On Decomposability of Multilinear Sets *

Alberto Del Pia † Aida Khajavirad ‡

May 8, 2017

Abstract

We consider the Multilinear set $\mathcal{S}$ defined as the set of binary points (x, y) satisfying a collection of multilinear equations of the form $y_I = \prod_{i \in I} x_i$, $I \in \mathcal{I}$, where $\mathcal{I}$ denotes a family of subsets of $\{1, \ldots, n\}$ of cardinality at least two. Such sets appear in factorable reformulations of many types of nonconvex optimization problems, including binary polynomial optimization. A great simplification in studying the facial structure of the convex hull of the Multilinear set is possible when $\mathcal{S}$ is decomposable into simpler Multilinear sets $\mathcal{S}_j$, $j \in J$; namely, the convex hull of $\mathcal{S}$ can be obtained by convexifying each $\mathcal{S}_j$, separately. In this paper, we study the decomposability properties of Multilinear sets. Utilizing an equivalent hypergraph representation for Multilinear sets, we derive necessary and sufficient conditions under which $\mathcal{S}$ is decomposable into $\mathcal{S}_j$, $j \in J$, based on the structure of pair-wise intersection hypergraphs. Our characterizations unify and extend the existing decomposability results for the Boolean quadric polytope. Finally, we propose a polynomial-time algorithm to optimally decompose a Multilinear set into simpler subsets. Our proposed algorithm can be easily incorporated in branch-and-cut based global solvers as a preprocessing step for cut generation.

Key words: Multilinear functions; Convex hull; Decomposition; Zero-one polynomial optimization; Factorable relaxations; Polynomial-time algorithm

1 Introduction

Central to the efficiency of global optimization algorithms is their ability to construct sharp and cheaply computable convex relaxations. Factorable programming techniques are used widely in global optimization of mixed-integer nonlinear optimization problems (MINLPs) for bounding general nonconvex functions [20]. These techniques iteratively decompose a factorable function, through the introduction of variables and constraints for intermediate functional expressions, until each intermediate expression can be outer-approximated by a convex feasible set [31]. Current general-purpose MINLP solvers [26, 21, 32] rely on factorable relaxations and as a result, enhancing the quality of such relaxations has a significant impact on our ability to solve a wide range of nonconvex problems.

Factorable reformulations of many types of MINLPs contain a collection of multilinear equations of the form $y_I = \prod_{i \in I} x_i$, $I \in \mathcal{I}$, where $\mathcal{I}$ denotes a family of subsets of $\mathcal{N} = \{1, \ldots, n\}$ of cardinality at least two. Examples include quadratic programs, polynomial programs, and multiplicative programs. Building sharp convex relaxations for multilinears has been the subject of extensive research by the mathematical programming community for over four decades now (cf. [24, 12, 25, 27, 18, 15, 13, 14]) and it is well-understood that the quality of these relaxations has a significant impact on the performance of MINLP solvers [2, 23, 1, 21, 22]. Let us define the nonconvex set associated with all multilinear expressions present in a factorable reformulation of a MINLP as

$$\tilde{\mathcal{S}} = \{(x, y) : y_I = \prod_{i \in I} x_i,\ I \in \mathcal{I},\ x_i \in [0, 1],\ \forall i \in J_1,\ x_i \in \{0, 1\},\ \forall i \in J_2\},$$

*This research was supported in part by National Science Foundation award CMMI-1634768.

†Department of Industrial and Systems Engineering & Wisconsin Institute for Discovery, University of Wisconsin-Madison. E-mail: delpia@wisc.edu.

‡Department of Chemical Engineering, Carnegie Mellon University. E-mail: aida@cmu.edu.

where J_1 and J_2 form a partition of $\mathcal{N}$ and are the index sets corresponding to continuous and binary variables, respectively. It is well-known that the convex hull of the mixed-integer set $\tilde{\mathcal{S}}$ is a polytope and the projection of its vertices onto the space of x variables is given by $\{0,1\}^n$ (cf. [30]). It then follows that the facial structure of the convex hull of $\tilde{\mathcal{S}}$ can be equivalently studied by considering the following binary set:

$$\mathcal{S} = \Big\{(x,y) : y_I = \prod_{i\in I} x_i, I \in \mathcal{I},\ x \in \{0,1\}^n\Big\}. \tag{1}$$

In particular, the set $\mathcal{S}$ represents the feasible region of a linearized unconstrained $0-1$ polynomial optimization problem. Throughout this paper, we refer to the set $\mathcal{S}$ as the *Multilinear set* and refer to its polyhedral convex hull as the *Multilinear polytope* (MP). Moreover, we refer to $r = \max\{|I| : I \in \mathcal{I}\}$ as the degree of the Multilinear set. If all multilinear terms in $\mathcal{S}$ are bilinears, i.e., $r = 2$, the corresponding Multilinear polytope coincides with the Boolean quadric polytope first defined and studied by Padberg [24] in the context of unconstrained $0-1$ quadratic programming. In contrast to the rich literature on structural properties of the Boolean quadric polytope [24, 4, 7], similar polyhedral studies for higher degree Multilinear polytopes are quite scarce [8, 15, 14]. In the special case where $r = n$ and the set $\mathcal{I}$ contains all subsets of $\mathcal{N}$, a complete linear description for the convex hull of the Multilinear set is available [28]. In practice, however, we often have $n \gg r$ and the set $\mathcal{I}$ consists of a small fraction of subsets of $\mathcal{N}$. In this paper, we are particularly interested in such Multilinear sets.

A great simplification in studying the facial structure of the Multilinear polytope is possible when the corresponding Multilinear set is *decomposable* into simpler Multilinear sets. More precisely, let $\mathcal{S}$ be a Multilinear set that can be represented as an intersection of a collection of Multilinear sets $\mathcal{S}_j,\ j \in J$. Clearly, the convex set obtained by intersecting the convex hulls of sets $\mathcal{S}_j$ is a superset of the convex hull of $\mathcal{S}$ as the convexification operation does not, in general, distribute over intersection. It is highly desirable to identify conditions under which we have

$$\mathrm{conv}\Big(\bigcap_{j\in J} \mathcal{S}_j\Big) = \bigcap_{j\in J} \left(\mathrm{conv}\mathcal{S}_j\right),$$

as in such cases characterizing the convex hull of $\mathcal{S}$ simplifies to characterizing the convex hull of each $\mathcal{S}_j$ separately. In this paper, we study decomposability properties of Multilinear sets. To this end, as in [15], we define an equivalent *hypergraph representation* for $\mathcal{S}$. Recall that a hypergraph G is a pair (V, E) where $V = V(G)$ is the set of nodes of G, and $E = E(G)$ is a set of subsets of V of cardinality at least two, called the edges of G (see Berge [6] for an introduction to hypergraphs). The *rank* of G is the maximum cardinality of an edge in E. In the special case where G has no edges, we say that the rank of G is one. Throughout this paper, we consider hypergraphs without loops and parallel edges. With any hypergraph $G = (V, E)$, we associate a Multilinear set $\mathcal{S}_G$ defined as follows:

$$\mathcal{S}_G = \Big\{z \in \{0,1\}^d : z_e = \prod_{v\in e} z_v,\ e \in E\Big\}, \tag{2}$$

where $d = |V| + |E|$. We denote by MP_G the polyhedral convex hull of $\mathcal{S}_G$. Note that the variables z_v, $v \in V$, in (2) correspond to the variables x_i, $i \in \mathcal{N}$, in (1) and the variables z_e, $e \in E$, in (2) correspond to the variables y_I, $I \in \mathcal{I}$, in (1). For quadratic sets, our hypergraph representation simplifies to the *graph representation* defined by Padberg [24] to study the Boolean quadric polytope QP_G. Given a hypergraph $G = (V, E)$, and a subset V' of V, the *section hypergraph* of G induced by V' is the hypergraph $G' = (V', E')$, where $E' = \{e \in E : e \subseteq V'\}$. Given hypergraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we denote by $G_1 \cap G_2$ the hypergraph $(V_1 \cap V_2, E_1 \cap E_2)$, and we denote by $G_1 \cup G_2$, the hypergraph $(V_1 \cup V_2, E_1 \cup E_2)$. Now, consider a hypergraph G, and let G_j, $j \in J$, be distinct section hypergraphs of G such that $\cup_{j\in J} G_j = G$. We say that the set $\mathcal{S}_G$ is *decomposable into sets* $\mathcal{S}_{G_j}$, *for* $j \in J$, if the following relation holds

$$\mathrm{conv}\mathcal{S}_G = \bigcap_{j\in J} \mathrm{conv}\bar{\mathcal{S}}_{G_j}, \tag{3}$$

where, $\bar{\mathcal{S}}_{G_j}$, $j \in J$ is the set of all points in the space of $\mathcal{S}_G$ whose projection in the space defined by G_j is $\mathcal{S}_{G_j}$. If the hypergraph G is not connected, then it is simple to show that $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{G_j}$,

$j \in J$, where G_j, $j \in J$, are the connected components of G. Henceforth, we assume that G is a connected hypergraph.

There has been an interesting stream of research on developing decomposition techniques for NP-hard combinatorial optimization problems. In the following, we briefly review some of the major findings in this area. Chvátal [9] proves that the stable set polytope decomposes into simpler polytopes if the underlying graph contains clique separators. Barahona [3] gives an explicit description for the cut polytope when the underlying graph is not contractible to K_5 by showing that the cut polytope decomposes into simpler polytopes when the intersection graph is a clique of cardinality less than or equal to three. In [5], the authors develop decomposition schemes for the balanced induced subgraph polytope, stable set polytope and the acyclic subgraph polytope, when the corresponding graphs contain one- or two-node cut sets. In his doctoral dissertation, Margot [19] studies general decomposition techniques for combinatorial optimization polytopes and shows that the so called "projected faces property" is sufficient for decomposability of a polytope. Subsequently, in [10], the authors provide a characterization of the projected faces property.

In Section 2, we establish *necessary and sufficient* conditions for decomposability of $\mathcal{S}_G$ into $\mathcal{S}_{G_j}$, $j \in J$, based on the structure of pair-wise intersection hypergraphs $G_j \cap G_{j'}$, for $j, j' \in J$ with $j \neq j'$. Subsequently, in Section 3, we study a more general decomposition technique which allows decomposition of many more types of Multilinear sets associated with sparser hypergraphs. In particular, as we detail in Sections 2 and 3, our characterizations unify and extend the existing decomposability results for the Boolean quadric polytope [24].

It is well-understood that branch-and-cut based MINLP solvers would highly benefit from our decomposition results as such techniques lead to significant reductions in CPU time during cut generation [2, 23, 1, 22]. Our results in Sections 2 and 3 provide easily verifiable conditions under which a Multilinear set can be decomposed into lower dimensional Multilinear sets *without* compromising the quality of the resulting relaxation. In Section 4, we consider the problem of decomposing a Multilinear set $\mathcal{S}_G$ into a collection of non-decomposable Multilinear sets $\mathcal{S}_{G_k}$, $k \in \mathcal{K}$. This problem is related to the problem of decomposing a graph by clique separators, a common tool used for various graph problems such as graph coloring and finding maximum independent sets [29, 17]. We present a polynomial-time algorithm to decompose a Multilinear set in terms of its hypergraph representation. Given a connected rank-r hypergraph $G = (V, E)$, we prove that our proposed algorithm gives an *optimal decomposition* of $\mathcal{S}_G$ in $O(r|E|(|V| + |E|))$ time. We demonstrate that the proposed algorithm performs significantly better than alternative decompositions obtained by a naive application of our decomposition results.

2 Decomposability of Multilinear Sets

In this section, we study decomposability properties of the Multilinear set $\mathcal{S}_G$. Suppose that G_1 and G_2 are distinct section hypergraphs of G such that $G_1 \cup G_2 = G$. We present a necessary and sufficient condition for decomposability of $\mathcal{S}_G$ into Multilinear sets $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$, based on the structure of the intersection hypergraph $G_1 \cap G_2$. In the remainder of this paper, we say that a hypergraph $G = (V, E)$ is *complete* if all subsets of V of cardinality at least two are present in E.

2.1 A sufficient condition for decomposability of Multilinear Sets

The following theorem provides a sufficient condition for decomposability of $\mathcal{S}_G$ into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$.

Theorem 1. *Let G be a hypergraph, and let G_1,G_2 be section hypergraphs of G such that $G_1 \cup G_2 = G$ and $G_1 \cap G_2$ is a complete hypergraph. Then the set $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$.*

Proof. Clearly the inclusion "$\subseteq$" in (3) holds since $\mathcal{S}_G \subseteq \bar{\mathcal{S}}_{G_1} \cap \bar{\mathcal{S}}_{G_2}$. Thus, it suffices to show the reverse inclusion. As G_1 and G_2 are different from G, both $G \setminus G_1$ and $G \setminus G_2$ are nonempty. Let $\tilde{z} \in \text{conv}\bar{\mathcal{S}}_{G_1} \cap \text{conv}\bar{\mathcal{S}}_{G_2}$ and let z^K contain those components of $\tilde{z}$ corresponding to nodes and edges of the complete hypergraph $K = G_1 \cap G_2$. For $i = 1, 2$, let z^i be the vector containing those components of $\tilde{z}$ corresponding to nodes and edges in $G_i \setminus K$. Using these definitions, we can now write, up to reordering variables, $\tilde{z} = (z^1, z^K, z^2)$. Let $k := |V(K)|$. Since K is a complete hypergraph, the set $\mathcal{S}_K$ has dimension $\sum_{i=1}^{k} \binom{k}{i}$ and consists of $\sum_{i=0}^{k} \binom{k}{i}$ affinely independent points, implying that $\text{conv}\mathcal{S}_K$ is a simplex. It then follows

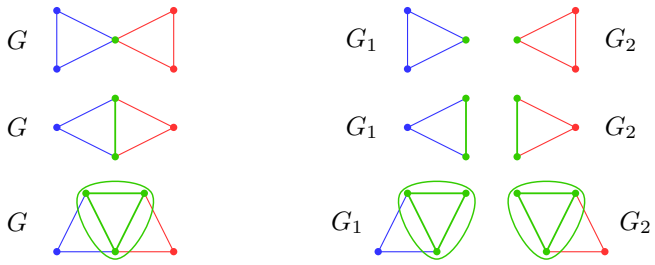

Figure 1: Some examples of hypergraphs G for which by Theorem 1, the set $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$.

that the vector $z^K \in \text{conv}\mathcal{S}_K$ can be written in a unique way as a convex combination of points in $\mathcal{S}_K$; i.e., there exists a unique vector of multipliers λ with $\lambda \geq 0$ and $\sum_{s\in\mathcal{S}_K} \lambda_s = 1$ such that $z^K = \sum_{s\in\mathcal{S}_K} \lambda_s s$.

The vector $(z^1, z^K) \in \text{conv}\mathcal{S}_{G_1}$ can be written as a convex combination of points in $\mathcal{S}_{G_1}$:

$$\begin{aligned}(z^1, z^K) &= \sum_{(r,s)\in\mathcal{S}_{G_1}} \mu'_{r,s}(r,s) \qquad \text{for } \mu' \geq 0 \text{ with } \sum_{(r,s)\in\mathcal{S}_{G_1}} \mu'_{r,s} = 1\\ &= \sum_{s\in\mathcal{S}_K} \sum_{r:(r,s)\in\mathcal{S}_{G_1}} \mu'_{r,s}(r,s)\\ &= \sum_{s\in\mathcal{S}_K} \Big(\sum_{r:(r,s)\in\mathcal{S}_{G_1}} \mu'_{r,s} \Big) \sum_{r:(r,s)\in\mathcal{S}_{G_1}} \mu_{r,s}(r,s),\end{aligned}$$

where in the last equation we introduced new multipliers $\mu_{r,s} := \mu'_{r,s} / \sum_{r':(r',s)\in\mathcal{S}_{G_1}} \mu'_{r',s}$. Clearly $\mu \geq 0$ and $\sum_{r:(r,s)\in\mathcal{S}_{G_1}} \mu_{r,s} = 1$ for every $s \in \mathcal{S}_K$. Since z^K can be written in a unique way as a convex combination of points in $\mathcal{S}_K$, we obtain $\sum_{r:(r,s)\in\mathcal{S}_{G_1}} \mu'_{r,s} = \lambda_s$. Hence, $(z^1, z^K) = \sum_{s\in\mathcal{S}_K} \lambda_s \sum_{r:(r,s)\in\mathcal{S}_{G_1}} \mu_{r,s}(r,s)$. Symmetrically, we obtain $(z^K, z^2) = \sum_{s\in\mathcal{S}_K} \lambda_s \sum_{t:(s,t)\in\mathcal{S}_{G_2}} \nu_{s,t}(s,t)$, for $\nu \geq 0$ with $\sum_{t:(s,t)\in\mathcal{S}_{G_2}} \nu_{s,t} = 1$. Hence, the following holds:

$$(z^1, z^K, z^2) = \sum_{s\in\mathcal{S}_K,\, r:(r,s)\in\mathcal{S}_{G_1},\, t:(s,t)\in\mathcal{S}_{G_2}} \lambda_s \mu_{r,s} \nu_{s,t}(r,s,t). \tag{4}$$

Since G_1 and G_2 are section hypergraphs of G, each 0−1 vector (r,s,t) in (4) is in $\mathcal{S}_G$. Moreover, the multipliers $\lambda_s \mu_{r,s} \nu_{s,t}$ are nonnegative and satisfy

$$\sum_{s\in\mathcal{S}_K,\, r:(r,s)\in\mathcal{S}_{G_1},\, t:(s,t)\in\mathcal{S}_{G_2}} \lambda_s \mu_{r,s} \nu_{s,t} = 1,$$

This implies $\tilde{z} \in \text{conv}\mathcal{S}_G$. □

Figure 1 illustrates some simple hypergraphs G for which the Multilinear set $\mathcal{S}_G$ is decomposable into two Multilinear sets $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$. To draw a hypergraph G, throughout this paper, we represent the nodes in $V(G)$ by points, and the edges in $E(G)$ by closed curves enclosing the corresponding set of points.

We should remark that Theorem 1 can be alternatively proved using the "projected faces property" introduced by Margot [19], as the Multilinear polytope $\text{MP}_{G_1\cap G_2}$ is a simplex. However, for simplicity of presentation and keeping the text self-contained we chose to provide a proof that solely relies on elementary arguments.

Theorem 1 unifies the existing decomposability results for the Boolean quadric polytope QP_G (cf. [3, 24]):

Corollary 1. *Consider a graph $G = G_1 \cup G_2$, where G_1 and G_2 are induced subgraphs of G with $V(G_1) \cap V(G_2) = \{u\}$, for some $u \in V(G)$, or $V(G_1) \cap V(G_2) = \{u, v\}$, for some $\{u, v\} \in E(G)$. Then QP_G is decomposable into QP_{G_1} and QP_{G_2}.*

By a recursive application of Theorem 1, we obtain a sufficient condition for decomposability of $\mathcal{S}_G$ into sets $\mathcal{S}_{G_j}$, for $j \in J$.

Theorem 2. *Let G be a hypergraph, and let G_j, $j \in J$, be section hypergraphs of G such that $\cup_{j\in J} G_j = G$. Suppose that for all $j, j' \in J$ with $j \neq j'$, the intersection $G_j \cap G_{j'}$ is the same complete hypergraph $\bar{G}$. Then $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{G_j}$, for $j \in J$.*

2.2 A necessary condition for decomposability of Multilinear Sets

In this section, we demonstrate the tightness of Theorem 1; namely, we show that if a rank-r hypergraph $\bar{G}$ is not complete, then for any integer $r' \geq \max\{r, 2\}$, there always exists a rank-r' hypergraph $G = G_1 \cup G_2$ with $\bar{G} = G_1 \cap G_2$ such that $\mathcal{S}_G$ is *not* decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$. We will make use of the following two lemmata to prove this claim. In the remainder of the paper, for notational simplicity, given a node $v.$, we sometimes write $z.$ instead of $z_{v.}$. Similarly, given an edge $e.$, we sometimes write $z.$ instead of $z_{e.}$.

Lemma 1. *Consider the hypergraph $G = (V, E)$ with $V = \{v_i : i = 1, \ldots, r+3\}$ for some $r \geq 2$. Let $J = \{3, \ldots, r+3\}$ and let E contain the following set of edges: $e_{1i} = \{v_1, v_i\}$ for all $i \in J$, $e_{2i} = \{v_2, v_i\}$ for all $i \in J$, and $e_I = \{v_i : i \in I\}$ for every $I \subset J$ of cardinality between 2 and r. Then the inequality given by*

$$r(r-1)z_1 + (r-1)z_2 + r\sum_{i\in J} z_i - (r-1)\sum_{i\in J} z_{1i} - \sum_{i\in J} z_{2i} - \sum_{\substack{I\subset\{3,\ldots,r+3\}\\ |I|=r}} z_I \leq r^2 - 1 \tag{5}$$

is facet-defining for MP_G.

Proof. The validity of inequality (5) for MP_G can be verified by considering the four cases corresponding to different combinations of $(z_1, z_2) \in \{0, 1\}^2$. We now show that inequality (5) defines a facet of MP_G. To do so, we provide three families of points in MP_G that satisfy the inequality (5) tightly, and show that the hyperplane $az = \alpha$ (unique up to a scaling) passing through all such points is the supporting hyperplane corresponding to the half-space implied by (5).

(i) Let $\mathcal{J}$ denote the set of all subsets of J with cardinality between 0 and $r-1$. For each $I \in \mathcal{J}$, construct a point with $z_1 = z_2 = 1$, $z_i = 1$ for all $i \in I$, and $z_i = 0$ otherwise. The variables z_e, for $e \in E$, are computed accordingly. It is simple to verify that all such points satisfy inequality (5) tightly. Substituting such a tight point with $I = \emptyset$ in $az = \alpha$, we obtain

$$a_1 + a_2 = \alpha. \tag{6}$$

Similarly, setting $I = \{i\}$ for all $i \in J$, yields $a_1 + a_2 + a_i + a_{1i} + a_{2i} = \alpha$. From (6), it follows that

$$a_i + a_{1i} + a_{2i} = 0 \tag{7}$$

for all $i \in J$. Moreover, letting $I = \{i, j\}$ for $3 \leq i < j \leq r+3$, yields $a_1+a_2+a_i+a_{1i}+a_{2i}+a_j+a_{1j}+a_{2j}+a_{ij} = \alpha$. Since $a_1 + a_2 = \alpha$, $a_i + a_{1i} + a_{2i} = 0$, and $a_j + a_{1j} + a_{2j} = 0$, we conclude that $a_{ij} = 0$. Utilizing a similar argument in a recursive manner for subsets I with larger cardinalities, we obtain:

$$a_I = 0, \quad \forall I \subset J,\ |I| \geq 2. \tag{8}$$

(ii) Let $\mathcal{K}$ denote the set of all subsets of J of cardinality $r-1$ or r. For each $I \in \mathcal{K}$, construct a point with $z_1 = 1$, $z_2 = 0$, $z_i = 1$ for all $i \in I$, and $z_i = 0$ otherwise. The variables z_e, for $e \in E$, are computed accordingly. All these points satisfy inequality (5) tightly. First, consider the points with $|I| = r$. By (8), we have $a_{I'} = 0$ for all $I' \subset I$ with $|I'| \geq 2$. Thus, substituting for such points in $az = \alpha$, we obtain

$$a_1 + \sum_{i\in I} a_i + \sum_{i\in I} a_{1i} + a_I = \alpha. \tag{9}$$

Now, consider the set of tight points corresponding to subsets of J of cardinality $r-1$. Suppose that for each $j \in I$, where I is defined in (9), the new tight point is obtained by letting $z_i = 1$ for all $i \in I \setminus \{j\}$ and $z_j = 0$. Substituting this point in $az = \alpha$ yields $a_1 + \sum_{i\in I\setminus\{j\}} a_i + \sum_{i\in I\setminus\{j\}} a_{1i} = \alpha$. We now add these r equations for every $j \in J$ and subtract the result from equation (9) multiplied by $r-1$ to obtain $a_1 - (r-1)a_I = \alpha$. Since, this relation holds for all $I \in \mathcal{K}$ with $|I| = r$, we conclude that

$$a_I = \lambda := (a_1 - \alpha)/(r-1), \quad \forall I \subset J,\ |I| = r. \tag{10}$$

(iii) Let $\mathcal{L}$ denote the set of all subsets of J of cardinality r or $r+1$. For each $I \in \mathcal{L}$, construct a tight point with $z_1 = z_2 = 0$, $z_i = 1$ for all $i \in I$, and $z_i = 0$ otherwise. The variables z_e, for $e \in E$, are computed accordingly. Substituting the point with $I = J$ in $az = \alpha$ yields

$$\sum_{i \in J} a_i + (r+1)\lambda = \alpha. \tag{11}$$

In addition, for each $j \in J$, substituting the tight point with $I = J \setminus \{j\}$, we obtain

$$\sum_{i \in J \setminus \{j\}} a_i + \lambda = \alpha. \tag{12}$$

Subtracting the two equations gives $a_i + r\lambda = 0$ for all $i \in J$. Hence,

$$a_i = \mu := -r\lambda, \quad \forall i \in J. \tag{13}$$

Combining equations (10), (13) and (9), we obtain $a_1 + r\mu + \sum_{i \in I} a_{1i} + \lambda = \alpha$ for all $I \subset J$ with $|I| = r$. Now consider two equations from this system, one with $I = J \setminus \{j\}$, and another with $I = J \setminus \{k\}$ for some $j, k \in J$ such that $j \neq k$. Subtracting these two equations we obtain $a_{1j} = a_{1k}$. By applying this argument recursively, it follows that

$$a_{1i} = \nu_1 := (\alpha - a_1 - r\mu - \lambda)/r, \quad \forall i \in J. \tag{14}$$

In addition, equation (7) simplifies to $\mu + \nu_1 + a_{2i} = 0$, implying

$$a_{2i} = \nu_2 := -\mu - \nu_1, \quad \forall i \in J. \tag{15}$$

To summarize, using equations (6), (7), (8), (10), (11), (12), (13), (14), (15), we obtain the following system of equations

$$a_1 + a_2 = \alpha \tag{16}$$
$$\mu + \nu_1 + \nu_2 = 0 \tag{17}$$
$$a_1 + (r-1)\mu + (r-1)\nu_1 = \alpha \tag{18}$$
$$a_1 + r\mu + r\nu_1 + \lambda = \alpha \tag{19}$$
$$(r+1)\mu + (r+1)\lambda = \alpha \tag{20}$$
$$r\mu + \lambda = \alpha. \tag{21}$$

If $\alpha = 0$, then it can be checked that the only solution of the above system is the zero vector. Thus, without loss of generality, we assume $\alpha = r^2 - 1$. Equations (20)-(21) imply $\mu = r$ and $\lambda = -1$. It follows that equations (18)-(19) can be written as $a_1 + (r-1)\nu_1 = r - 1$ and $a_1 + r\nu_1 = 0$, implying $a_1 = r(r-1)$ and $\nu_1 = -(r-1)$. Finally, from (16) we obtain $a_2 = r - 1$, and (17) yields $\nu_2 = -1$. Therefore, inequality (5) defines a facet of MP_G. □

Lemma 2. *Let G be a hypergraph, and let G_1, G_2 be section hypergraphs of G such that $G_1 \cup G_2 = G$ and the set $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$. Let H be a section hypergraph of G such that $V(H) \setminus V(G_1)$ and $V(H) \setminus V(G_2)$ are both nonempty. For $j = 1, 2$, let H_j be the section hypergraph of G induced by $V(G_j) \cap V(H)$. Then the set $\mathcal{S}_H$ is decomposable into $\mathcal{S}_{H_1}$ and $\mathcal{S}_{H_2}$.*

Proof. As in the proof of Theorem 1, we define a vector $(z^1, \bar{z}, z^2)$ such that $(z^1, \bar{z}) \in \mathrm{conv}\mathcal{S}_{H_1}$ and $(\bar{z}, z^2) \in \mathrm{conv}\mathcal{S}_{H_2}$. We show that $(z^1, \bar{z}, z^2) \in \mathrm{conv}\mathcal{S}_H$.

Let $\bar{s}$ be obtained from $\bar{z}$ by adding zero coefficients to the components corresponding to nodes and edges that are in $\bar{G} := G_1 \cap G_2$ but not in $\bar{H} := H_1 \cap H_2$. For $j = 1, 2$, let s^j be obtained from z^j by adding zero coefficients to the components corresponding to nodes and edges that are in G_j but not in H_j or in $\bar{G}$.

Let $p \in \mathcal{S}_{H_j}$, and let $\tilde{p}$ be obtained from p by adding zero coefficients to the components that are in G_j but not in H_j. Since H_j is a section hypergraph of G_j, it follows that the vector $\tilde{p}$ is in $\mathcal{S}_{G_j}$. Consequently, we have $(z^1, \bar{z}) \in \mathrm{conv}\mathcal{S}_{H_1}$ and $(\bar{z}, z^2) \in \mathrm{conv}\mathcal{S}_{H_2}$, which in turn imply $(s^1, \bar{s}) \in \mathrm{conv}\mathcal{S}_{G_1}$ and $(\bar{s}, s^2) \in \mathrm{conv}\mathcal{S}_{G_2}$. By decomposability of $\mathcal{S}_G$ into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$, it follows that $(s^1, \bar{s}, s^2) \in \mathrm{conv}\mathcal{S}_G$. Therefore $(s^1, \bar{s}, s^2)$ can be written as a convex combination of points in $\mathcal{S}_G$. By dropping from each point in such a convex combination the components corresponding to nodes and edges present in G but not in H, we obtain $(z^1, \bar{z}, z^2) \in \mathrm{conv}\mathcal{S}_H$. □

We are now in position to prove the converse of Theorem 1.

Theorem 3. *Let $\bar{G}$ be a rank-r hypergraph that is not complete. Then for any integer $r' \geq \max\{r, 2\}$, there exists a rank-r' hypergraph $G = G_1 \cup G_2$, where G_1 and G_2 are section hypergraphs of G with $\bar{G} = G_1 \cap G_2$, such that the set $\mathcal{S}_G$ is not decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$.*

Proof. We start by proving that if the rank-r hypergraph $\bar{G}$ is not complete, then there exists a hypergraph G of rank $\max\{r, 2\}$ such that $G = G_1 \cup G_2$, where G_1 and G_2 are section hypergraphs of G with $\bar{G} = G_1 \cap G_2$, and the set $\mathcal{S}_G$ is not decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$. Subsequently we show that the same statement holds for a rank-r' hypergraph G with $r' > \max\{r, 2\}$.

We search for a section hypergraph of $\bar{G}$, denoted by $\bar{H}$, that has q nodes, for some $q \in \{2, \ldots, |V(\bar{G})|\}$, such that $\bar{H}$ contains all edges of cardinality between two and $q-1$ but does not contain the edge of cardinality q. Observe that since $\bar{G}$ is not complete, it always contains such a section hypergraph. We first show that there exists a hypergraph $H = H_1 \cup H_2$, where H_1 and H_2 are section hypergraphs of H with $\bar{H} = H_1 \cap H_2$, such that the set $\mathcal{S}_H$ is not decomposable into $\mathcal{S}_{H_1}$ and $\mathcal{S}_{H_2}$. Subsequently, we employ the result of Lemma 2 to complete the proof. Let v_1 and v_2 be new nodes (not in $V(\bar{G})$), let $V(H_1) = V(\bar{H}) \cup \{v_1\}$, $V(H_2) = V(\bar{H}) \cup \{v_2\}$, $E(H_1) = E(\bar{H}) \cup \{\{v_1, v\} : v \in V(\bar{H})\}$, and $E(H_2) = E(\bar{H}) \cup \{\{v_2, v\} : v \in V(\bar{H})\}$. By construction $\bar{H} = H_1 \cap H_2$. We now identify a facet defining inequality for MP_H with nonzero coefficients corresponding to edges in both $E(H_1) \setminus E(\bar{H})$ and $E(H_2) \setminus E(\bar{H})$. Two cases arise:

(i) $q = 2$. In this case, there exist nodes $u, w \in V(\bar{G})$ such that $\{u, w\} \notin E(\bar{G})$. Therefore, the graph H is a cordless cycle with the node set given by $\{v_1, v_2, u, w\}$ and the edge set given by $\{\{v_1, u\}, \{v_1, w\}, \{v_2, u\}, \{v_2, w\}\}$. It is well-known that the inequality

$$-z_{v_2} - z_w - z_{\{v_1,u\}} + z_{\{v_1,w\}} + z_{\{v_2,u\}} + z_{\{v_2,w\}} \leq 0,$$

defines a facet of $\mathrm{conv}\mathcal{S}_H$ (cf. [24]). Since $\{v_1, u\} \in E(H_1) \setminus E(\bar{H})$, while $\{v_2, u\} \in E(H_2) \setminus E(\bar{H})$, it follows that the set $\mathcal{S}_H$ is not decomposable to $\mathcal{S}_{H_1}$ and $\mathcal{S}_{H_2}$.

(ii) $q > 2$. In this case, the hypergraph H is of the form considered in the statement of Lemma 1 (with $r = q - 1$) and hence, the inequality given by

$$\begin{aligned}(q-1)(q-2)z_{v_1} + (q-2)z_{v_2} + (q-1)\sum_{v \in V(\bar{H})} z_v + \\ -(q-2)\sum_{v \in V(\bar{H})} z_{\{v_1,v\}} - \sum_{v \in V(\bar{H})} z_{\{v_2,v\}} - \sum_{\substack{e \in E(\bar{H}) \\ |e| = q-1}} z_e \leq (q-1)^2 - 1\end{aligned}$$

is facet defining for $\mathrm{conv}\mathcal{S}_H$. Thus, $\mathcal{S}_H$ is not decomposable into $\mathcal{S}_{H_1}$ and $\mathcal{S}_{H_2}$.

Let $G := H_1 \cup H_2 \cup \bar{G}$. Clearly, the rank of G equals $\max\{r, 2\}$. Define G_1 and G_2 to be section hypergraphs of G induced by $V(H_1) \cup V(\bar{G})$ and $V(H_2) \cup V(\bar{G})$, respectively. It is simple to check that $G = G_1 \cup G_2$ and $\bar{G} = G_1 \cap G_2$. Since H is a section hypergraph of G, $V(H) \setminus V(G_1) = \{v_2\}$, and $V(H) \setminus V(G_2) = \{v_1\}$, by Lemma 2, the set $\mathcal{S}_G$ is not decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$.

Now let r' be an integer greater than $\max\{r, 2\}$, and let the hypergraph $\tilde{G}_2$ be obtained from G_2 by adding $r' - 1$ new nodes, denoted by W, and a new edge $\{v_2\} \cup W$. Now define the rank-r' hypergraph $\tilde{G} := G_1 \cup \tilde{G}_2$. Then it is simple to check that by Lemma 2 the set $\mathcal{S}_{\tilde{G}}$ is not decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{\tilde{G}_2}$, and this completes the proof. □

Figure 2 illustrates some simple hypergraphs G for which the set $\mathcal{S}_G$ is *not* decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$.

In [24], Padberg poses a question regarding the decomposability of the Boolean quadric polytope when the intersection graph is a clique of cardinality greater than two. In our context, his question can be equivalently stated as follows: let $\bar{G}$ be a graph with $|V(\bar{G})| \geq 3$ and with $E(\bar{G})$ containing all subsets of $V(\bar{G})$ of cardinality two. Given any two distinct graphs G_1 and G_2 with $\bar{G} = G_1 \cap G_2$, $V(G_1) \setminus V(\bar{G}) \neq \emptyset$, and $V(G_2) \setminus V(\bar{G}) \neq \emptyset$, is $\mathrm{QP}_{G_1 \cup G_2}$ always decomposable into QP_{G_1} and QP_{G_2}? The proof of Theorem 3 implies that the answer to this question is *negative* for every $\bar{G}$ with three or more nodes.

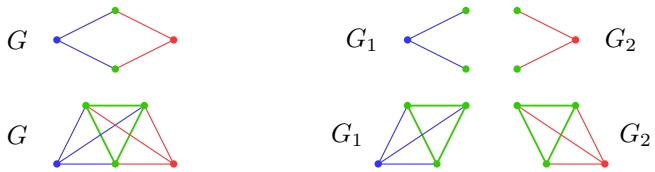

Figure 2: Some examples of hypergraphs G for which the set $\mathcal{S}_G$ is *not* decomposable into $\mathcal{S}_{G_1}$ and $\mathcal{S}_{G_2}$.

3 Decomposability of Multilinear sets with sparse intersection hypergraphs

The decomposability results given in Section 2 are based upon the assumption that the pair-wise intersection hypergraphs are complete. In this section, we explore the relation between the sparsity of the intersection hypergraph $\bar{G}$ and the extent to which G is decomposable. To this end, given a hypergraph $\bar{G}$, we define its *incompleteness number* $\kappa(\bar{G})$ to be the difference between the number of edges of a complete hypergraph on $V(\bar{G})$ and the number of edges of $\bar{G}$; that is, $\kappa(\bar{G}) = 2^{|V(\bar{G})|} - |V(\bar{G})| - |E(\bar{G})| - 1$. The following theorem provides a decomposition scheme for Multilinear sets whose corresponding intersection hypergraph $\bar{G}$ is not complete; i.e., $\kappa(\bar{G}) > 0$.

Theorem 4. *Let G be a hypergraph, and let G_j, $j \in J$, be section hypergraphs of G such that $\cup_{j\in J} G_j = G$. Suppose that, for all $j, j' \in J$ with $j \neq j'$, the intersection $G_j \cap G_{j'}$ is the same hypergraph $\bar{G}$. Denote by $\bar{\mathcal{K}}$ the set of all subsets of J of cardinality $2^{\kappa(\bar{G})}$. Let $\tilde{G}_{\mathcal{K}} = \cup_{j\in\mathcal{K}} G_j$ for each $\mathcal{K} \in \bar{\mathcal{K}}$. Then $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{\tilde{G}_{\mathcal{K}}}$, $\mathcal{K} \in \bar{\mathcal{K}}$.*

Proof. Let $\kappa = \kappa(\bar{G})$. If $\kappa = 0$, the result follows from Theorem 2. Henceforth, we assume that $\kappa \geq 1$. Let G' be the hypergraph obtained from G by adding the κ edges corresponding to all subsets of $V(\bar{G})$ that are not present in $E(G)$. Moreover, we denote by G'_j the section hypergraph of G' induced by $V(G_j)$. By Theorem 2, the Multilinear set $\mathcal{S}_{G'}$ is decomposable into $\mathcal{S}_{G'_j}$, for $j \in J$. Therefore, the support hypergraph of each facet-defining inequality of $\mathrm{MP}_{G'}$ is contained in some hypergraph G'_j, for $j \in J$.

To obtain a facet-description of MP_G from that of $\mathrm{MP}_{G'}$, we project out, via Fourier elimination, the κ variables corresponding to edges that we have artificially added to G in order to obtain G'. By projecting out one variable, the support hypergraph of each new inequality is contained in the union of at most two hypergraphs G'_j, $j \in J$. Similarly, by projecting out the next variable, the support hypergraph of each new inequality can be contained in the union of at most four hypergraphs G'_j, $j \in J$. In this way, once we project out all κ variables, the support hypergraph of each inequality is contained in the union of at most 2^κ hypergraphs G'_j, $j \in J$. Hence, the theorem follows. □

In Theorem 4, letting $\kappa(\bar{G}) = 0$, yields Theorem 2 (and Theorem 1, if $|J| = 2$). Moreover, letting $\kappa(\bar{G}) = 1$, we obtain the following:

Corollary 2. *Let G be a hypergraph, and let G_j, $j \in J$, be section hypergraphs of G such that $\cup_{j\in J} G_j = G$. Suppose that for all $j, j' \in J$ with $j \neq j'$, the intersection $G_j \cap G_{j'}$ is the same hypergraph $\bar{G}$, and that $\bar{G}$ can be obtained by removing one edge from the complete hypergraph on $V(\bar{G})$. Then $\mathcal{S}_G$ is decomposable into the sets $\mathcal{S}_{G_j \cup G_{j'}}$, for $j, j' \in J$ with $j \neq j'$.*

We demonstrate the applicability of Corollary 2 with an example.

Example 1. *Consider the hypergraph G with $V(G) = \{v_1, v_2, v_3, v_4, v_5, v_6\}$ and $E(G) = \{e_{12}, e_{13}, e_{23}, e_{1234}, e_{1235}, e_{1236}\}$, where edge e_I contains the nodes with indices in I. Let G_1, G_2, and G_3 be section hypergraphs of G induced by the subsets of nodes $\{v_1, v_2, v_3, v_4\}$, $\{v_1, v_2, v_3, v_5\}$, and $\{v_1, v_2, v_3, v_6\}$, respectively (see Figure 3). In this case, $\mathcal{S}_G$ is not decomposable into $\mathcal{S}_{G_1}$, $\mathcal{S}_{G_2}$, and $\mathcal{S}_{G_3}$, as for example the inequality*

$$z_4 - z_{1234} + z_{1235} \leq 1$$

defines a facet of $\mathcal{S}_G$. However, the pair-wise intersection hypergraph; i.e., $\bar{G} := G_1 \cap G_2 = G_2 \cap G_3 = G_3 \cap G_1$ can be obtained by removing one edge (in this case e_{123}) from the complete hypergraph on v_1, v_2, v_3. Therefore, by Corollary 2, the Multilinear set $\mathcal{S}_G$ is decomposable into subsets $\mathcal{S}_{G_1 \cup G_2}$, $\mathcal{S}_{G_2 \cup G_3}$, and $\mathcal{S}_{G_3 \cup G_1}$. ◇

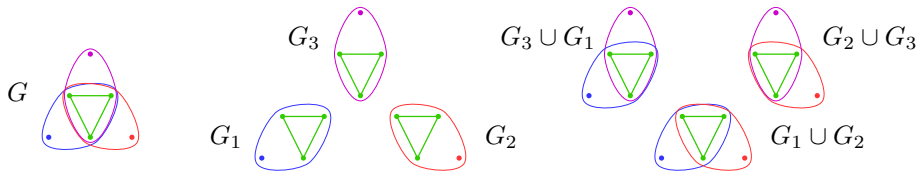

Figure 3: The Multilinear set $\mathcal{S}_G$ is not decomposable into the sets $\mathcal{S}_{G_1}$, $\mathcal{S}_{G_2}$, and $\mathcal{S}_{G_3}$. However, by Corollary 2, it is decomposable into $\mathcal{S}_{G_1 \cup G_2}$, $\mathcal{S}_{G_2 \cup G_3}$, and $\mathcal{S}_{G_3 \cup G_1}$.

As a direct consequence of Corollary 2, we now present *new* sufficient conditions under which the Boolean quadric polytope is decomposable into simpler sets.

Corollary 3. *Let G be a graph, and let G_j, $j \in J$, be induced subgraphs of G such that $\cup_{j \in J} G_j = G$. Suppose that for all $j, j' \in J$ with $j \neq j'$, the intersection $G_j \cap G_{j'}$ is the same graph $\bar{G}$, and that $\bar{G}$ has one of the following forms:*

(i) $\bar{G}$ consists of two isolated nodes, i.e., $V(\bar{G}) = \{v, w\}$, and $E(\bar{G}) = \emptyset$.

(ii) $\bar{G}$ consists of a triangle, i.e., $V(\bar{G}) = \{u, v, w\}$, and $E(\bar{G}) = \{\{u, v\}, \{v, w\}, \{w, u\}\}$.

Then QP_G is decomposable into $QP_{G_j \cup G_{j'}}$ for all $j, j' \in J$ with $j \neq j'$.

Proof. Follows directly from Corollary 2 by using the fact that the intersection graph $\bar{G}$ can be obtained by (i) removing the edge $\{v, w\}$ from the complete graph on v, w and (ii) removing the edge $\{u, v, w\}$ from the complete hypergraph on u, v, w. □

Example 2. *Consider the graph G with $V(G) = \{v_i : i = 1, \ldots, r+2\}$ for some $r \geq 3$. Suppose that $E(G)$ consist of the following set of edges: $\{v_1, v_j\}$ and $\{v_2, v_j\}$ for all $j \in J = \{3, \ldots, r+2\}$. Denote by G_j, $j \in J$, the subgraph of G induced by the nodes v_1, v_2, v_j. It is then simple to check that the pair-wise intersection graph $\bar{G}$ consists of two nodes v_1, v_2. Therefore, by Part (i) of Corollary 3, $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{G_j \cup G_{j'}}$, for all $j, j' \in J$ with $j \neq j'$. Now, consider one set $\mathcal{S}_{G_j \cup G_{j'}}$. The graph $G_j \cup G_{j'}$ consists of a cordless cycle of length four. It then follows that $MP_{G_j \cup G_{j'}}$ is obtained by adding the odd-cycle inequalities to the standard linearization of $\mathcal{S}_{G_j \cup G_{j'}}$ (cf. [24]). Thus, the Boolean quadric polytope associated with the graph G is obtained by adding all odd-cycle inequalities corresponding to $r(r-1)/2$ chordless cycles of length four to the standard linearization of $\mathcal{S}_G$.* ◇

4 A polynomial-time algorithm for decomposition of Multilinear sets

In this section, we present a simple and efficient algorithm for *optimally* decomposing Multilinear sets into simpler and non-decomposable Multilinear sets based on our results in Section 2. Our proposed algorithm can be easily incorporated in branch-and-cut based MINLP solvers as a preprocessing step for cut generation. Throughout this section, whenever a Multilinear set $\mathcal{S}_G$ is decomposable into subsets $\mathcal{S}_{G_j}$, $j \in J$, we refer to the family G_j, $j \in J$, as a *decomposition* of G. Without loss of generality, we assume that G is a connected hypergraph; that is, if the hypergraph G is not connected, then it is simple to see that $\mathcal{S}_G$ is decomposable into $\mathcal{S}_{G_k}$, $k \in K$, where G_k, $k \in K$, denote the connected components of G. Thus, in this case, our algorithm can be employed to further decompose each connected component G_k. Now, consider a hypergraph G and let $p \subset V(G)$. Denote by $\bar{G}$ the section hypergraph of G induced by p. We say that p *decomposes* G if the following two conditions are satisfied:

(a) The hypergraph $\bar{G}$ is complete.

(b) There exist section hypergraphs G_j, $j \in J$, of G, with $V(G_j) \setminus V(G_{j'}) \neq \emptyset$ for all $j, j' \in J$ with $j \neq j'$, that together with $\bar{G}$, satisfy the hypothesis of Theorem 2.

In condition (b) defined above, by letting $V(G_j) \setminus V(G_{j'}) = \emptyset$ for some $j, j' \in J$, we obtain $\cup_{i \in J \setminus \{j\}} G_i = G$. Thus, we can apply Theorem 2 to the family G_i, $i \in J \setminus \{j\}$ instead, and obtain a more compact decomposition. Furthermore, for a connected hypergraph G, it can be shown that each hypergraph G_j, $j \in J$, is a connected hypergraph as well. For a fixed p that decomposes G, the decomposition obtained by utilizing Theorem 2 is not unique, in general. That is, there might exist several families of section hypergraphs G_j, $j \in J$, with $\bigcup_{j \in J} G_j = G$, whose pair-wise intersection hypergraph is $\bar{G}$. Clearly, among all such decompositions, we are interested in the ones for which p does not decompose any of the G_j. It can be shown that such a decomposition of G is indeed unique. Henceforth, we refer to this unique decomposition as the *p-decomposition* of G.

In general, a Multilinear set $\mathcal{S}_G$ is decomposable into simpler sets via a series of p-decompositions of G until none of the newly generated Multilinear sets are decomposable. Given a hypergraph G, we define its *full-decomposition* as a decomposition of G given by a family G_k, $k \in K$, with the following two properties:

(i) There exists no G_k, for some $k \in K$, and $p \subset V(G_k)$ such that p decomposes G_k.

(ii) No hypergraph G_s, $s \in K$, in the decomposition is a section hypergraph of another hypergraph G_t, $t \in K$, with $t \neq s$.

We should remark that if G_s is a section hypergraph of G_t for some $s, t \in K$ with $s \neq t$, then MP_{G_s} corresponds to a face of MP_{G_t}. Thus, removing G_s from a decomposition of G, translates into removing redundant inequalities from the facet description of MP_G, which is highly beneficial from a computational point of view.

Finding a full-decomposition of a hypergraph has certain similarities to the problem of decomposing a graph by means of clique separators, where a clique separator is defined as a clique whose removal disconnects the graph. Given a graph G with n nodes and m edges, Tarjan [29] presents an $O(nm)$-time algorithm to decompose the graph. The decomposition obtained using this algorithm depends on the ordering of the clique separators and is far from unique, in general. Leimer [17] introduces a modification of Tarjan's algorithm in which minimal clique separators are utilized in the recursive decomposition of the graph. The author shows that the output of this algorithm is unique and that the number of subgraphs in the decomposition is minimal. In the remainder of this section, we present results on uniqueness and optimality of our proposed decomposition algorithm (see Propositions 3 and 5), which are of similar flavor to the results presented in [29, 17] regarding the decomposition of graphs by clique separators. While it is possible to obtain alternative proofs for our aforementioned results using the machinery developed in [17], to keep the presentation simple, we utilize our techniques to prove these propositions.

In the following, we first show how to obtain the p-decomposition of G algorithmically. Subsequently, we define a general algorithm to obtain a full-decomposition of a hypergraph and detail on possible enhancements.

4.1 An algorithm to obtain the p-decomposition of G

We start by introducing some graph terminology. Throughout this section, we assume that a hypergraph is represented by an incidence-list in which edges are stored as objects, and every edge stores its incident nodes. In order to use efficient searching algorithms, we assume that the vertex list for each edge is sorted. Otherwise, such a sorted data structure for a rank-r hypergraph can be obtained in $O(r|E|)$ time by using some integer sorting algorithm such as counting sort [11]. In addition, we assume that the edges of E are sorted in increasing cardinality, and edges of the same cardinality are sorted lexicographically. For a rank-r hypergraph, such a sorting order can be obtained using the least significant digit (LSD) radix sort in $O(r|E|)$ operations (cf. [11]). For graphs however, we consider a slightly different data structure as it is widely-used for some of the graph algorithms that we utilize in this paper. We represent a graph by an adjacency-list in which nodes are stored as objects, and every nodes stores its adjacent nodes.

Given a rank-r hypergraph $G = (V, E)$, we define a *graph reduction* of G as a graph $G' = (V, E')$ obtained from G by replacing each edge of cardinality at least three with one cycle containing all of its nodes, where no node is repeated. Furthermore, to obtain a simple graph, all parallel edges are removed. A hypergraph can have many different graph reductions, in general. It can be shown that any graph reduction of G has at most $r|E|$ edges and can be obtained in $O(r|E|)$ time: we construct a graph reduction of a hypergraph

G in two steps: (i) starting from the incidence-list of the hypergraph G, we first generate the adjacency-list for the (multi)-graph $\hat{G} = (V, \hat{E})$ obtained by replacing each edge of cardinality at least three in G by a cycle containing all of its nodes, where no node is repeated, (ii) given the adjacency-list representation of the (multi)-graph $\hat{G}$, we compute the adjacency-list representation of the equivalent simple graph $G' = (V, E')$, where E' consists of the edges in $\hat{E}$ with all multiple edges between two nodes replaced by a single edge; note that the adjacency-list of a multi-graph is similar to that of a simple graph except that in a multi-graph the list of adjacent nodes for each node may contain repeated elements. It is simple to check that both of these steps can be performed in $O(r|E|)$ time.

Given a hypergraph $G = (V, E)$ and an edge $\tilde{e} \in E$, the hypergraph $G' = (V', E')$ obtained from G by *contracting* $\tilde{e}$ is defined as $V' = V \setminus \tilde{e} \cup \{\tilde{v}\}$, where $\tilde{v}$ is a new node, and $E' = \{e : e \in E,\ e \cap \tilde{e} = \emptyset\} \cup \{e \setminus \tilde{e} \cup \{\tilde{v}\} : e \in E,\ e \cap \tilde{e} \neq \emptyset\}$. For a rank-$r$ hypergraph G, the hypergraph G' can be constructed in $O(r|E|)$ time. To see this, note that since by assumption the vertex lists corresponding to all edges of G are sorted, for each $e \in E$ we can obtain $e \setminus \tilde{e}$ in $O(\max(|e|, |\tilde{e}|))$ time. It then follows that G' can be obtained in $O(r|E|)$ time. Finally, given a graph $G = (V, E)$, and a node $v \in V$, we denote by $G \setminus v$, the graph obtained from G by removing node v and all edges containing v. In the sequel, for notational simplicity, we sometimes identify a node, say v, with the set containing it, say $\{v\}$. The following proposition provides a simple algorithm for constructing the p-decomposition of a hypergraph G.

Proposition 1. *Given a connected rank-r hypergraph $G = (V, E)$ and $p \subset V$, we can test if p decomposes G, and, if so, obtain the p-decomposition of G in $O(r|E|)$ time.*

Proof. Clearly if p decomposes G, then $p \in V \cup E$. We first check if condition (a) in the definition of p-decomposition is satisfied; that is, if the section hypergraph induced by p is complete. This condition is trivially satisfied if $p \in V$. Thus, assume that $p \in E$. It then suffices to check if each subset of p of cardinality at least two is an edge of G. Clearly, if $2^{|p|} - |p| - 1 > |E|$, then such subsets of p are not all present in E. Therefore, suppose that $2^{|p|} - |p| - 1 \leq |E|$. By assumption, edges of G are sorted in increasing cardinality and edges of the same cardinality are sorted lexicographically. By using a similar ordering for the subsets of p, we can check in $O(|E|)$ time, if the section hypergraph induced by p is complete.

We now assume that the section hypergraph induced by p is complete, and we show how to check if condition (b) holds. Let G' be the hypergraph obtained from G by contracting p, which can be constructed in $O(r|E|)$ time. Let $\tilde{v} \in V(G')$ be the new node added to V after contraction of p in G, and let G'' be a graph reduction of G', which has at most $r|E|$ edges and can be obtained in $O(r|E|)$ time. It is then easy to see that p decomposes G if and only if $G'' \setminus \tilde{v}$ is a disconnected graph, which can be tested using the classical depth-first search algorithm of Hopcroft and Tarjan [16] that runs in $O(r|E|)$ time.

Now assume that p decomposes G. We show how to obtain the p-decomposition of G. Let V_j, $j \in J$, be the subset of nodes of G corresponding to the connected components of $G'' \setminus \tilde{v}$. Denote by G''_j the subgraph induced by $V_j \cup \{\tilde{v}\}$, for each $j \in J$. Then the depth-first search algorithm of [16] can further be augmented to label edges of G'' corresponding to different subgraphs G''_j, $j \in J$, in $O(r|E|)$ time. Define the hypergraph G_j, for each $j \in J$, as the section hypergraph of G induced by $V_j \cup p$. It is simple to check that G_j, $j \in J$, is the p-decomposition of G. To characterize the edge set for each G_j, we first note that by definition, each $e \in E$ with $e \subseteq p$ is present in all G_j, $j \in J$. To characterize the remaining distinct edges, it suffices to label edges of G according to the labeling available for the edges of G'' as described above; suppose that for each edge e in G we associate a pointer to an edge e'' in G'' with $e \supseteq e''$. It then follows that any two edges e_1 and e_2 in G belong to the same G_j if and only if the corresponding edges e''_1 and e''_2 in G'' are present in the same G''_j. Therefore, hypergraphs G_j, $j \in J$, can now be characterized in $O(|E|)$ time. □

We should remark that for a given p that decomposes a hypergraph G into G_j, $j \in J$, the output of the p-decomposition test described in the proof of Proposition 1 provides a labeling of the edges of G that belong to exactly one new hypergraph G_j, with the understanding that the edges contained in p are present in all G_j, $j \in J$. Consequently, generating a complete list of edges belonging to each G_j for all $j \in J$ can be done in $|E| + (|J| - 1)|\bar{E}|$ steps, where $\bar{E}$ denotes the set of edges contained in p. Since, $|J| \leq |E|$ and $|\bar{E}| \leq |E|$, we conclude that the cost of storing hypergraphs G_j, $j \in J$, in the incidence-list format, in the worst case is $O(|E|^2)$.

4.2 Full-decompositions

In the following, we define a general algorithm to obtain a full-decomposition of a hypergraph G.

`Gen_dec` : General full-decomposition algorithm

Input: A hypergraph G
Output: A full-decomposition of G
Initialize the family $\mathcal{L} = \{G\}$;
while $\mathcal{L}$ *does not satisfy property (i) of full-decomposition* **do**
  select a hypergraph $\tilde{G} \in \mathcal{L}$ and $p \subset V(\tilde{G})$;
  if p *decomposes* $\tilde{G}$ **then**
    let G_j, $j \in J$, be the p-decomposition of $\tilde{G}$;
    let $\tilde{J}$ be the subset of J such that each G_j, $j \in \tilde{J}$, is not a section hypergraph of any hypergraph in $\mathcal{L}$ different from $\tilde{G}$;
    in $\mathcal{L}$, replace $\tilde{G}$ with G_j, $j \in \tilde{J}$;
return $\mathcal{L}$;

Proposition 2. *The family $\mathcal{L}$ returned by* `Gen_dec` *is a full-decomposition of G.*

Proof. To check that $\mathcal{L}$ is indeed a full-decomposition of G, it suffices to show that property (ii) of full-decomposition is satisfied; that is, in $\mathcal{L}$, there exist no two distinct hypergraphs G_s and G_t such that G_s is a section hypergraph of G_t. We prove this statement by induction on the iterations of the algorithm. That is, we now assume that it is true at the point $\tilde{G}$ and p are selected.

By condition (b) in the definition of p-decomposition, no hypergraph G_j, $j \in \tilde{J}$, can be a section hypergraph of a different hypergraph G_j, $j \in \tilde{J}$. Let $u \in \tilde{J}$, and consider the hypergraph G_u. By definition of $\tilde{J}$, the hypergraph G_u is not a section hypergraph of any hypergraph in $\mathcal{L}$ different from $\tilde{G}$. Therefore, we only need to show that no hypergraph in $\mathcal{L}$ different from $\tilde{G}$ is a section hypergraph of G_u. By induction, no hypergraph in $\mathcal{L}$ different from $\tilde{G}$ is a section hypergraph of $\tilde{G}$. As G_u is a section hypergraph of $\tilde{G}$, it follows that no hypergraph in $\mathcal{L}$ different from $\tilde{G}$ is a section hypergraph of G_u. □

In Algorithm `Gen_dec`, we have not specified which $\tilde{G} \in \mathcal{L}$ and $p \subset V(\tilde{G})$ to choose at every iteration. We refer to different choices of $\tilde{G}$ and p throughout the execution of `Gen_dec`, as *decomposition orders*. In the sequel, we denote a specific decomposition order by the sequence of choices that defines it, where each choice consists of a pair $(\tilde{G}, p)$, for some hypergraph $\tilde{G} \in \mathcal{L}$ and a set of nodes $p \subset V(\tilde{G})$ that is tested for p-decomposition of $\tilde{G}$, as described in Proposition 1. The next proposition demonstrates that a full-decomposition of a hypergraph obtained by `Gen_dec` does not depend on the specific decomposition order used.

Proposition 3. *The full-decomposition of a hypergraph obtained by Algorithm* `Gen_dec` *is independent of the decomposition order.*

Proof. Assume by contradiction that G has two different full-decompositions $\mathcal{L}_1$ and $\mathcal{L}_2$. Let $\hat{G}$ be a hypergraph with the maximum number of nodes among the hypergraphs in the symmetric difference of $\mathcal{L}_1$ and $\mathcal{L}_2$. Without loss of generality, assume $\hat{G} \in \mathcal{L}_1$. We show that $\hat{G}$ is not a section hypergraph of any hypergraph in $\mathcal{L}_2$. Otherwise, there exists $\hat{G}' \in \mathcal{L}_2$ such that $\hat{G}$ is a section hypergraph of $\hat{G}'$. Clearly $|V(\hat{G}')| > |V(\hat{G})|$. Thus, by maximality of $\hat{G}$, it follows that $\hat{G}'$ is also in $\mathcal{L}_1$, contradicting property (ii) in the definition of a full-decomposition.

Therefore, the hypergraph $\hat{G}$ is a section hypergraph of G that is not a section hypergraph of any hypergraph in $\mathcal{L}_2$. Let $(\tilde{G}, p)$ be the last pair in the decomposition order that yields $\mathcal{L}_2$ for which $\hat{G}$ is a section hypergraph of $\tilde{G}$. This implies that $\hat{G}$ is not a section hypergraph of any hypergraph in the p-decomposition of $\tilde{G}$. We will show that $p \cap V(\hat{G})$ decomposes $\hat{G}$, contradicting the fact that $\mathcal{L}_1$ is a full-decomposition of G. Let G_j, $j \in J$, denote the p-decomposition of $\tilde{G}$. Let J' be the subset of indices $j \in J$ such that G_j contains at least a node of $\hat{G}$ that is not in p. Since $\hat{G}$ is not a section hypergraph of

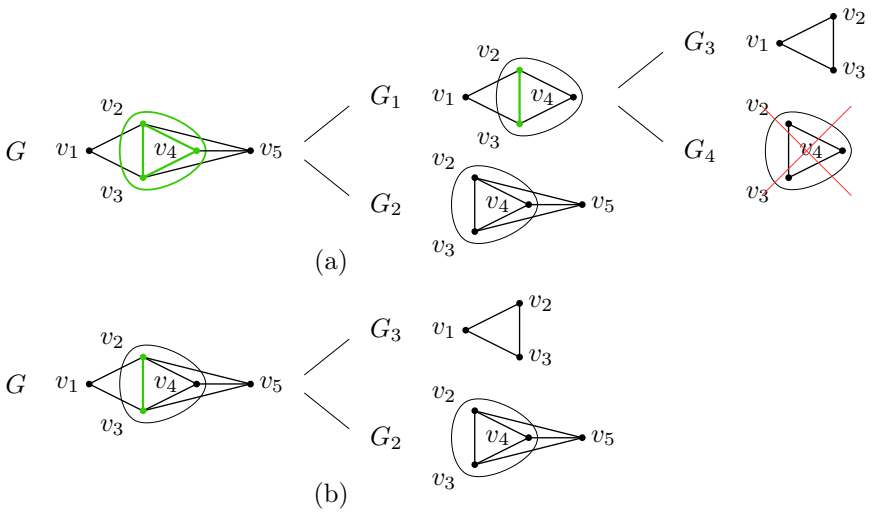

Figure 4: Using two different decomposition orders to obtain the full-decomposition of the hypergraph G in Example 3: while the full-decomposition of G is independent of decomposition orders, different decomposition orders may generate distinct redundant hypergraphs.

any G_j, $j \in J$, it follows that $|J'| \geq 2$. For every $j \in J'$, let G'_j be the section hypergraph of G_j induced by $V(G_j) \cap V(\hat{G})$. Clearly G'_j, $j \in J'$, are section hypergraphs of $\hat{G}$, and $\cup_{j \in J'} G'_j = \hat{G}$. Moreover, the hypergraph $G'_j \cap G'_{j'}$ for all $j, j' \in J'$, is the complete hypergraph on the nodes $p \cap V(\hat{G})$. This implies that $p \cap V(\hat{G})$ decomposes $\hat{G}$. However, this contradicts with the fact that $\mathcal{L}_1$ is a full-decomposition of G. □

By Proposition 3, all decomposition orders yield the same full-decomposition of a hypergraph G. Henceforth, we will speak of *the* full-decomposition of G. However, as we argue next, different decomposition orders result in different computational costs for Algorithm `Gen_dec`. Let us revisit `Gen_dec`; to ensure that property (ii) in the definition of the full-decomposition is satisfied, every time the p-decomposition of $\tilde{G}$ is generated, each new hypergraph G_j is compared with the existing ones and is added to $\mathcal{L}$ only if it is not a section hypergraph of another hypergraph in $\mathcal{L}$. Let us refer to the section hypergraphs not added to $\mathcal{L}$; i.e., G_j with $j \in J \setminus \tilde{J}$, as *redundant* hypergraphs. The following example shows that different decomposition orders in Algorithm `Gen_dec` may result in distinct redundant hypergraphs.

Example 3. *Consider the hypergraph $G = (V, E)$ depicted in Figure 4. It is simple to verify that $p_1 = \{v_2, v_3, v_4\}$ decomposes G and the p_1-decomposition of G is given by G_1, G_2, where G_1 and G_2 are section hypergraphs of G induced by $\{v_1, v_2, v_3, v_4\}$ and $\{v_2, v_3, v_4, v_5\}$, respectively. Now consider the hypergraph G_1; it can be seen that $p_2 = \{v_2, v_3\}$ decomposes G_1, and the p_2-decomposition of G_1 is given by G_3, G_4, where G_3 and G_4 are section hypergraphs of G_1 induced by $\{v_1, v_2, v_3\}$ and $\{v_2, v_3, v_4\}$, respectively. The hypergraph G_4 is redundant as it is a section hypergraph of G_2. Moreover, after additional p-decomposition tests, it can be verified that G_2 and G_3 cannot be further decomposed. Thus, we obtain the full decomposition of G given by G_2, G_3 (see Figure 4(a)). Let us denote the decomposition order used in this case by $\mathcal{O}_1$.*

Now we consider a different decomposition order $\mathcal{O}_2$ to obtain the full-decomposition of G. It is simple to check that $p_2 = \{v_2, v_3\}$ decomposes G and that the p_2-decomposition of G is given by G_2, G_3, where G_2 and G_3 are as defined above and we find that they cannot be decomposed any further after a number of p-decomposition tests, as defined by $\mathcal{O}_2$ (see Figure 4(b)).

For brevity, we have not included the full description of decomposition orders $\mathcal{O}_1$ and $\mathcal{O}_2$, utilized above. However, it is important to note in the sequence defined by $\mathcal{O}_1$, the pair (G, p_1) appears prior to the pair (G_1, p_2), whereas, in $\mathcal{O}_2$, the pair (G, p_2) appears as the first element. We will detail on the significance of this difference in the next section.

In this example, the decomposition order $\mathcal{O}_2$ seems to "outperform" $\mathcal{O}_1$: by utilizing $\mathcal{O}_2$ no redundant hypergraph was generated and the full-decomposition of G was obtained after one decomposition, while applying $\mathcal{O}_1$ leads to the generation of one redundant hypergraph, and two recursive decompositions were needed to obtain the full-decomposition. ◇

Thus far, via Example 3, we have observed that the generation of redundant hypergraphs depends on the decomposition order used in `Gen_dec`. As the redundancy check is computationally expensive in general, a natural question is whether it is possible to characterize a decomposition order that does not generate redundant hypergraphs for any input hypergraph. We will answer this question rigorously in the next section.

Before proceeding further, let us take a closer look at the computational cost of `Gen_dec`; as we detail in the next section, Algorithm `Gen_dec` can be implemented as a sequence of p-decomposition tests as defined by the specific decomposition order used. It then follows that the length of the decomposition order utilized in `Gen_dec` is a reasonable measure for the overall computational cost of this algorithm. That is, we would like to identify a decomposition order consisting of the minimum number of pairs $(\tilde{G}, p)$. Clearly if some $p \subset V(\tilde{G})$ decomposes $\tilde{G}$, then $p \in V(\tilde{G}) \cup E(\tilde{G})$. Henceforth, at each iteration of Algorithm `Gen_dec`, we select a hypergraph $\tilde{G} \in \mathcal{L}$ and a subsets of nodes $p \in V(\tilde{G}) \cup E(\tilde{G})$. Given any decomposition order $\mathcal{O}$, in order to satisfy condition (i) in the definition of the full-decomposition of a hypergraph G via Algorithm `Gen_dec`, each $p \in V(G) \cup E(G)$ has to be tested for the p-decomposition of some section hypergraph of G at least once during the execution of the algorithm; that is, each $p \in V(G) \cup E(G)$ should appear in at least one pair $(\tilde{G}, p)$ in $\mathcal{O}$. It then follows that for a hypergraph G, every decomposition order contains at least $|V(G)| + |E(G)|$ pairs. Clearly, one can use a variety of tricks to reduce the upper bound $|V(G)| + |E(G)|$, based on the structure of the given hypergraph. For instance, suppose that at given iteration, we select a pair $(\tilde{G}, p)$, where $p \in V(\tilde{G}) \cup E(\tilde{G})$. Subsequently, we apply the p-decomposition test as described by Proposition 1 and it turns out that the section hypergraph of $\tilde{G}$ induced by p is not complete. It then follows that the section hypergraph of $\tilde{G}$ induced by any $q \in V(G) \cup E(G)$ with $q \supset p$ is not complete either and therefore does not need to be considered for a p-decomposition test in `Gen_dec`. Clearly, such techniques can be incorporated in any decomposition order to reduce the running time of `Gen_dec`. However, in the remainder of this paper, for simplicity of presentation and without loss of generality, we consider a basic implementation of `Gen_dec` in which every subset of nodes $p \in V(G) \cup E(G)$ is tested for p-decomposition in the course of the algorithm. Such an assumption enables us to obtain an optimal decomposition order with the minimum length $|V(G)| + |E(G)|$, which in addition does not generate any redundant hypergraphs.

4.3 The optimal full-decomposition algorithm

In this section, we derive the "best" decomposition order for `Gen_dec`, and present an efficient algorithm to obtain the full-decomposition of a given hypergraph. To this end, we first establish an important property regarding the recursive decomposition of hypergraphs. This property enables an efficient implementation of Algorithm `Gen_dec`, by eliminating many unnecessary decomposition tests in its "while loop". In the following, we refer to hypergraphs G_j in `Gen_dec` as the *children* of $\tilde{G}$, while $\tilde{G}$ is called the *parent* of each G_j. The *ancestors* of G_j are the parent of G_j, and the ancestors of the parent of G_j. In addition, the *descendants* of $\tilde{G}$ are the children of $\tilde{G}$ and the descendants of the children of $\tilde{G}$.

Proposition 4. *Let $\tilde{G}$ be a hypergraph, and let $p \subseteq V(\tilde{G})$. Suppose that the pair $(\tilde{G}, p)$ is considered at some iteration of Algorithm* `Gen_dec`*. Then p does not decompose any descendants of $\tilde{G}$ in this algorithm.*

Proof. We assume that the section hypergraph of $\tilde{G}$ induced by p is complete, as otherwise p does not decompose any hypergraph. First assume that p does not decompose $\tilde{G}$. We show that such a p does not decompose any of the descendants of $\tilde{G}$ generated during the course of Algorithm `Gen_dec`. To obtain a contradiction, let $H_{\bar{j}}$ be the first descendant of $\tilde{G}$ generated by `Gen_dec` that can be decomposed by p. Let H denote the parent of $H_{\bar{j}}$ and suppose that $q \subset V(H)$ decomposes H into H_j, $j \in J$, where we have $\bar{j} \in J$. Note that by assumption, p does not decompose H. We now utilize p to decompose $H_{\bar{j}}$ into $H_{\bar{j}k}$, $k \in K$. Clearly, $q \subset V(H_{\bar{j}})$.

We now show that $q \nsubseteq p$. Assume by contradiction that $q \subseteq p$. Let H_p denote the complete hypergraph with node set p. Then H_p is a section hypergraph of $H_{\bar{j}}$ and of no other H_j with $j \neq \bar{j}$. It can be now checked that p decomposes H into $H_j \cup H_p$ for all $j \in J$, which gives us a contradiction. We now assume $q \nsubseteq p$. Hence $q \subset V(H_{\bar{j}\bar{k}})$ for only one $\bar{k} \in K$, since by assumption $V(H_{\bar{j}k}) \cap V(H_{\bar{j}k'}) = p$ for all $k, k' \in K$ with $k \neq k'$. Now, let $\hat{H}_{\bar{j}\bar{k}} = \bigcup_{j \in J \setminus \{\bar{j}\}} H_j \cup H_{\bar{j}\bar{k}}$. It then follows that $V(\hat{H}_{\bar{j}\bar{k}}) \cap V(H_{\bar{j}k}) = p$ for all $k \in K \setminus \{\bar{k}\}$ and it can be checked that p decomposes H into $\hat{H}_{\bar{j}\bar{k}}$ and $H_{\bar{j}k}$, $k \in K \setminus \{\bar{k}\}$, which is in contradiction with the

assumption that p does not decompose H. Hence, if p does not decompose $\tilde{G}$, then it does not decompose any descendants of $\tilde{G}$ generated by Algorithm `Gen_dec`.

Now assume that p decomposes $\tilde{G}$ into G_j, $j \in J$. Then by definition of p-decomposition, the set p does not decompose any of the resulting hypergraphs G_j and therefore, by the above proof, it does not decompose any of their descendants either. □

Next, we define a special sequence of choices $\bar{\mathcal{O}}$ in the the execution of Algorithm `Gen_dec` with highly desirable algorithmic properties. At a given iteration of `Gen_dec`, we say that $p \in V(\tilde{G}) \cup E(\tilde{G})$ is *tested* in $\tilde{G}$, if the pair $(\tilde{G}, p)$ has been already considered in an earlier iteration of `Gen_dec`. To characterize $\bar{\mathcal{O}}$, it suffices to define the pair $(\tilde{G}, p)$ at at each iteration of `Gen_dec`: at a given iteration, any hypergraph in the current family $\mathcal{L}$ can be chosen as $\tilde{G}$. Let the list $\{q_k, k \in K\}$ contain all nodes and edges of $\tilde{G}$ ordered by increasing cardinality. We define p to be the first element q_k in the above list that is *not* tested in $\tilde{G}$ and in any ancestor of $\tilde{G}$. The sequence $\bar{\mathcal{O}}$ ends when no such pair $(\tilde{G}, p)$ can be found.

Proposition 5. *The sequence $\bar{\mathcal{O}}$ is a decomposition order. Moreover, it creates no redundant hypergraphs.*

Proof. Let $\bar{\mathcal{O}}$ be given by $(G_1, p_1), (G_2, p_2), \ldots, (G_t, p_t)$, for some positive integer t. To show that $\bar{\mathcal{O}}$ is a decomposition order, we prove that it yields the full-decomposition of G. Let $\bar{\mathcal{L}}$ be the family of hypergraphs obtained by execution of Algorithm `Gen_dec` with the decomposition order $\bar{\mathcal{O}}$. Let $\tilde{G}$ denote a hypergraph in $\bar{\mathcal{L}}$. By definition of $\bar{\mathcal{O}}$, each $\tilde{p} \in V(\tilde{G}) \cup E(\tilde{G})$ is tested in $\tilde{G}$ or in an ancestor of $\tilde{G}$. We only need to show that no $\tilde{p}$ decomposes $\tilde{G}$. If $\tilde{p}$ is tested in $\tilde{G}$, then clearly $\tilde{p}$ does not decompose $\tilde{G}$. Thus, suppose that $\tilde{p}$ is not tested in $\tilde{G}$ implying $\tilde{p}$ is tested in an ancestor of $\tilde{G}$, denoted by $\hat{G}$. By Proposition 4, it follows that $\tilde{p}$ does not decompose any descendant of $\hat{G}$. As $\tilde{G}$ is a descendant of $\hat{G}$, we conclude that $\tilde{p}$ does not decompose $\tilde{G}$. Therefore, the decomposition order $\bar{\mathcal{O}}$ yields the full-decomposition of G.

We now show that decomposition order $\bar{\mathcal{O}}$ creates no redundant hypergraphs. To obtain a contradiction, let $\tilde{G}$ be the first redundant hypergraph generated and suppose that $\tilde{G}$ is one of the hypergraphs in the p_j-decomposition of G_j for some positive integer j. This implies that at the iteration of Algorithm `Gen_dec` where the pair (G_j, p_j) is selected, there exists a hypergraph $\hat{G} \in \mathcal{L}$ different from G_j such that $\tilde{G}$ is a section hypergraph of $\hat{G}$. In the sequence $(G_1, p_1), (G_2, p_2), \ldots, (G_t, p_t)$, let G_k be the last hypergraph that is an ancestor of both $\tilde{G}$ and $\hat{G}$. Let $\tilde{G}'$ be the child of G_k that is an ancestor of $\tilde{G}$, or $\tilde{G}$ itself. Similarly, let $\hat{G}'$ be the child of G_k that is an ancestor of $\hat{G}$, or $\hat{G}$ itself. Clearly $V(\tilde{G}) \subseteq V(\tilde{G}')$, and $V(\tilde{G}) \subseteq V(\hat{G}) \subseteq V(\hat{G}')$. Therefore, $V(\tilde{G}) \subseteq V(\tilde{G}') \cap V(\hat{G}') = p_k$. By definition of p-decomposition, we have that $p_j \subset V(\tilde{G})$, thus we have $p_j \subset p_k$. However, this is a contradiction, since G_k is an ancestor of G_j and, by definition of $\bar{\mathcal{O}}$, the ancestors of G_j are decomposed using sets of cardinality at most $|p_j|$. Therefore, we conclude that the decomposition order $\bar{\mathcal{O}}$ generates no redundant hypergraphs. □

As a direct consequence of Proposition 5, in Algorithm `Gen_dec` with decomposition order $\bar{\mathcal{O}}$, at every iteration we have $J = \tilde{J}$. That is, by employing $\bar{\mathcal{O}}$ in Algorithm `Gen_dec`, we can eliminate the redundancy check, which is computationally expensive in general. As we detailed before, for a hypergraph G, any decomposition order must contain at least $|V(G)| + |E(G)|$ pairs. We now show that $\bar{\mathcal{O}}$ is optimal in the sense that the p-decomposition test is performed exactly once for each $p \in V(G) \cup E(G)$.

Proposition 6. *Consider a hypergraph G with n nodes and m edges. Let the decomposition order $\bar{\mathcal{O}}$ for G be given by $(G_1, p_1), (G_2, p_2), \ldots, (G_t, p_t)$. Then $t = n + m$, and $p_i \neq p_j$ if $i \neq j$.*

Proof. Let the decomposition order $\bar{\mathcal{O}}$ for G be given by $(G_1, p_1), (G_2, p_2), \ldots, (G_t, p_t)$. We show that $p_i \neq p_j$ if $i \neq j$, which directly implies $t = n + m$, since each p_i is in $V(G) \cup E(G)$.

Assume by contradiction that there exist indices i, j with $i \neq j$ such that $p_i = p_j$. By Proposition 4, G_i is not an ancestor of G_j, and G_j is not an ancestor of G_i. In the sequence $(G_1, p_1), (G_2, p_2), \ldots, (G_t, p_t)$, let G_k be the last hypergraph that is an ancestor of both G_i and G_j. Let G_i' be the child of G_k that is an ancestor of G_i, or G_i itself. Similarly, let G_j' be the child of G_k that is an ancestor of G_j, or G_j itself. Clearly $p_i \subset V(G_i) \subseteq V(G_i')$, and $p_j \subset V(G_j) \subseteq V(G_j')$. Therefore, $p_i \subset V(G_i') \cap V(G_j') = p_k$. However, this is a contradiction, since G_k is an ancestor of G_i and, by definition of $\bar{\mathcal{O}}$, the ancestors of G_i are decomposed using sets of cardinality at most $|p_i|$. Therefore, we conclude that $p_i \neq p_j$ if $i \neq j$. □

We should remark that while $\bar{O}$ contains the minimum number of p-decomposition tests for Algorithm `Gen_dec`, it might be possible to obtain the full-decomposition of hypergraphs with a smaller number of p-decomposition tests using more sophisticated algorithmic frameworks. For example, in [29, 17], the authors utilize the concept of perfect elimination orderings for chordal graphs to identify clique separators of general graphs in n iterations, where n is the number of nodes of the graph. We leave it as an open question whether similar elimination orderings can be defined for constructing the full-decomposition of hypergraphs.

A natural question regarding the applicability of our decomposition scheme in the context of MINLP solvers is the final number and size of hypergraphs present in the full-decomposition of a given hypergraph. That is, while decomposition of a Multilinear set $\mathcal{S}_G$ into lower-dimensional Multilinear sets enables us to convexify $\mathcal{S}_G$ more efficiently, the presence of a large number of overlapping hypergraphs in the full-decomposition of G leads to a significant increase in the size of the resulting relaxations, which in turn deteriorates the performance of branch-and-cut based MINLP solvers. The following proposition shows that our decomposition algorithm always leads to relaxations of reasonable size.

Proposition 7. *Consider a hypergraph G with $n \geq 2$ nodes and $m \geq 1$ edges. Then the full-decomposition of G consists of at most $\min\{n-1, m\}$ hypergraphs. Moreover, the total number of hypergraphs generated in the course of Algorithm* `Gen_dec` *with decomposition order $\bar{\mathcal{O}}$ is at most $\min\{2n-3, 2m-1\}$.*

Proof. By Proposition 3, the full-decomposition of a hypergraph G is independent of the decomposition order. Thus in the following, we consider Algorithm `Gen_dec` with the decomposition order $\bar{\mathcal{O}}$. Both upper bounds on the final and total number of hypergraphs generated by `Gen_dec` follow directly by setting $\hat{G} := G$ and $k := 1$ in the following claim.

Claim 1. Let $\hat{G}$ be a hypergraph with $\hat{n}$ nodes and $\hat{m}$ edges considered at some iteration of Algorithm `Gen_dec` *with decomposition order $\bar{\mathcal{O}}$ applied to G. Assume that for each pair $(\tilde{G}, p)$ tested in the algorithm, where $\tilde{G}$ is $\hat{G}$ or a descendant of $\hat{G}$, we have $|p| \geq k$, for some integer $k \leq \hat{n}$. Then:*

(i) The number of hypergraphs in the full-decomposition of G that are $\hat{G}$ or descendants of $\hat{G}$ is at most $\max\{1, \hat{n} - k\}$.

(ii) The number of hypergraphs that are $\hat{G}$ or descendants of $\hat{G}$ is at most $\max\{1, 2(\hat{n} - k) - 1\}$.

(iii) The number of hypergraphs in the full-decomposition of G that are $\hat{G}$ or descendants of $\hat{G}$ is at most $\max\{1, \hat{m} - 2^k + k + 1\}$.

(iv) The number of hypergraphs that are $\hat{G}$ or descendants of $\hat{G}$ is at most $\max\{1, 2(\hat{m} - 2^k + k + 1) - 1\}$.

Proof of claim We prove the claim by induction on $\hat{n} - k \geq 0$. First, we show the base case $\hat{n} - k = 0$. In this case, for every pair $(\hat{G}, p)$ tested by the algorithm we have $|p| = k = \hat{n}$. This implies that $\hat{G}$ will not be further decomposed and hence the claim is satisfied.

Next, we assume $\hat{n} - k > 0$ and we show the inductive step. If $\hat{G}$ is not decomposable, we are done. Therefore, we now consider a pair $(\hat{G}, \hat{p})$ such that $\hat{p}$ decomposes $\hat{G}$. Let G_j, $j \in J$, be the $\hat{p}$-decomposition of $\hat{G}$, let $\hat{k} := |\hat{p}| \geq k$, let n_j be the number of nodes of G_j, and let m_j be the number of edges of G_j. Note that $\hat{k} < n_j < \hat{n}$, and

$$\hat{n} = \sum_{j \in J} (n_j - \hat{k}) + \hat{k}, \tag{22}$$

$$\hat{m} = \sum_{j \in J} (m_j - 2^{\hat{k}} + \hat{k} + 1) + 2^{\hat{k}} - \hat{k} - 1. \tag{23}$$

By definition of the decomposition order $\bar{\mathcal{O}}$, for each pair $(\tilde{G}, p)$ tested in the algorithm, where $\tilde{G}$ is G_j or a descendant of G_j, we have $|p| \geq \hat{k}$. Note that $n_j - \hat{k} < \hat{n} - k$. Hence, we can apply induction to each hypergraph G_j.

We now derive the bound given in (i). By induction, the number of hypergraphs in the full-decomposition of G that are G_j or descendants of G_j is at most $\max\{1, n_j - \hat{k}\}$. Therefore, the number of hypergraphs in

the full-decomposition of G that are $\hat{G}$ or descendants of $\hat{G}$ is at most

$$\sum_{j\in J}\max\{1, n_j - \hat{k}\} = \sum_{j\in J}(n_j - \hat{k}) = \hat{n} - \hat{k} \leq \hat{n} - k,$$

where the first equality follows from $n_j - \hat{k} \geq 1$, the second equality follows from (22), and the last inequality follows from $\hat{k} \geq k$.

Next, we derive the bound given in (ii). By induction, the number of hypergraphs that are G_j or descendants of G_j is at most $\max\{1, 2(n_j - \hat{k}) - 1\}$. Therefore, the number of hypergraphs that are $\hat{G}$ or descendants of $\hat{G}$ is at most

$$\begin{aligned}1 + \sum_{j\in J}\max\{1, 2(n_j - \hat{k}) - 1\} = 1 + \sum_{j\in J}(2(n_j - \hat{k}) - 1) = \\ = 1 + 2\sum_{j\in J}(n_j - \hat{k}) - |J| \leq 1 + 2(\hat{n} - \hat{k}) - 2 \leq 2(\hat{n} - k) - 1,\end{aligned}$$

where the first equality follows from $2(n_j - \hat{k}) - 1 \geq 1$, since $n_j - \hat{k} \geq 1$, the first inequality follows from (22) and $|J| \geq 2$, and the last inequality follows from $\hat{k} \geq k$.

We now derive the bound given in (iii). By induction, the number of hypergraphs in the full-decomposition of G that are G_j or descendants of G_j is at most $\max\{1, m_j - 2^{\hat{k}} + \hat{k} + 1\}$. Therefore, the number of hypergraphs in the full-decomposition of G that are $\hat{G}$ or descendants of $\hat{G}$ is at most

$$\begin{aligned}\sum_{j\in J}\max\{1, m_j - 2^{\hat{k}} + \hat{k} + 1\} = \sum_{j\in J}(m_j - 2^{\hat{k}} + \hat{k} + 1) = \\ = \hat{m} - 2^{\hat{k}} + \hat{k} + 1 \leq \hat{m} - 2^k + k + 1,\end{aligned}$$

where the first equality follows from $m_j - 2^{\hat{k}} + \hat{k} + 1 \geq 1$ since G_j is connected, the second equality follows from (23), and the last inequality follows from $\hat{k} \geq k$.

Finally, we derive the bound given in (iv). By induction, the number of hypergraphs that are G_j or descendants of G_j is at most $\max\{1, 2(m_j - 2^{\hat{k}} + \hat{k} + 1) - 1\}$. Therefore, the number of hypergraphs that are $\hat{G}$ or descendants of $\hat{G}$ is at most

$$\begin{aligned}1 + \sum_{j\in J}\max\{1, 2(m_j - 2^{\hat{k}} + \hat{k} + 1) - 1\} = 1 + \sum_{j\in J}(2(m_j - 2^{\hat{k}} + \hat{k} + 1) - 1) = \\ = 1 + 2\sum_{j\in J}(m_j - 2^{\hat{k}} + \hat{k} + 1) - |J| \leq 1 + 2(\hat{m} - 2^{\hat{k}} + \hat{k} + 1) - 2 \leq 2(\hat{m} - 2^k + k + 1) - 1,\end{aligned}$$

where the first equality follows from $2(m_j - 2^{\hat{k}} + \hat{k} + 1) - 1 \geq 1$ since $m_j - 2^{\hat{k}} + \hat{k} + 1 \geq 1$, the first inequality follows from (23) and $|J| \geq 2$, and the last inequality follows from $\hat{k} \geq k$. □

It can be shown that for a graph that is a path, all four bounds given in Proposition 7 are tight.

We now present an *optimal* full-decomposition algorithm, obtained by an efficient incorporation of the decomposition order $\bar{\mathcal{O}}$ in Algorithm `Gen_dec`. To simplify the presentation, at a given iteration of `Gen_dec`, we say that $p \in V(\tilde{G}) \cup E(\tilde{G})$ is *checked* in $\tilde{G}$, if p is tested in $\tilde{G}$ or in an ancestor of $\tilde{G}$. In this algorithm, the input hypergraph $G = (V, E)$ is represented by its incidence-list, where, as described before, the edges are sorted in increasing cardinality, and edges of the same cardinality are sorted lexicographically. Subsequently, each hypergraph $\tilde{G}$ generated in the course of this algorithm is characterized by two integer arrays: $I_1(\tilde{G})$ containing the indices of those edges of G that are present in $\tilde{G}$, and $I_2(\tilde{G})$ containing the indices of those elements of $V(\tilde{G}) \cup E(\tilde{G})$ that are not checked in $\tilde{G}$. Moreover, we assume the indices in I_1 and I_2 are in the same order as their corresponding nodes and edges in the hypergraph G.

`Opt_dec` **:** Optimal full-decomposition algorithm

Input: A hypergraph G
Output: The full-decomposition of G
Initialize the lists $\mathcal{L}_1 = \{G\}$ and $\mathcal{L}_2 = \{\}$;
Initialize the integer arrays $I_1(G)$ and $I_2(G)$;
while $\mathcal{L}_1$ *is nonempty* **do**
  Let $\tilde{G}$ be the first element in $\mathcal{L}_1$;
  for *each* $i \in I_2(\tilde{G})$ **do**
    if p_i *decomposes* $\tilde{G}$ **then**
      let G_j, $j \in J$, be the p_i-decomposition of $\tilde{G}$;
      remove $\tilde{G}$ from $\mathcal{L}_1$;
      for *each* $j \in J$ **do**
        if $I_2(G_j) \neq \emptyset$ **then**
          insert G_j in $\mathcal{L}_1$;
        else
          insert G_j in $\mathcal{L}_2$;
      exit the **for loop**;
  if $\tilde{G}$ *is still present in* $\mathcal{L}_1$ **then**
    remove $\tilde{G}$ from $\mathcal{L}_1$ and insert it in $\mathcal{L}_2$;
return $\mathcal{L}_2$;

In Algorithm `Opt_dec`, we define two distinct lists $\mathcal{L}_1$ and $\mathcal{L}_2$ to store the intermediate and final hypergraphs, respectively; namely, the list $\mathcal{L}_1$ contains all hypergraphs $\tilde{G}$ with at least one unchecked element $p_i \in V(\tilde{G}) \cup E(\tilde{G})$ for some $i \in I_2(\tilde{G})$, whereas, the list $\mathcal{L}_2$ contains all hypergraphs that cannot be further decomposed (by Proposition 4); i.e., $I_2(\tilde{G}) = \emptyset$ for all $\tilde{G} \in \mathcal{L}_2$. Each time $\tilde{G} \in \mathcal{L}_1$ is decomposed into G_j, $j \in J$, the hypergraph $\tilde{G}$ is removed from $\mathcal{L}_1$, hypergraphs G_j with $I_2(G_j) \neq \emptyset$ are inserted in $\mathcal{L}_1$ and hypergraphs G_j with $I_2(G_j) = \emptyset$ are inserted in $\mathcal{L}_2$. In addition, if a hypergraph $\tilde{G}$ cannot be decomposed after all sets associated with $I_2(\tilde{G})$ are tested in $\tilde{G}$, we remove it from $\mathcal{L}_1$ and insert it in $\mathcal{L}_2$. The algorithm terminates when the list $\mathcal{L}_1$ is empty. By using linked lists or dynamic arrays to store pointers to each hypergraph in $\mathcal{L}_1$ and $\mathcal{L}_2$, the above insertion and removal operations can be performed efficiently in time and memory. That is, each single insertion or removal operation can be done in $O(1)$ time, as for example, in a linked list implementation, it amounts to a simple rearrangement of pointers to the head of the list.

It is simple to see that Algorithm `Opt_dec` is an efficient implementation of the decomposition order $\bar{\mathcal{O}}$ in Algorithm `Gen_dec`. This can be seen by noting that the indices in I_2, for each $\tilde{G}$, are ordered such that the corresponding edges are sorted in increasing cardinality. Thus, by Proposition 5, we have:

Proposition 8. *Algorithm* `Opt_dec` *terminates with the full-decomposition of G and creates no redundant hypergraphs.*

Finally, we analyze the worst-case running time of Algorithm `Opt_dec` as a function of the rank, number of nodes, and number of edges of the input hypergraph G.

Proposition 9. *Consider a connected rank-r hypergraph G with n nodes and m edges. Then, the running time of Algorithm* `Opt_dec` *is $O(rm(n+m))$.*

Proof. The initialization step consists of forming the incidence-list representation of the input hypergraph G and the integer vectors $I_1(G)$, $I_2(G)$. As described before, for a rank-r hypergraph with m edges, its sorted incidence-list can be obtained in $O(rm)$ time. In addition, initializing $I_1(G)$ and $I_2(G)$ takes m and $n+m$ steps, respectively.

We now proceed to the main body of the algorithm. We claim that the "while loop" of this algorithm is executed at most $n+m$ times. In fact, each time the while loop is executed, *at least* one p-decomposition test is performed, and by Proposition 6 `Opt_dec` consists of a total number of $n+m$ p-decomposition tests.

It then follows that, the "while loop" in `Opt_dec` is executed *at most* $n+m$ times. Now consider the outer "for loop" in `Opt_dec`. Again by Proposition 6, this for loop is executed exactly $n+m$ times; that is, once for each element in $V(G) \cup E(G)$ as indicated by the I_2 arrays and it terminates when there exists no unchecked element in any of the hypergraphs generated by `Opt_dec`.

Now consider some $p_i \in V(G) \cup E(G)$ with $i \in I_2(\tilde{G})$ for some $\tilde{G} \in \mathcal{L}_1$. We would like to find an upper bound on the running time of the p_i-decomposition test for $\tilde{G}$. By assumption, the initial hypergraph G is a connected rank-r hypergraph. Moreover, it is simple to check that all children of G obtained by a single application of Proposition 1 are also connected as their corresponding graph reductions are biconnected. By a recursive application of this argument, it follows that all hypergraphs $\tilde{G}$ in $\mathcal{L}_1$ at any iteration of Algorithm `Opt_dec` are connected rank-r' hypergraphs, where $r' \leq r$. Therefore, by Proposition 1, the running time of each p-decomposition test is $O(rm)$, implying that the overall computational cost of performing p-decomposition tests in `Opt_dec` is $O(rm(n+m))$.

Finally, we analyze the cost of storing the hypergraphs generated in the course of `Opt_dec`. Clearly, for any hypergraph $\tilde{G}$ generated by this algorithm, we have $|I_1(\tilde{G})| + |I_2(\tilde{G})| \leq n + 2m$, as $|I_1(\tilde{G})| \leq m$ and $|I_2(\tilde{G})| \leq n+m$. Now, consider the hypergraphs G_j, $j \in J$, obtained from the p-decomposition of $\tilde{G}$. By Proposition 1, the output of a p-decomposition test provides a labeling of the edges of $\tilde{G}$ that belong to exactly one new hypergraph G_j, with the understanding that the edges contained in p are present in all G_j, $j \in J$. It then follows that for each G_j, the integer arrays $I_1(G_j)$ and $I_2(G_j)$ can be constructed in $O(n+m)$ steps. Furthermore, by Proposition 7, the total number of hypergraphs generated by Algorithm `Opt_dec` is $O(\min\{n,m\})$. Hence, the overall cost of storing hypergraphs in the proposed algorithm is $O(\min\{n,m\}(n+m))$. As we described before, by employing a linked list implementation of $\mathcal{L}_1$ and $\mathcal{L}_2$, each single insertion or removal of a hypergraph can be done in constant time, implying that the overall cost of insertion and removal operations is $O(n+m)$.

Thus, the total running time of Algorithm `Opt_dec` is given by $O(rm(n+m))$. □

As we described throughout this section, in comparison to Algorithm `Gen_dec` with an arbitrary decomposition order, the advantages of Algorithm `Opt_dec` are two folds. First, the number of p-decomposition tests applied by `Opt_dec` to obtain the full-decomposition of a hypergraph with n nodes and m edges is exactly $n+m$, which is the *minimum* number of tests needed to obtain the full-decomposition of any hypergraph. Second, no redundant hypergraph is generated in the course of `Opt_dec`, and hence the costly redundancy test (as described in `Gen_dec`) is not required. The following example demonstrates that Algorithm `Opt_dec` significantly outperforms a naive implementation of Algorithm `Gen_dec`.

Example 4. *Consider a hypergraph $G = (V, E)$ with $V := \{v_i : i = 1, \ldots, r\} \cup \{w_i : i = 1, \ldots, r\}$ for some $r \geq 3$. Let E contain the following set of edges: $E_1 := \{\{v_i, w_i\} : i = 1, \ldots, r\}\}$ and $E_2 := \{\{v_i : i \in I\} : I \subseteq \{1, \ldots, r\},\ |I| \geq 2\}$. In this case, we have $n := |V| = 2r$ and $m := |E| = 2^r - 1$. In the following, we denote by $K_{l,q}$ the complete hypergraph on the nodes $\{v_l, \ldots, v_q\}$, where $1 \leq l \leq q \leq r$.*

We first utilize Algorithm `Opt_dec` to decompose G: after r p-decomposition tests where $p = v_i$ for $i = 1, \ldots, r$, we obtain $r+1$ hypergraphs, r of which consist of a single edge of the form $\{v_i, w_i\}$, for $i = 1, \ldots, r$, and the last one is the complete hypergraph $K_{1,r}$. By performing an additional $n+m-r = r+m$ p-decomposition tests, Algorithm `Opt_dec` confirms that these $r+1$ hypergraphs cannot be further decomposed and thus form the full-decomposition of G. Clearly, we could improve the performance of Algorithm `Opt_dec`, by inserting every new complete hypergraph G_j in $\mathcal{L}_2$ without performing any additional p-decomposition test, as a complete hypergraph is not decomposable.

Next, we demonstrate the significance of our optimal decomposition algorithm by analyzing the performance of a naive implementation of Algorithm `Gen_dec` applied to the hypergraph G defined above. That is, we define a decomposition order different from $\bar{\mathcal{O}}$ and we do not make use of Proposition 4 to eliminate unnecessary p-decomposition tests. Suppose that in the first iteration of `Gen_dec`, we choose $p = \{v_1, \ldots, v_r\}$. It then follows that `Gen_dec` decomposes G into r hypergraphs of the form $G_i = K_{1,r} \cup H_i$, for all $i = 1, \ldots, r$, where H_i consists of the single edge $\{v_i, w_i\}$. In the next iteration, `Gen_dec` selects one of these hypergraphs; without loss of generality, suppose that we pick G_1. Subsequently, `Gen_dec` perform $|V(K_{2,r})| + |E(K_{2,r})| = 2^{r-1} - 1$ p-decomposition tests for all $p \in V(K_{2,r}) \cup E(K_{2,r})$. It is simple to check that none of such tests decomposes G_1. In the next iteration, we let $p = \{v_1, \ldots, v_{r-1}\}$. It then follows that `Gen_dec` decomposes G_1 into the two hypergraphs $K_{1,r-1} \cup H_1$ and $K_{1,r}$. At this stage, performing the redundancy test reveals that $K_{1,r}$ is a redundant hypergraph, as for example it is a section hypergraph of G_2. Next, we select the hypergraph $\tilde{G} = K_{1,r-1} \cup H_1$

and apply $|V(K_{2,r-1})| + |E(K_{2,r-1})| = 2^{r-2} - 1$ *p-decomposition tests for all* $p \in V(K_{2,r-1}) \cup E(K_{2,r-1})$*, none of which decompose* $\tilde{G}$*. In the next iteration, we let* $p = \{v_1, \ldots, v_{r-2}\}$ *to obtain a decomposition of* $\tilde{G}$ *given by* $K_{1,r-2} \cup H_1$ *and* $K_{1,r-1}$*. Again, it is simple to check that* $K_{1,r-1}$ *is a redundant hypergraph. Applying such a decomposition order recursively, it can be shown that the total number of p-decomposition tests performed in the course of the algorithm is given by* $n+m+r(\sum_{i=1}^{r-1} 2^i - 1)+r(r-1) = n+m+n(m-1)/2$*. That is, while Algorithm* `Opt_dec` *requires* $n+m$ *decomposition tests, this naive implementation of* `Gen_dec`*, requires* $n(m-1)/2$ *additional p-decomposition tests to obtain a full-decomposition of* G*. In addition, the total number of redundant hypergraphs generated in the process is given by* $r(r-1) - 1 = n(n-2)/4 - 1$*. Thus, we conclude that the algorithmic enhancements presented in this section have a significant impact on the performance of the proposed decomposition algorithm.* ◇

We conclude this paper by noting that an interesting future direction is to develop an optimal decomposition algorithm that incorporates our theoretical results presented in Section 3; namely, it would be interesting to investigate an optimal decomposition of hypergraphs with sparse intersections. Such a generalization will enable us to decompose many more types of Multilinear sets into simpler sets.

References

[1] X. Bao, A. Khajavirad, N.V. Sahinidis, and M. Tawarmalani. Global optimization of nonconvex problems with multilinear intermediates. *Mathematical Programming Computation*, 7(1):1–37, 2015.

[2] X. Bao, N.V. Sahinidis, and M. Tawarmalani. Multiterm polyhedral relaxations for nonconvex, quadratically-constrained quadratic programs. *Optimization Methods and Software*, 24:485–504, 2009.

[3] F. Barahona. The max-cut problem on graphs not contractible to K_5. *Operations Research Letters*, 2(3):107–111, 1983.

[4] F. Barahona and A.R. Mahjoub. On the cut polytope. *Mathematical Programming*, 36:157–173, 1986.

[5] F. Barahona and A.R. Mahjoub. Compositions of graphs and polyhedra I–IV. *SIAM Journal on Discrete Mathematics*, 7(3):359–402, 1994.

[6] C. Berge. *Hypergraphs: Combinatorics of Finite Sets*. North-Holland Mathematical Library. Elsevier Science, 1984.

[7] E. Boros and P.L. Hammer. The max-cut problem and quadratic $0-1$ optimization; polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, 33:151–180, 1991.

[8] C. Buchheim and G. Rinaldi. Efficient reduction of polynomial zero-one optimization to the quadratic case. *SIAM Journal on Optimization*, 18:1398–1413, 2007.

[9] V. Chvátal. On certain polytopes associated with graphs. *Journal of Combinatorial Theory, Series B*, 18(2):138–154, 1975.

[10] M. Conforti and K. Pashkovich. The projected faces property and polyhedral relations. *Mathematical Programming, Series A*, 156(1–2):331–342, 2016.

[11] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.

[12] Y. Crama. Concave extensions for non-linear $0-1$ maximization problems. *Mathematical Programming*, 61:53–60, 1993.

[13] Y. Crama and E. Rodríguez-Heck. A class of valid inequalities for multilinear 0–1 optimization problems. *Discrete Optimization*, 2017.

[14] A. Del Pia and A. Khajavirad. The multilinear polytope for γ-acyclic hypergraphs. *Under review*, 2016.

[15] A. Del Pia and A. Khajavirad. A polyhedral study of binary polynomial programs. *Mathematics of Operations Research*, 42(2):389–410, 2017.

[16] J. Hopcroft and R.Tarjan. Efficient algorithms for graph manipulation. *Communications of the ACM*, 16:372–378, 1973.

[17] H.-G. Leimer. Optimal decomposition by clique separators. *Discrete mathematics*, 113(1-3):99–123, 1993.

[18] J. Luedtke, M. Namazifar, and J.T. Linderoth. Some results on the strength of relaxations of multilinear functions. *Mathematical Programming*, 136:325–351, 2012.

[19] F. Margot. *Composition de polytopes combinatoires: une approche par projection*. PhD thesis, École polytechnique fédérale de Lausanne, 1994.

[20] G.P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.

[21] R. Misener and C.A. Floudas. ANTIGONE: algorithms for continuous/integer global optimization of nonlinear equations. *Journal of Global Optimization*, 59(2-3):503–526, 2014.

[22] R. Misener, J.B. Smadbeck, and C.A. Floudas. Dynamically generated cutting planes for mixed-integer quadratically constrained quadratic programs and their incorporation into GloMIQO 2. *Optimization Methods and Software*, 30(1):215–249, 2015.

[23] M. Namazifar. *Strong relaxations and computations for multilinear programming*. PhD thesis, University of Wisconsin–Madison, 2011.

[24] M. Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45:139–172, 1989.

[25] A.D. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10:425–437, 1997.

[26] N.V. Sahinidis. *BARON 14.3.1: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2014.

[27] H.D. Sherali. Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *Acta Mathematica Vietnamica*, 22:245–270, 1997.

[28] H.D. Sherali and W.P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal of Discrete Mathematics*, 3:411–430, 1990.

[29] R.E. Tarjan. Decomposition by clique separators. *Discrete mathematics*, 55(2):221–232, 1985.

[30] M. Tawarmalani. Inclusion certificates and simultaneous convexification of functions. *Working paper*, 2010.

[31] M. Tawarmalani and N.V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Dordrecht, 2002.

[32] S. Vigerske and A. Gleixner. SCIP: Global optimization of mixed-integer nonlinear programs in a branch-and-cut framework. Technical Report 16–24, ZIB, Berlin, 2016.