subseteq J$ so that $[H_i]_2 \simeq \Box_{j\in J'} G_j$ and H_i is a prime factor of H. To this end, we will introduce (pre-)coordinates and (pre)-layers.

Definition 2.4 *((Factorization) coordinatization). Let H be isomorphic to some product $\square_{i\in I} H_i$, where each factor H_i has vertex set $\{1,\dots,l_i\}$. A* factorization coordinatization *or* coordinatization *for short, is an isomorphism Υ from H to $\square_{i\in I} H_i$. Thus, Υ assigns to a vertex $v \in V(H)$ a vector of coordinates $(v_i)_{i\in I}$ where $1 \le v_i \le l_i$ is a vertex in $V(H_i)$*

Hence, a coordinatization gives in an explicit way the information of the underlying product structure of H. Hence, to find a factorization of H one can equivalently ask for a coordinatization of H, a fact that we will utilize in our algorithm. Note that the coordinatization w.r.t. a given product decomposition is unique up to relabeling the vertices in each factor H_i.

We will also need a notion which is similar to a coordinatization but is implied by a factorization of the 2-section $[H]_2$ rather than a decomposition of H.

Definition 2.5 *(Pre-coordinatization). Let H be a given hypergraph and assume that $[H]_2$ has a coordinatization $\Upsilon\colon V([H]_2) \to \bigtimes_{i\in I}\{1,\dots,l_i\}$. Since $V([H]_2) = V(H)$ we infer that Υ is a bijective map on $V(H)$ that assigns to each vertex $v \in V(H)$ a unique coordinate-vector $(v_i)_{i\in I}$ where $1 \le v_i \le l_i$. This map is called* pre-coordinatization *of H.*

For convenience, we will usually omit the function Υ and identify every vertex $v \in V(H)$ with its (pre-)coordinate vector, i.e., we will write $v = (v_i)_{i\in I}$ rather than $\Upsilon(v) = (v_i)_{i\in I}$.

Definition 2.6 *(Layers and pre-layers). Let $H \simeq \square_{i\in I} H_i$ with given respective coordinatization $v = (v_i)_{i\in I} \in V(H)$ and $I' \subseteq I$. The I'-*layer through *v (denoted by $H^v_{I'}$) with respect to this coordinatization is the sub-hypergraph induced by the vertices $\{u = (u_i)_{i\in I} \mid i \notin I' \Longrightarrow u_i = v_i\}$, i.e., we fix all coordinates except those contained in the set I'. Note, $H^v_{I'} \simeq \square_{i\in I'} H_i$.*

Analogously, as layers are defined by means of a coordinatization define the pre-layers *by means of a pre-coordinatization.*

For simplicity we write H^v_i instead of $H^v_{\{i\}}$ and i-(pre-)layer rather than $\{i\}$-(pre-)layers.

For later reference, we need the following observation and lemma. If $H \simeq \square_{i\in I} H_i$ and e is an arc of H, then all vertices in $V(e)$ are contained in the same i-layer H^v_i for some $i \in I$ and $v \in V(H)$, i.e. they only differ in the i-th coordinate. The same is true for pre-layers.

Lemma 2.7. *Let H be a hypergraph and let Υ be a pre-coordinatization of H. Then every arc e of H contains vertices of exactly one pre-layer w.r.t Υ, that is, all vertices in $V(e)$ only differ in the same i-th coordinate.*

Proof. Every hyperarc e forms a complete subgraph $K_{|V(e)|}$ in $[H]_2$. Moreover, complete subgraphs must be contained entirely in one of the i-layers of $[H]_2$, as complete graphs are so-called S-prime graphs, see e.g. [4,15,17,24]. Hence, each hyperarc is contained in one i-pre-layer of H. □

Note, any isomorphism from $[H]_2$ to $\square_{i\in I}[H_i]_2$ and thus, a pre-coordinatization of H, is a coordinatization of H if and only if H has a factorization $\square_{i\in I} H_i$. Lemma 2.3 immediately implies that every coordinatization of H is also a pre-coordinatization of H, while the converse is not true in general. On the other hand, we have the following result for so-called increments of coordinates.

Definition 2.8 *(Increments of coordinates). Given a pre-coordinatization $\Upsilon\colon V(H) \to \bigtimes_{i\in I}\{1,\dots,l_i\}$, of H and a vertex $v = (v_1,\dots,v_i,\dots,v_k) \in V(H)$ we define $\mathrm{inc}(v,i)$ (w.r.t. Υ) as the vertex with coordinates $(v_1,\dots,v_i+1,\dots,v_k)$ where we set $v_i + 1 := 1$ if $v_i + 1 > l_i$.*

For an (ordered) set of vertices $W \subset V$ we define the (ordered) set $\mathrm{inc}(W,i)) = \{\mathrm{inc}(w,i) \mid w \in W\}$.

Finally, we denote for an arc $e = (t(e), h(e))$ its increment $(\mathrm{inc}(t(e),i), \mathrm{inc}(h(e),i))$ by $\mathrm{inc}(e,i)$.

Lemma 2.9. *Let $H = (V,E)$ be a hypergraph and $\Upsilon\colon V \to \bigtimes_{i\in I}\{1,\dots,l_i\}$ be a pre-coordinatization of H. If for each arc $e \in E$ (where the vertices of e differ only in the j-th coordinate), there is an arc $\mathrm{inc}(e,i) \in E$ for all $i \neq j$, then Υ is a coordinatization of H.*

Proof. By Lemma 2.7, all vertices within one arc $e \in E$ differ in precisely one coordinate. Let $e \in E$ be an arbitrary hyperarc and assume the vertices differ in the j-th coordinate.

Let $\mathcal{H}_j$ be the set of j-pre-layers contained in H. Let $i \neq j$ be an arbitrary index $i \in I$. Assume that for each hyperedge e contained in some j-pre-layer $H(1)$ all "incremental copies" $\mathrm{inc}(e,i)$ are also contained in E, then there is a homomorphism from $H(1) = \langle V(H(1))\rangle$ to $H(2) = \langle (inc(V(H(1)),i))\rangle$, where $H(2)$ corresponds to some other j-pre-layer. Assume that for all such "consecutive" j-pre-layer there is a homomorphism from $H(l)$ to $H(l+1)$, $1 \le l \le l_i - 1$. By construction, after $l_i - 1$ incremental steps we arrive at the l_i-th j-pre-layer $H(l_i)$ and hence, $H(1) = \langle (inc(V(H(l_i)),i))\rangle$. If there is a homomorphism from $H(l_i)$ to $H(1)$, then there is trivially an isomorphism between all such j-pre-layers $H(1), H(2), \dots, H(l_i) \in \mathcal{H}_j$. Thus, if for all arcs $e \in E$, where the vertices of e differ precisely in this j-th coordinate, there is a hyperarc $\mathrm{inc}(e,i) \in E$ for all $i \neq j$, then there isomorphism between all j-pre-layers contained in $\mathcal{H}_j$ for this fixed $j \in I$.

If this is true for *all* arcs $e \in E$, and thus, for *all* i-pre-layers with $i \in I$, then *all* such i-pre-layers are isomorphic for each $i \in I$.

In particular, we can define for vertices v, w and an index $i \in I$ the map $g_i^{vw}: H_j^v \to H_j^w$ which maps every vertex in H_i^v to the unique vertex in H_i^w with the same i-coordinate. By the preceding arguments, for each $i \in J$ the map g_i^{vw} is an isomorphism between the i-pre-layers in H for all $v, w \in V(H)$.

Finally, assume for contradiction that Υ is a not a coordinatization and hence, Υ is not an isomorphism from H to any product $\square_{i\in I} H_i$. Hence, there must be some $i \in I$ such that not all i-layers are isomorphic by means of g_i^{vw}, a contradiction. □

Lemma 2.7 allows defining an equivalence relation R_Υ on the hyperedge set $E(H)$ for a given pre-coordinatization $\Upsilon: V(H) \to \bigtimes_{i\in I}\{1, \dots, l_i\}$ of H, as follows: $(e, f) \in R_\Upsilon$ if $e \in H_i^v$ and $f \in H_i^w$ for some $i \in I$ and $v, w \in V(H)$. In other words, e and f are in relation R_Υ if they are both contained in the i-pre-layers for the same fixed $i \in I$. Note, in case that Υ is a coordinatization the relation R_Υ is also known as product relation, that is, each equivalence of R_Υ contains the hyperedges of all copies of some (not necessarily prime) factor of H. In order to avoid confusion, we sometimes write that $R_\Upsilon(H)$ to indicate that R_Υ is defined on the edge set of H.

Given two pre-coordinatizations Υ_1 and Υ_2, we say that Υ_1 is finer than Υ_2, while Υ_2 coarser than Υ_1 if R_{Υ_1} is finer than R_{Υ_2}. We can immediately infer the next result.

Lemma 2.10. *Let H be a hypergraph and let Υ_1 and Υ_2 be pre-coordinatizations of H. Then Υ_1 is finer than Υ_2 if and only if the factorization of $[H]_2$ corresponding to Υ_2 can be obtained from the factorization $\square_{i\in I} G_i$ corresponding to Υ_1 by combining some of these factors, i.e., I can be partitioned into $I_1, \dots, I_l$ so that the pre-coordinatization Υ_2 is an isomorphism between $[H]_2$ and $\square_{j=1}^{l} G'_j$ where $G'_j \simeq \square_{i\in I_j} G_i$.*

Proof. By definition Υ_1 is finer than Υ_2 if and only if R_{Υ_1} is finer than R_{Υ_2}. By definition of the Cartesian product and by Theorem 2.2 this is the case if and only if every factor in the factorization corresponding to Υ_2 is a combination of the factors in the coordinatization Υ_1. □

Remark 2. *By Lemma 2.3, every coordinatization of H is also a pre-coordinatization of H. This implies together with Lemma 2.10 that* every *pre-coordinatization of H can be achieved by starting with the pre-coordinatization w.r.t. the prime factorization of $[H]_2$ and then combining the corresponding pre-layers of H to obtain the layers w.r.t. the prime factorization of H. In other words, one needs to find a partition $\biguplus_{j\in J} I_j$ of the index set I and then combine all pre-layers corresponding to indices in the same part into one.*

As we shall see later, in our algorithm we will *only* check increments w.r.t. the pre-coordinatization Υ' coming from the PFD of $[H]_2$. However, we have to prove that this is indeed sufficient (Theorem 3.1). In order to apply Lemma 2.9 to validate whether we end up with a coordinatization Υ of H we would need to check increments with respect to this coarser (pre-)coordinatization Υ. Now one might hope that increments with respect to the coarser pre-coordinatization are automatically increments with respect to the finer pre-coordinatization Υ' or that at least the coarser pre-coordinates can be chosen in a suitable way. However, this is not the case as the following example shows.

Assume that at some point we need to combine i- and j-pre-layers of sizes 3 and 4 respectively. The resulting k-pre-layers with respect to the new coordinatization will each contain 12 vertices labeled $1, \dots, 12$. We now claim that no matter how we assign the new labels, there is always at least one increment $\mathrm{inc}(\cdot, k)$ which is not an increment $\mathrm{inc}(\cdot, i)$ or $\mathrm{inc}(\cdot, j)$. Assume for a contradiction that all increments $\mathrm{inc}(\cdot, k)$ were either of the form $\mathrm{inc}(\cdot, i)$ or $\mathrm{inc}(\cdot, j)$. By applying $\mathrm{inc}(\cdot, k)$ recursively 12 times to a vertex, we end up at the same vertex again. This means, that we have applied $\mathrm{inc}(\cdot, i)$ a number of times which must be divisible by 3 and $\mathrm{inc}(\cdot, j)$ a number of times which must be divisible by 4. However, no suitable multiples of 3 and 4 add up to 12.

The latter example shows that the single check of increments with respect to the PFD of $[H]_2$ is not sufficient to invoke Lemma 2.9 to conclude that some coarser pre-coordinatization is indeed a coordinatization. For this purpose, we need the following additional lemma.

Lemma 2.11. *Let $H = (V, E)$ be a hypergraph, let Υ_1, Υ_2 be pre-coordinatizations of H such that Υ_1 is finer than Υ_2. Let inc_1 and inc_2 the respective increment maps. Assume that for each arc $e \in E$ (where the vertices of e differ only in the j-th coordinate w.r.t. Υ_2), there is an arc $\mathrm{inc}_1(e, i) \in E$ for all $i \notin I_j$ (where I_j is defined as in Lemma 2.10). Then there is an arc $\mathrm{inc}_2(e, k)$ for all $k \neq j$ and hence Υ_2 is a coordinatization of H.*

Proof. The coordinates of the vertices in $\mathrm{inc}_2(e, k)$ w.r.t. Υ_1 can be obtained from those of vertices in e by only changing coordinates outside I_j. This can be achieved by successive applications of $\mathrm{inc}_1(\cdot, i)$ for $i \notin I_j$. Since we started at an edge (namely e) and each of those applications takes edges to edges we also end at an edge $\mathrm{inc}_2(e, k)$. Hence, Lemma 2.9 implies that Υ_2 is a coordinatization. □

Algorithm 1 `PFD_of_Di-Hypergraphs`.

1: **INPUT:** A hypergraph $H = (V, E)$;
2: $H \leftarrow$ `Preprocessing`(H); ▷∗*Now,* $H = (V_{\text{lex}}, E_{\text{lex}})$ *and vertex coordinates are known.*
3: Let $G_{\text{aux}} = (\{1, \ldots, k\}, \emptyset)$ ▷∗*Here,* k *is the number of prime factors of* $[H]_2$.;
4: **for** each $e \in E_{\text{lex}}$ **do**
5: Let $j \in \{1, \ldots, k\}$ be the *(unique)* coordinate where distinct $x, y \in V(e)$ differ;
6: **for** $i \in I = \{1, \ldots, k\} \setminus \{j\}$ **do**
7: **if** $\text{inc}(e, i) \notin E_{\text{lex}}$ **then**
8: add edge ij to G_{aux};
9: **end if**
10: **end for**
11: **end for**
12: `Combine`$(H = (V_{\text{lex}}, E_{\text{lex}}), G_{\text{aux}})$;
13: **OUTPUT:** PFD $\square_{i=1}^{n} H_i$ of H;

Algorithm 2 `Preprocessing`.

1: **INPUT:** A connected hypergraph $H = (V, E)$;
2: Compute $\square$-PFD of $[H]_2 = \square_{i=1}^{k} G_i$ and vertex-coordinates with the Imrich–Peterin-Algorithm [21];
3: Compute the list V_{lex} of lexicographic ordered vertices (w.r.t. their coordinates);
4: **for** each $e \in E$ **do**
5: Reorder $t(e)$ and $h(e)$ w.r.t. the lexicographic order of the vertices;
6: **end for**
7: Compute the list E_{lex} of lexicographic ordered arcs; w.r.t. to the lexicographic ordered sets $t(e)$ and then $h(e)$;
8: **return** $(V_{\text{lex}}, E_{\text{lex}})$ with respective vertex-coordinates;

Algorithm 3 `Combine`.

1: **INPUT:** A hypergraph $H = (V, E)$ with pre-coordinates w.r.t the PFD of $[H]_2$, a graph G_{aux};
2: Compute connected components $I_1, \ldots, I_l$ of G_{aux};
3: **for** each $e \in E_{\text{lex}}$ **do**
4: Let $j \in \{1, \ldots, k\}$ be the *(unique)* coordinate where distinct $x, y \in V(e)$ differ;
5: Let I_l be the connected component containing vertex j;
6: Assign color l to hyperedge e;
7: **end for**
8: compute $[H]_2$ where edges ij obtains the unique color of e where $i, j \in V(e)$;
9: compute coordinates of all vertices in V in $[H]_2$; ▷∗*This pre-coordinatization is the PFD-coordinatization of* H
10: **return** $\square$-PFD $\square_{i=1}^{n} H_i$ of H;

3. PFD-algorithm for directed hypergraphs

3.1. Workflow

We give here a summary of the workflow of the algorithm to compute the prime factor of connected directed hypergraphs. The top-level control structure is summarized in Algorithm 1 `PFD_of_Di-Hypergraphs` in which the subroutines `Preprocessing` (Algorithm 2) and `Combine` (Algorithm 3) are used.

As input of `PFD_of_Di-Hypergraphs` a connected hypergraph $H = (V, E)$ is expected. First of all, subroutine `Preprocessing` is called. Here, the PFD of $[H]_2 = \square_{i=1}^{k} G_i$ and the respective coordinatization Υ of $[H]_2$ is computed by application of the algorithm of Imrich and Peterin [21]. Then the vertices, the vertices within the arcs and the arcs are ordered in lexicographic order. This helps to achieve the desired time-complexity in later steps.

By definition, coordinatization Υ of $[H]_2$ is a pre-coordinatization of H. By construction of Υ and Lemma 2.3, the pre-coordinatization Υ is at least as fine as the coordinatization of H w.r.t. its PFD. By Remark 2 it suffices to find a suitable partition of $I = \{1, \ldots, k\}$ to derive the prime factors of H. To this end, we initialize in Line 3 of Algorithm 1 the auxiliary graph G_{aux}, where each vertex i represents an element of I. The edge set is left empty. We might later add edges in order trace back which equivalence classes of R_Υ have to be combined, i.e., all vertices within one connected components of G_{aux} will then be in one class of the respective partition of I.

We continue to check in the for-loop in Lines 4–11 of Algorithm 1, if for each arc $e \in E_{\text{lex}}$ that is contained in some j-layer its "copies" are also contained in "incremental-neighboring" j-layers, i.e., we check if $\text{inc}(e, i) \in E_{\text{lex}}$. If this is not the case, then we add the edge ij to G_{aux}. Finally, we use the information of the connected components $I_1, \ldots, I_l$ of G_{aux} that partition the set I in order to determine the prime factors of the given hypergraph H. To this end, the subroutine `Combine` is called and an edge-colored 2-section $[H]_2$ is computed. That is, each edge $e \in E([H]_2)$ that is contained in the copy of factor G_i with $i \in I_s$ obtains color s. In other words, all prime factors G_j, $j \in I_s$ of $[H]_2$ are combined to a single factor of $[H]_2$ and the edges in the respective I_s-layers obtain color s. W.r.t. this coloring it is possible to efficiently determine new

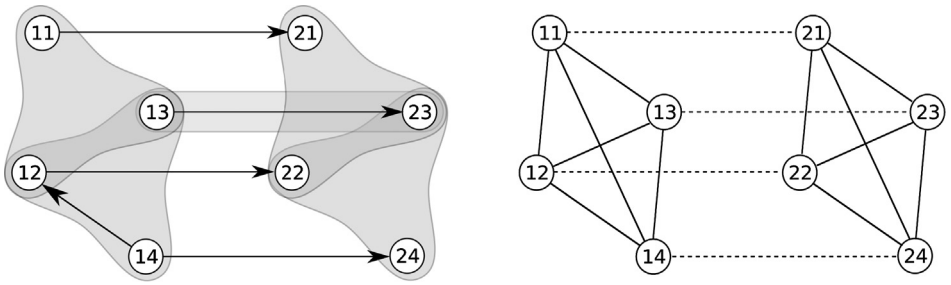

Fig. 1. The prime hypergraph H (left-hand side) with arc set $E(H) = \{e_1 = (\{11, 12, 13\}, \{12, 13, 14\}), KA : e_2 = (\{21, 22, 23\}, \{22, 23, 24\}), e_3 = (\{14\}, \{12\}), e_4 = (\{11\}, \{21\}), e_5 = (\{12\}, \{22\}), e_6 = (\{13, 23\}, \{13, 23\}), e_7 = (\{14\}, \{24\}), e_8 = (\{13\}, \{23\})\}$ admits a non-trivial prime factorization of its 2-section $[H]_2$ (right-hand side) into $K_4 \square K_2$. The vertices of H are labeled w.r.t. its pre-coordinatization given by the coordinatization of $[H]_2$. For the first arc e_4 of the lexicographic ordered arc set $E_{\text{lex}} = \{e_4, e_1, e_5, e_8, e_6, e_3, e_7, e_2\}$ we can observe that the increment $\text{inc}(e_4, 2) = (\{\{12\}, \{22\}) = e_5 \in E_{\text{lex}}$. Analogously, $\text{inc}(e_1, 1) = e_2$, $\text{inc}(e_5, 2) = e_8$, and $\text{inc}(e_8, 2) = e_7$ are all contained in E_{lex}. However, when we arrive at the hyperedge e_6 we obtain that $\text{inc}(e_6, 2) = (\{14, 24\}, \{14, 24\})$ is not an arc of E_{lex}. Hence, applying Algorithm 1 would lead to an edge 12 in G_{aux}, resulting in a connected auxiliary graph and, H would be determined as prime. A second example for an arc $e \in E_{\text{lex}}$ with $\text{inc}(e, i) \notin E_{\text{lex}}$ is the edge $e = e_7$.

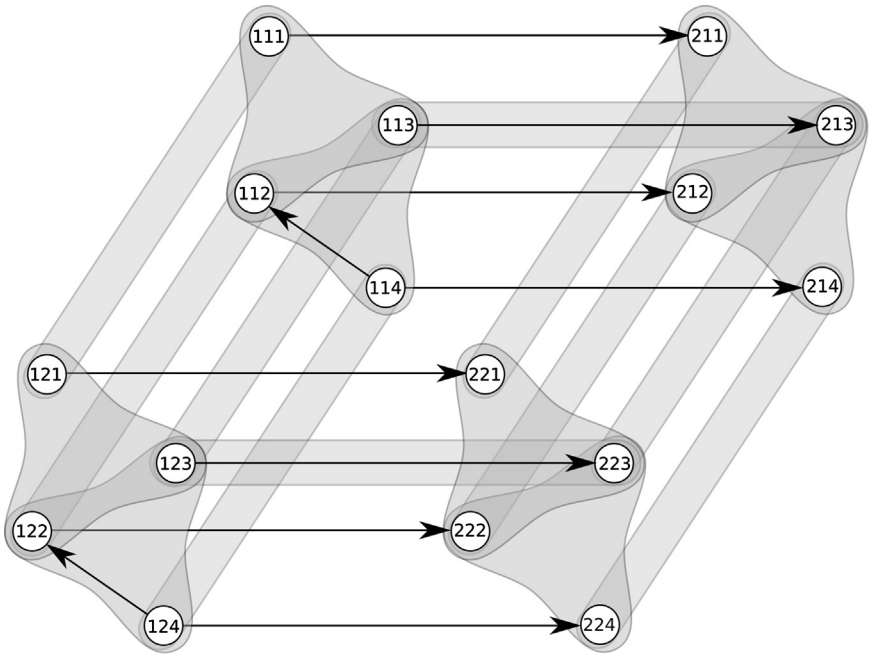

Fig. 2. The non-prime hypergraph $H = H_1 \square H_2$ is the product of the directed prime hypergraph H_1 in Fig. 1 and an undirected hypergraph H_2 with two vertices and one hyperedge. The vertices of H are labeled w.r.t. its pre-coordinatization given by the coordinatization of $[H]_2$. The auxiliary graph G_{aux} is initialized as the graph with three vertices and empty edge-set in Algorithm 1. While the increment $\text{inc}(e, 2) = \{(123, 223), (123, 223)\}$ of the arc $e = \{(113, 213), (113, 213)\}$ is still contained in E_{lex}, the increment $\text{inc}(e, 3) = \{(124, 224), (124, 224)\}$ is not. Hence, the edge 13 is added to G_{aux}. Since the increments of all hyperedges of the form $\{(i1j, i2j), (i1j, i2j)\}$ are contained in E_{lex}, no further edges will be added to G_{aux}. Hence, the sub-hypergraph induced by vertices with identical 2nd coordinate, i.e., the $\{1, 3\}$-layers, constitute the copies the prime factor H_1, while the sub-hypergraphs induced by vertices with identical 1st and 3rd coordinate, i.e., the 2-layers, are copies of the prime factor H_2.

vertex coordinates in $[H]_2$ which is then a factorization coordinatization of H. We will show, that this leads to a "finest" coordinatization of H and hence, to the prime factors $H_1, \ldots, H_r$ of H.

For illustrative examples see Figs. 1 and 2.

3.2. Correctness

We are now in the position to prove the correctness of the algorithm `PFD_of_Di-Hypergraphs`, summarized in the following theorem.

Theorem 3.1. *Algorithm 1 is sound and complete.*

Proof. Given a hypergraph $H = (V, E)$. We start with a preprocessing and call in Algorithm 1 the Algorithm 2. Here, the PFD of $[H]_2 = \square_{i=1}^{k} G_i$ and the respective coordinatization Υ of $[H]_2$ is computed. This coordinatization Υ is by definition a pre-coordinatization of H. Since $[H]_2$ is an undirected graph, it is allowed to apply the algorithm of Imrich and Peterin [21]. Finally, the vertices, the vertices within the arcs and the arcs are ordered in lexicographic order. The latter task is not important for the correctness of the algorithm, but for the time-complexity that we will consider later on.

We are now in Line 3 of Algorithm 1. By construction of Υ and Lemma 2.3, the pre-coordinatization Υ is at least as fine as the coordinatization of H w.r.t. its PFD. By Lemma 2.10 it suffices to find a suitable partition of $I = \{1, \dots, k\}$. To this end, we initialize in the auxiliary graph G_{aux} where each vertex i represents an element of I. The edge set is left empty. We might later add edges in order trace back which equivalence classes have to be combined, i.e., all vertices within one connected components of G_{aux} will then be in one class of the respective partition of I.

Now consider the for-loop in Lines 4–11. For each $e \in E_{\text{lex}}$ we check in which coordinates the vertices in $V(e)$ differ. Since Υ is a pre-coordinatization and by Lemma 2.7, this is exactly one coordinate for each hyperarc. Let $e \in E_{\text{lex}}$ be a chosen arc and assume that all vertices in $V(e)$ differ in the j-th coordinate. Now, it is checked if for arc e its "copies" are contained in each G_j^w-layer where $w \in inc(V(e), i)$. If for some arc $e \in E_{\text{lex}}$ we observe that there is no hyperedge $\text{inc}(e, i) = (\text{inc}(t(e), i), \text{inc}(h(e), i) \in E_{\text{lex}}$ then there is no "copy" of e in some j-pre-layer through w with $w \in inc(V(e), i)$. In this case we add the edge ij to G_{aux} if not already set. The latter tasks are repeated for all hyperarcs $e \in E_{\text{lex}}$.

Finally, in Line 12 the Algorithm 3 is called. The task of this subroutine is to combine the pre-coordinates and thus, the pre-layers in order to determine the layers of the final prime-factors of H. Let $I_1, \dots, I_r$ be the connected components of G_{aux}. Clearly, $\Pi = \{I_1, \dots, I_r\}$ is a partition of I. Let each I_j having l_j elements. Lemma 2.10 and Remark 2 imply that $\widehat{\Upsilon}: V(H) \to \bigtimes_{l=1}^{r} \{1, \dots, l_r\}$ is a pre-coordinatization of H. It remains to show that

(1.) $\widehat{\Upsilon}$ is a coordinatization and
(2.) $\widehat{\Upsilon}$ is at least as fine as the coordinatization given by the PFD of H.

Claim (1.): By construction, all $e \in E_{\text{lex}}$ where the vertices differ in the i-th coordinate w.r.t. Υ are now contained in some I_s-layer where $i \in I_s \in \Pi$. Moreover, for all e in some I_s-layer the increments $\text{inc}(e, j)$ with $j \neq i$ and $j \notin I_s$ must be contained in H, as otherwise we would have added the edge ij to G_{aux} and hence, $j \in I_s$. As the latter is true for all I_s-layers contained in Π we can apply Lemma 2.11 and conclude that $\widehat{\Upsilon}$ is a coordinatization of H.

Claim (2.): Given the pre-coordinatization Υ of H. By construction of Υ and Lemma 2.3, Υ is at least as fine as the coordinatization of H w.r.t. its PFD. Thus, there is a partition $\Pi' = \{I'_1, \dots, I'_t\}$ of I w.r.t. the PFD of H. It remains to show that if there are two indices $i, j \in I_s \in \Pi$, then i, j are also contained in the same class of Π'. If $i, j \in I_s \in \Pi$ then they are in same connected component C_s of G_{aux}. Hence, it suffices to consider pairs $i, j \in I_s \in \Pi$ that are connected by an edge. Assume, for contradiction that i, j are in different classes of Π'. W.l.o.g. let $i \in I'_1$ and $j \in I'_2$. Moreover, let $H = H_{I'_1} \square H_{\cup_{l \geq 2} I'_l}$ Hence, for *all* $l \in I'_1$ and thus, in particular for $l = i$ it holds that for all arcs e in some I'_2-layer there is an arc $(\text{inc}(t(e), i), \text{inc}(h(e), i) \in E_{\text{lex}}$. The same holds with the role of i and j switched. However, in this case we would not add the edge ij to G_{aux}, a contradiction.

To finish the PFD-computation we have to compute $\widehat{\Upsilon}$. To this end, we compute the 2-section $[H]_2$ with edges xy colored with color j whenever x and y are contained in some edge e that is contained in some j-layer of H. Lemma 2.3 implies that $\widehat{\Upsilon}$ is also a coordinatization of $[H]_2$ and hence, all edges with same color j in $[H]_2$ are contained in the same equivalence class of $R_{\widehat{\Upsilon}}([H]_2)$. In other words, $(e, f) \in R_{\widehat{\Upsilon}}(H)$ if and only if $(xy, uv) \in R_{\widehat{\Upsilon}}([H]_2)$ for all distinct $x, y \in V(e)$ and distinct $u, v \in V(f)$. By construction, $R_{\widehat{\Upsilon}}([H]_2)$ is a product relation of $[H]_2$, and thus we can apply again a method proposed the by Imrich and Peterin (cf. Theorem 5.1. in [21]), in order to obtain the desired coordinates and hence, $\widehat{\Upsilon}$. □

3.3. Time complexity

In order to prove the time-complexity results, we first give the following lemma.

Lemma 3.2. *Let H be a hypergraph, let $[H]_2 = \square_{i=1}^{k} H_i$ be a factorization of its 2-section into k factors, and let m and n be the number of arcs and vertices of H, respectively. Then for any $l \in \mathbb{N}_0$ it holds that $k^l \log m = O(n)$.*

Proof. If m_i and n_i are the numbers of arcs and vertices of the factors then $m_i \leq (2^{n_i})^2$, since we have to consider tail and head independently. Let N be the maximum number of vertices of a factor. Then we have

$$m = \sum_{i=1}^{k} m_i \prod_{\substack{j=1 \\ j \neq i}}^{k} n_j \leq \sum_{i=1}^{k} 2^{n_i} \prod_{j=1}^{k} n_j \leq k \cdot 2^N \prod_{j=1}^{k} n_j.$$

Taking logarithms on both sides of the inequality gives

$$\log m \leq \log k + N + \sum_{j=1}^{k} \log(n_i) \leq k + N + k \cdot N \leq c \cdot k \cdot N$$

for some suitable constant c.

On the other hand by bounding the size of every factor except the biggest one from below by 2 we get $n = \prod_{i=1}^{k} n_i \geq N \cdot 2^{k-1}$. Clearly, $N \cdot 2^{k-1} \geq c' \cdot N \cdot k^l$ for some suitable constant c' depending on l. Together with the estimate for $\log m$ this proves the lemma. □

The next two lemmas are concerned with the time-complexity of the subroutines `Preprocessing` and `Combine`.

Lemma 3.3. *Let $H = (V, E)$ be a connected hypergraph with $|V| = n$, $|E| = m$ and rank r. Then Algorithm 2 performs in $O(r^2 mn)$ time. If we assume that H has bounded rank, then Algorithm 2 has time-complexity $O(m \log^2(n))$.*

Proof. In Line 2, the first task is the computation of the 2-section $[H]_2$. To this end, we initialize an adjacency list $N[1], \ldots, N[n]$ with empty entries, which can be done in $O(n)$ time. We add for each arc $e \in E$ and each pair $\{i, j\} \in \binom{V(e)}{2}$ the vertex i to $N[j]$ and j to $N[i]$, if these vertices are not already contained in the respective adjacency lists. Hence, we must check whether $i \in N[j]$ or not. To this end, assume that $N[j]$ is already ordered. Hence we need $O(\log(n))$ comparisons to verify if $i \in N[j]$. If this is not the case, the vertex i is added to $N[j]$ on the respective position so that $N[j]$ stays sorted. Analogously, we add j to $N[i]$, whenever j is not contained in $N[i]$. As for each arc $e \in E$ there are at most $\binom{r}{2} = O(r^2)$ pairs $\{i, j\}$ and for each such pair we have $O(\log(n))$ comparisons we end in a time-complexity of $O(n + mr^2 \log(n))$ to create the adjacency list $N[1], \ldots, N[n]$. These lists serve than as input for the algorithm of Imrich and Peterin which computes the PFD of the 2-section in $O(|E([H]_2)| + n)$ time. Since $[H]_2$ is connected and thus, $[H]_2$ has at least $|V| - 1$ edges, the PFD algorithm runs in fact in $O(|E([H]_2|) = O(mr^2)$ time. Hence, the total time complexity of Line 2 of Algorithm 2 is $O(n + mr^2 \log(n) + mr^2) = O(mr^2 \log(n))$.

In what follows, let k be the number of factors of $[H_2]$ and let each $v \in V$ be identified with its respective (pre-) coordinate vector $(v_1, \ldots, v_k)$ computed by the Imrich–Peterin-Algorithm. Note, k is bounded by $\log(n)$.

In Line 3, the list V of vertices is reordered in lexicographic order w.r.t. the vertex coordinates, i.e., $v < w$ if there is some $i \in \{1, \ldots, k\}$ with $v_j \leq w_j$ for all $j \in \{1, \ldots, i-1\}$ and $v_i < w_i$. This task can be done in $O(n \log(n) k) = O(n \log^2(n))$. Since $mr \geq n$ we obtain that $O(n \log^2(n)) = O(mr \log^2(n))$. This new ordered vertex list is called V_{lex}.

We are now concerned with the for-loop in Line 4. For each hyperedge $e \in E$ we reorder the vertices of its head and tail w.r.t. to the order of the vertices in V_{lex}. Each hyperedge contains at most r vertices and hence, this task can be done in $O(r \log(r))$ time. Therefore, the entire for-loop (Lines 4–6) takes $O(mr \log(r))$ time.

Finally, the arcs are reordered w.r.t. the lexicographic ordered sets $t(e)$ and $h(e)$. We say $e < f$ if $t(e) < t(f)$ or $t(e) = t(f)$ and $h(e) < h(f)$, whereby the tails, resp., heads are compared w.r.t. the lexicographic order of their vertices. To determine if $t(e) < t(f)$ or $h(e) < h(f)$ for some arcs $e, f \in E$, the at most $2r$ pairs of vertices must be compared, whereby the comparison of each such pair can be done in $O(1)$ time, since the vertices are already ordered in the tails and heads. The reordering of the arcs need than $O(m \log(m))$ comparisons, where each comparison can be done in $O(r)$ time, by the preceding arguments. Hence, the creation of E_{lex} takes $O(rm \log(m))$ time. By Lemma 3.2 this is $O(rmn)$. Moreover, if we assume that the rank r is bounded, then $m \leq \sum_{i=1}^{r} \binom{n}{r} \leq rn^r = O(n^r)$. Hence $O(\log(m)) = O(\log(n^r)) = O(r \log(n)) = O(\log(n))$. In this case the time complexity for determining E_{lex} is $O(m \log(m)) = O(m \log(n))$.

Taken together the latter arguments, we end in overall time complexity for Algorithm 2 of $O(r^2 mn)$ and if the rank r is bounded with $O(m \log^2(n))$. □

Lemma 3.4. *Let $H = (V, E)$ be a connected hypergraph with $|V| = n$, $|E| = m$ and rank r. Moreover, assume that the graph G_{aux} has k vertices and m' edge with $k \leq \log(n)$. Then Algorithm 3 preforms in $O(mnr^2)$ time. If we assume that H has bounded rank, then Algorithm 3 has time-complexity $O(m \log^2(n))$.*

Proof. Determining the connected components of G_{aux} in Line 2 can be done in $O(k + m') = O(\log(n) + \log^2(n))$ time by application of the classical breadth-first search. While doing this, we will in addition record in $O(1)$ time for each vertex in which connected component it is contained. Let $I_1, \ldots, I_l$ be the connected components of G_{aux}.

For each of the m arcs we have to find the indices where the vertices of the particular arc differs. To this end, it suffices to take any two vertices x and y of $V(e)$ and to compare their k coordinates which takes $O(k)$ time. Let j be the coordinate where the two vertices differ. We need to check in which of the connected components I_s the vertex is contained in, which can be done in $O(1)$ time, since we have already recorded for each vertex of G_{aux}, in which component it is contained in. Now, the color for each arc can be recorded in $O(1)$ time. Hence, the for-loop (Lines 3–7) has overall-time complexity $O(mk) = O(m \log(n))$.

To compute the 2-section in Line 8 with colored edges we initialize an extended adjacency list $N[1], \ldots, N[n]$ where whenever we add some $i \in N[j]$ we also record the respective unique color of ij as a 2nd parameter. Recording this parameter can be done in $O(1)$ time, as for each arc $e \in E$ it is known which color it has. Hence, we can argue analogously as in

the proof of Lemma 3.3, and state that the 2-section with additionally colored edges can be computed in $O(n+mr^2\log(n))$ time.

Finally, the vertex-coordinates in $[H]_2$ can be computed in $O(m')=O(mr^2)$ time, see Theorem 5.1. in [21].

Hence the overall-time complexity of Algorithm 3 is $O(\log^2(n)+m\log(n)+n+mr^2\log(n)+mr^2)=O(n+m\log^2(n)r^2)$. Since $mr\geq n$ and $\log^2(n)=O(n)$, the latter can be expressed as $O(mnr^2)$. If we assume in addition that the rank r is bounded we get $O(n+m\log^2(n)r^2)=O(mr+m\log^2(n)r^2)=O(m\log^2(n))$. □

We are now in the position to determine the time-complexity of algorithm `PFD_of_Di-Hypergraphs`.

Theorem 3.5. *Let $H=(V,E)$ be a connected hypergraph with $|V|=n$, $|E|=m$ and rank r. Then Algorithm 1 computes the PFD of H in $O(mnr^2)$ time. If the rank r is bounded the time-complexity of Algorithm 1 is $O(m\log^2(n))$.*

Proof. We suppose both the vertices and the hyperarcs of H implemented as integers and E implemented as an $m\times 2$ array, where each entry $E[e,i]$ contains the list of vertices in $t(e)$ if $i=1$ and $h(e)$ if $i=2$. In Line 2 we call `Preprocessing`(H) which takes $O(r^2mn)$ time and if r is bounded $O(m\log^2(n))$ time (Lemma 3.3).

In what follows, let $k\leq\log(n)$ be the number of factors of $[H_2]$ and assume that each $v\in V$ is identified with its respective (pre-)coordinate vector $(v_1,\ldots,v_k)$.

In Line 3 the auxiliary graph is be initialized. In particular, we initialize G_{aux} as adjacency list, i.e., we create empty lists $N[1],\ldots,N[k]$ which can be done in $O(k)$ time.

We are now concerned with the for-loop in Lines 4–11. For each of the m arcs we have to find the indices where the vertices of the particular arc differs. To this end, any two vertices x and y of $V(e)$ are chosen and their k coordinates are compared, which takes $O(k)$ time. The nested for-loop (Lines 6–10) is executed for all coordinates i where the vertices of arc e are identical and it is checked whether $\text{inc}(e,i)=(\text{inc}(t(e),i),\text{inc}(h(e),i)$ is contained in E_{lex} or not. The increment $(\text{inc}(t(e),i),\text{inc}(h(e),i)$ can be computed in $O(r)$ time. Note, the vertices within $\text{inc}(t(e),i)$ and $\text{inc}(h(e),i)$ are still lexicographically ordered as only vertex-coordinates are incremented that have been identical for the vertices within the arc and thus, their i-th positions are all still equal after the computation of $\text{inc}(e,i)$. We now check whether $(\text{inc}(t(e),i),\text{inc}(h(e),i)\notin E_{\text{lex}}$. Since E_{lex} is already ordered, binary search finds the corresponding arc using at most $O(\log(m))$ comparisons of arcs and since head and tail of each arc are in lexicographic order comparing two arcs takes $O(r)$ time. Therefore, the if-condition in Line 7 takes $O(r+r\log(m))=O(r\log(m))$ time. In case, $(\text{inc}(t(e),i),\text{inc}(h(e),i)\notin E_{\text{lex}}$ we have to add a respective edge ij in G_{aux}, if not already set. Hence, to check whether ij exists in G_{aux}, we need to validate if $i\in N[j]$. To this end, assume that $N[j]$ is already ordered. Hence we need $O(\log(k))$ comparisons to verify if $i\in N[j]$. If this is not the case i is added to $N[j]$ on the respective position so that $N[j]$ stays sorted. Similarly, j is added to $N[i]$ whenever $i\notin N[j]$. Hence the nested for-loop in (Lines 6–10) has time complexity $O(k(r\log(m)+\log(k)))=O(kr\log(m))$, since the number of arcs m is at least as big as the number (non-trivial) factors k. Take together the latter arguments, the entire for-loop in Lines 4–11 has time-complexity $O(m(k+kr\log(m)))=O(mkr\log(m))$. By Lemma 3.2 this is $O(mnr)$. Moreover, if we assume that the rank r is bounded, then $m\leq n^r$ and hence, $O(\log(m))=O(\log(n^r))=O(r\log(n))=O(\log(n))$ In this case, the time complexity of Lines 4–11 is $O(mk\log(m))=O(mk\log(n))=O(m\log^2(n))$.

Finally, we use Algorithm 3 which performs in $O(mnr^2)$ time and if the rank r is bounded it has time-complexity $O(m\log^2(n))$ (Lemma 3.4).

To summarize, each step of Algorithm 1 can be performed in $O(mnr^2)$ time and if the rank r is bounded the time-complexity is $O(m\log^2(n))$. □

Acknowledgements

We thank the organizers of the 8th Slovenian Conference on Graph Theory (2015) in Kranjska Gora, where the authors participated, met and basically drafted the main ideas of this paper, while drinking a cold and tasty red Union, or was it a green Laško?

We also thank Wilfried Imrich and Iztok Peterin for helpful comments regarding the time-complexity of our algorithm.

References

[1] F. Aurenhammer, J. Hagauer, W. Imrich, Cartesian graph factorization at logarithmic cost per edge, Comput. Complexity 2 (1992) 331–349.
[2] C. Berge, Hypergraphs: Combinatorics of Finite Sets, vol. 45, North-Holland, Amsterdam, 1989.
[3] T. Black, Monotone properties of k-uniform hypergraphs are weakly evasive, in: Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS'15, ACM, New York, NY, USA, 2015, pp. 383–391.
[4] B. Brešar, On subgraphs of Cartesian product graphs and S-primeness, Discrete Math. 282 (2004) 43–52.
[5] A. Bretto, Hypergraphs and the Helly property, Ars Combin. 78 (2006) 23–32.
[6] A. Bretto, Applications of hypergraph theory: a brief overview, in: Hypergraph Theory. Mathematical Engineering, Springer International Publishing, 2013, pp. 111–116.
[7] A. Bretto, Y. Silvestre, T. Vallée, Cartesian product of hypergraphs: properties and algorithms, in: 4th Athens Colloquium on Algorithms and Complexity, ACAC 2009, in: EPTCS, vol. 4, 2009, pp. 22–28.
[8] A. Bretto, Y. Silvestre, T. Vallée, Factorization of products of hypergraphs: structure and algorithms, Theoret. Comput. Sci. 475 (2013) 47–58.

[9] C. Crespelle, E. Thierry, T. Lambert, A linear-time algorithm for computing the prime decomposition of a directed graph with regard to the Cartesian product, in: D.-Z. Du, G. Zhang (Eds.), Computing and Combinatorics, in: Lecture Notes in Computer Science, vol. 7936, Springer, Berlin, Heidelberg, 2013, pp. 469–480.
[10] W. Dörfler, Multiple covers of hypergraphs, Ann. N.Y. Acad. Sci. 319 (1) (1979) 169–176.
[11] T. Feder, Product graph representations, J. Graph Theory 16 (1992) 467–488.
[12] J. Feigenbaum, Directed Cartesian-product graphs have unique factorizations that can be computed in polynomial time, Discrete Appl. Math. 15 (1) (1986) 105–110.
[13] J. Feigenbaum, J. Hershberger, A. Schäffer, A polynomial time algorithm for finding the prime factors of Cartesian-product graphs, Discrete Appl. Math. 12 (1985) 123–138.
[14] R. Hammack, W. Imrich, S. Klavžar, Handbook of Product Graphs, 2nd edition, Discrete Mathematics and Its Applications, CRC Press, 2011.
[15] M. Hellmuth, On the complexity of recognizing S-composite and S-prime graphs, Discrete Appl. Math. 161 (7–8) (2013) 1006–1013.
[16] M. Hellmuth, M. Noll, L. Ostermeier, Strong products of hypergraphs: unique prime factorization theorems and algorithms, Discrete Appl. Math. 171 (2014) 60–71.
[17] M. Hellmuth, L. Ostermeier, P. Stadler, Diagonalized Cartesian products of S-prime graphs are S-prime, Discrete Math. 312 (1) (2012) 74–80, Algebraic Graph Theory – a volume dedicated to Gert Sabidussi on the occasion of his 80th birthday.
[18] M. Hellmuth, L. Ostermeier, P.F. Stadler, A survey on hypergraph products, Math. Comput. Sci. 6 (1) (2012) 1–32.
[19] W. Imrich, Kartesisches Produkt von Mengensystemen und Graphen, Studia Sci. Math. Hungar. 2 (1967) 285–290.
[20] W. Imrich, Über das schwache Kartesische Produkt von Graphen, J. Combin. Theory 11 (1) (1971) 1–16.
[21] W. Imrich, I. Peterin, Recognizing Cartesian products in linear time, Discrete Math. 307 (3–5) (2007) 472–483.
[22] A. Kaveh, B. Alinejad, Hypergraph products for structural mechanics, in: B. Topping (Ed.), Proceedings of the Eleventh International Conference on Computational Structures Technology, Civil-Comp Press, Stirlingshire, UK, 2012, paper 266.
[23] A. Kaveh, B. Alinejad, Hypergraph products for structural mechanics, Adv. Eng. Softw. 80 (2015) 72–81.
[24] S. Klavžar, A. Lipovec, M. Petkovšek, On subgraphs of Cartesian product graphs, Discrete Math. 244 (2002) 223–230.
[25] L. Ostermeier, M. Hellmuth, P.F. Stadler, The Cartesian product of hypergraphs, J. Graph Theory 70 (2) (2012) 180–196.
[26] G. Sabidussi, Graph multiplication, Math. Z. 72 (1) (1960) 446–457.
[27] H. Shapiro, The embedding of graphs in cubes and the design of sequential relay circuits, Bell Telephone Laboratories Memorandum, 1953, unpublished.
[28] M. Sonntag, Hamiltonian properties of the Cartesian sum of hypergraphs, J. Inf. Process. Cybern. 25 (3) (1989) 87–100.
[29] L. Szamkołowisz, Remarks on the Cartesian product of two graphs, Colloq. Math. 9 (1) (1962) 43–47.
[30] V.G. Vizing, The Cartesian product of graphs, Vychisl. Sistemy 9 (1963) 30–43.
[31] P. Winkler, Factoring a graph in polynomial time, European J. Combin. 8 (1987) 209–212.
[32] X. Zhu, On the chromatic number of the product of hypergraphs, Ars Combin. 34 (1992) 25–31.