Factorized binary polynomial optimization

Alberto Del Pia *

July 4, 2024

Abstract

In *binary polynomial optimization,* the goal is to find a binary point maximizing a given polynomial function. In this paper, we propose a novel way of formulating this general optimization problem, which we call *factorized binary polynomial optimization.* In this formulation, we assume that the variables are partitioned into a fixed number of sets, and that the objective function is written as a sum of r products of linear functions, each one involving only variables in one set of the partition. Our main result is an algorithm that solves factorized binary polynomial optimization in strongly polynomial time, when r is fixed. This result provides a vast new class of tractable instances of binary polynomial optimization, and it even improves on the state-of-the-art for quadratic objective functions, both in terms of generality and running time. We demonstrate the applicability of our result through the *binary tensor factorization problem,* which arises in mining discrete patterns in data, and that contains as a special case the *rank*-1 *Boolean tensor factorization problem.* Our main result implies that these problems can be solved in strongly polynomial time, if the input tensor has fixed rank, and a rank factorization is given. For the *rank*-1 *Boolean matrix factorization problem,* we only require that the input matrix has fixed rank.

Key words: binary polynomial optimization; binary quadratic optimization; polynomial time algorithm; binary tensor factorization; Boolean tensor factorization; Boolean matrix factorization

1 Introduction

Binary polynomial optimization, i.e., the problem of finding a binary point maximizing a given polynomial function, is a fundamental problem in discrete optimization with a wide range of applications across science and engineering. To formulate this optimization problem, we use the hypergraph representation from [14]. Given a hypergraph $H = (V, E)$, with edges of cardinality at least two, and $c : V \cup E \to \mathbb{Q}$, the associated optimization problem is

$$\begin{array}{ll} \max\limits_{x} & \sum\limits_{k \in V} c_k x_k + \sum\limits_{e \in E} c_e \prod\limits_{k \in E} x_k \\ \text{s. t.} & x_k \in \{0,1\} \qquad \forall k \in V. \end{array} \tag{E}$$

We call this problem *Explicit binary polynomial optimization,* since the polynomial objective function is given explicitly via its nonzero monomials. This problem is strongly NP-hard [26], and a recent stream of research led to the discovery of several polynomially solvable classes [3, 5, 6, 11–20].

In this paper, we propose a different way of formulating binary polynomial optimization, and show that it leads to the discovery of a vast class of tractable instances. In this different formulation,

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

we assume that the variables are partitioned into s sets, and that the polynomial objective function is given in a factorized form, written as a sum of products of linear functions, each one involving only variables in one set of the partition. Formally, this is the optimization problem

$$\begin{aligned} \max_{x^1,\dots,x^s} \quad & \sum_{I\in\mathcal{I}} \prod_{j\in I} \langle c^{I,j}, x^j \rangle \\ \text{s.t.} \quad & x^j \in \{0,1\}^{n_j} \qquad \forall j \in [s], \end{aligned} \tag{F}$$

where we denote by $\langle a, b\rangle$ the inner product of two vectors a, b of the same dimension. The data of the problem consists of positive integers s and n_j, for $j \in [s]$, a nonempty family $\mathcal{I}$ of subsets of $[s]$, and vectors $c^{I,j} \in \mathbb{Q}^{n_j}$ for $I \in \mathcal{I}$, $j \in I$. We call this problem *Factorized binary polynomial optimization.*

Reading this paper, the reader should always think to the following two parameters as being fixed numbers: $\max\{|e| : e \in E\}$ in Problem E, and s in Problem F. On the one hand, this is the setting in most applications of practical interest. On the other hand, by far most research so far is devoted to the cases in which these two parameters are equal to 2 [33, 40], which are already compelling and strongly NP-hard [26, 40].

While quite different on the surface, we will show that Problems E and F are, in fact, equivalent: we can reformulate Problem E as Problem F, and vice versa. A key advantage of Problem F is that it can reveal some inner sparsity of the problem that is hidden in Problem E. For example, classes of Problem E with $|E|$ as large as $\prod_{j\in[s]} n_j$, can be written in the form of Problem F, with $|\mathcal{I}| = 1$. In this paper, we show how we can exploit this inner sparsity of the input data to design an efficient algorithm. The main result of this paper is an algorithm that solves in strongly polynomial time Problem F, when $|\mathcal{I}|$ is a fixed number.

Theorem 1. *Problem F can be solved in strongly polynomial time, for any fixed s and $|\mathcal{I}|$.*

It is interesting to observe that, in Theorem 1, it is fundamental that, in each product in the objective function of Problem F, there is at most one linear function for each set of the partition. In fact, it is known that it is NP-hard to maximize the product of just two linear functions, plus a third linear function, over all binary points [7, 29]. As a corollary to Theorem 1, we obtain that also the slightly more general optimization problem obtained from Problem F by replacing each linear function in the objective with an affine function, can be solved in strongly polynomial time under the same assumptions (see Corollary 7).

Our algorithm for Problem F is combinatorial in nature, and is based on the construction of hyperplane arrangements. It is well understood that enumerating cells of hyperplane arrangements, or equivalently vertices of a zonotope, can help in the design of efficient algorithms with theoretical guarantees (see, e.g., [1,9,10,25]). Most of these algorithms construct only one arrangement in their execution. A key feature of our algorithm is that we construct, recursively, $s-1$ arrangements on top of each other. In the first iteration, we construct an arrangement of hyperplanes corresponding to the vector of variables x^s. These hyperplanes live in a carefully chosen low-dimensional "dual" space, corresponding to the products in the objective function that involve variables x^s, and at least another x^j. For each cell of the arrangement, we are able to construct a corresponding partial solution $\bar{x}^s \in \{0,1\}^{n_s}$, and a child instance of Problem F only in variables $x^1, x^2, \dots, x^{s-1}$. We then recurse on each child instance. This gives rise to a compact tree of subproblems, and at least one of the leaves will correspond to an optimal solution of Problem F. Our technique results in a quite elegant algorithm that is simple to state and implement, and only relies on the construction of hyperplane arrangements.

A consequence of the equivalence of Problems E and F, is that Problem F inherits the applications of Problem E. To demonstrate the applicability of Theorem 1, we consider the *rank-t binary tensor factorization problem* and the *rank-t Boolean tensor factorization problem.* In these two problems, the goal is to find t rank-1 binary tensors whose sum, or disjunction (respectively), best approximates a given tensor of order s. These problems arise in mining discrete patterns in data and have plenty of applications [24,39,43,47]. Their solutions provide a very useful tool for analyzing tensors to discover latent factors from them, and produce more interpretable and sparser results than normal factorization methods [38]. Most existing methods to tackle these problems rely on heuristics, and hence do not provide any guarantee on the quality of the solution [2,24,38,39,43]. The reader should always think to the parameters s and t, in these problems, as being fixed numbers: the case $s = 2$, $t = 1$ is the *rank-1 Boolean matrix factorization problem,* which is already NP-hard [28], well-studied, and with many applications (see, e.g., [27,34,35,37,45]). We will see that Theorem 1 implies the following tractability results.

Corollary 1. *The rank-t binary tensor factorization problem on a tensor of order s can be solved in strongly polynomial time if s, t are fixed, the input tensor has fixed rank, and a rank factorization is given.*

Corollary 2. *The rank-1 Boolean tensor factorization problem on a tensor of order s can be solved in strongly polynomial time if s is fixed, the input tensor has fixed rank, and a rank factorization is given.*

In fact, we do not even need a rank factorization of the input tensor, and any factorization of the tensor as the sum of a fixed number of rank-1 tensors is sufficient (see Corollaries 10 and 11). For the rank-1 Boolean matrix factorization problem, we obtain:

Corollary 3. *The rank-1 Boolean matrix factorization problem can be solved in strongly polynomial time if the input matrix has fixed rank.*

While the main contribution of this paper is Theorem 1 and its consequences, in this work we also lay the foundations for a theoretical study of Problem F, since, to the best of our knowledge, this problem has not been considered before. In Section 2, we see that Problem F is strongly NP-hard even in very restrictive settings. In Section 3, we discuss the relationship of Problems E and F: we show that the two problems are equivalent, that Problem F can reveal some inner sparsity of the problem that is hidden in Problem E, and we provide a link with the concept of tensor factorization and rank of a tensor. Section 4 is devoted to the binary tensor factorization problem and the Boolean tensor factorization problem. Our algorithm, the proof of Theorem 1 and its corollaries, are presented in Section 5.

In Section 6, we compare our Theorem 1 with known tractability results. The focus of this paper is on polynomial objective functions of degree three or more. In this case, we are not aware of any other known polynomially solvable class of Problem F, and we show that our result does not follow from known tractable classes of Problem E. Interestingly, our result also improves on the state-of-the-art for quadratic objective functions. Consider the optimization problem

$$\begin{aligned} \max_{x^1,\dots,x^s} \quad & \sum_{i,j\in[s],\ i<j} {x^i}^{\mathsf{T}} Q^{i,j} x^j + \sum_{j\in[s]} {c^j}^{\mathsf{T}} x^j \\ \text{s.t.} \quad & x^j \in \{0,1\}^{n_j} \qquad \forall j \in [s], \end{aligned} \tag{Q}$$

where $Q^{i,j} \in \mathbb{Q}^{n_i \times n_j}$, for every $i, j \in [s]$ with $i < j$, and $c^j \in \mathbb{Q}^{n_j}$, for every $j \in [s]$. As a corollary to Theorem 1, we obtain the following result:

Corollary 4. *Problem Q can be solved in strongly polynomial time, provided that s is fixed, and the rank of each matrix $Q^{i,j}$, for $i, j \in [s]$ with $i < j$, is fixed.*

It is known that Problem Q can be solved in polynomial time if $s = 2$ and the rank of $Q^{1,2}$ is fixed [32, 40, 42]. Therefore, Corollary 4 extends previously known tractable classes, both in terms of generality and running time. Furthermore, when $s = 2$, our algorithm constructs only one hyperplane arrangement, and our work results in significantly cleaner and shorter algorithm and arguments.

We remark that the main emphasis of this paper lies on the theoretical computational complexity of the algorithms presented. We do not refer to practically efficient implementations of the algorithms, which we believe should be studied in the future.

2 NP-hardness

In the next result, we show that Problem F is strongly NP-hard even in very restrictive settings. Our reduction is inspired from the one in the proof of Theorem 10.2 in [40], where the authors show that Problem Q is strongly NP-hard for $s = 2$. The similarity lies in the introduction of a copy of the original variables, and the addition of a penalty in the objective function to have the original variables match the copy, in an optimal solution. The main differences are: 1) Our reduction is form *simple max cut,* rather than from *binary quadratic optimization,* which allows us to show the hardness of the problem in very restrictive settings; 2) Our reduction is to Problem F with $s = 2$, whose objective function is given in a factorized form.

Proposition 1. *Problem F is strongly NP-hard even if $s = 2$, $n_1 = n_2$, $|\mathcal{I}| = n_1 + 2$, and $c_k^{I,j}$ is integer and bounded by n_1 in absolute value, for every $I \in \mathcal{I}$, $j \in I$, $k \in [n_j]$.*

Proof. Our reduction is from simple max cut, which is strongly NP-hard [26]. It is well-known that the simple max cut problem on a graph $G = (V, E)$ with $V = [n]$ can be formulated as

$$\begin{aligned} \max_{x} \quad & \sum_{\{i,j\} \in E,\ i<j} (x_i + x_j - 2x_i x_j) \\ \text{s.t.} \quad & x \in \{0,1\}^n . \end{aligned} \tag{MC}$$

Now consider the optimization problem

$$\begin{aligned} \max_{x,y} \quad & \sum_{\{i,j\} \in E,\ i<j} (x_i + y_j - 2x_i y_j) + n \sum_{i \in [n]} (2x_i y_i - x_i - y_i) \\ \text{s.t.} \quad & x, y \in \{0,1\}^n . \end{aligned} \tag{MC'}$$

We claim that Problem MC can be solved by Problem MC', in the sense that, given an optimal solution (x, y) to Problem MC', then x is optimal to Problem MC.

Denote by $\text{obj}(x)$ the objective value of x in Problem MC and by $\text{obj}'(x, y)$ the objective value of (x, y) in Problem MC'. It is simple to check that, given $x, y \in \{0,1\}^n$ with $x = y$, then $\text{obj}'(x, y) = \text{obj}(x)$. It then suffices to show that if (x, y) is optimal to Problem MC', then $x = y$. Let (x, y) be optimal to Problem MC' and assume, for a contradiction, that there is at least one index $\ell \in [n]$ such that $x_\ell \neq y_\ell$. Let $\tilde{y}$ be obtained from y by flipping component ℓ, so that $\tilde{y}_\ell = x_\ell$. We show that the objective value of $(x, \tilde{y})$ is strictly larger than the objective value of (x, y), which

contradicts the optimality of (x, y):

$$\begin{aligned}
\mathrm{obj}'(x,y) - \mathrm{obj}'(x,\tilde{y}) &= \sum_{\{i,j\}\in E,\ i<j} (1-2x_i)\,(y_j - \tilde{y}_j) + n \sum_{i\in[n]} (2x_i - 1)\,(y_i - \tilde{y}_i) \\
&= \sum_{i:\{i,\ell\}\in E,\ i<\ell} (1-2x_i)\,(y_\ell - \tilde{y}_\ell) + n\,(2x_\ell - 1)\,(y_\ell - \tilde{y}_\ell) \\
&= \sum_{i:\{i,\ell\}\in E,\ i<\ell} (1-2x_i)\,(1-2x_\ell) + n\,(2x_\ell - 1)\,(1-2x_\ell) \\
&= |\{i \in [\ell-1] : \{i,\ell\} \in E\}| - n \\
&\le (n-1) - n = -1.
\end{aligned}$$

To complete the proof, we write Problem MC' in the form of Problem F with $s = 2$, $n_1 = n_2 = n$, $|\mathcal{I}| = n + 2$, $x^1 = x$, and $x^2 = y$. We rewrite the objective function of Problem MC' as the sum of the following three functions:

$$\begin{aligned}
f(x) &:= \sum_{\{i,j\}\in E,\ i<j} x_i - n \sum_{i\in[n]} x_i \\
&= \sum_{i\in[n]} x_i \left(|\{j : \{i,j\} \in E,\ i < j\}| - n\right), \\
g(y) &:= \sum_{\{i,j\}\in E,\ i<j} y_j - n \sum_{j\in[n]} y_j \\
&= \sum_{j\in[n]} y_j \left(|\{i : \{i,j\} \in E,\ i < j\}| - n\right), \\
h(x,y) &:= -2 \sum_{\{i,j\}\in E,\ i<j} x_i y_j + 2n \sum_{i\in[n]} x_i y_i \\
&= \sum_{i\in[n]} \underbrace{2x_i}_{=:h_i(x)} \underbrace{\left(n y_i - \sum_{j:\{i,j\}\in E,\ i<j} y_j \right)}_{=:h_i(y)}.
\end{aligned}$$

Now, the function $f(x)$ corresponds to a set $I = \{1\}$ with $c^{I,1}$ the vector of coefficients of the linear function $f(x)$. The function $g(y)$ corresponds to a set $I = \{2\}$ with $c^{I,2}$ the vector of coefficients of the linear function $g(y)$. For every $i \in [n]$, the product $h_i(x)h_i(y)$ in $h(x, y)$ corresponds to a set $I = \{1, 2\}$ with $c^{I,1}$ and $c^{I,2}$ the vectors of coefficients of the linear functions $h_i(x)$ and $h_i(y)$, respectively. It is simple to check that all these vectors have integer components bounded by n in absolute value. □

3 Binary polynomial optimization: explicit vs factorized

In this section we compare Problem E and Problem F.

3.1 Equivalency of Problems E and F

Our first goal is to establish the equivalency of Problems E and F. While it is clear that we can directly reformulate Problem F as Problem E, by expanding all products in the objective function,

the opposite direction is less obvious. To reformulate Problem E as Problem F, we expand the technique that we used in the proof of Proposition 1, from quadratics to higher degree polynomials: the constructed instance of Problem F features s copies of the original variables, where s denotes the degree of the objective function of Problem E, and a penalty in the objective function forces all the copies to match, in an optimal solution.

In the remainder of the paper, we say that a hypergraph H is *s-partite,* if its node set can be partitioned into s sets, called the *sides* of H, such that every edge contains at most one node from each side.

Proposition 2. *Problem E on a hypergraph $H = (V, E)$ can be reformulated, in strongly polynomial time, as Problem F with $s := \max\{|e| : e \in E\}$ and $|\mathcal{I}| \leq |V| + |E| + s$.*

Proof. Consider an instance of Problem E, and let $\text{obj}(x)$ be its objective function. Construct a new instance of Problem E as follows: 1) The instance has variables $x^1, x^2, \ldots, x^s$, where $s := \max\{|e| : e \in E\}$, and each x^j, for $j \in [s]$, is a copy of the original vector of variables x of the original instance. 2) Let $f(x^1, x^2, \ldots, x^s)$ be obtained from $\text{obj}(x)$ by replacing, for every $e \in E$, each x_k with one of its copies x_k^j, for $j \in [s]$, so that the product $\prod_{k \in E} x_k$ contains at most one variable from each x^j, $j \in [s]$. 3) The objective function of the new instance is then defined by

$$\text{obj}'(x^1, x^2, \ldots, x^s) := f(x^1, x^2, \ldots, x^s) + M \sum_{k \in V} \left(s \prod_{j \in [s]} x_k^j - \sum_{j \in [s]} x_k^j \right),$$

where $M := \sum_{e \in E} |c_e| + 1$.

Note that the hypergraph $H' = (V', E')$ associated with the new instance of Problem E is s-partite, and is constructed as follows: 1) V' contains s copies of V, which we denote by V^j, for $j \in [s]$. 2) For every edge $e \in E$, we have an edge $e' \in E'$ obtained from e by replacing each node with one if its copies, so that e' contains at most one node from each V^j, for $j \in [s]$; Furthermore, for every $k \in V$, we have an edge in E' that contains all s copies of k. Therefore, $|E'| \leq |V| + |E|$.

Let $(x^1, x^2, \ldots, x^s)$ be an optimal solution to the new instance. We show that x^1 is optimal to the original instance. Define, for every $k \in V$, the penalty function

$$p_k(x_k^1, x_k^2, \ldots, x_k^s) := s \prod_{j \in [s]} x_k^j - \sum_{j \in [s]} x_k^j.$$

Given $(x^1, x^2, \ldots, x^s)$ binary with $x^1 = x^2 = \cdots = x^s$, we have $p_k(x_k^1, x_k^2, \ldots, x_k^s) = 0$ for every $k \in V$, thus $\text{obj}'(x^1, x^2, \ldots, x^s) = \text{obj}(x^1)$. It then suffices to show that if $(x^1, x^2, \ldots, x^s)$ is optimal to the new instance, then $x^1 = x^2 = \cdots = x^s$. Let $(x^1, x^2, \ldots, x^s)$ be optimal to the new instance and assume, for a contradiction, that there is at least one $\ell \in V$ such that $x_\ell^1 = x_\ell^2 = \cdots = x_\ell^s$ does not hold. Let $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^s)$ be obtained from $(x^1, x^2, \ldots, x^s)$ by setting all components $\tilde{x}_\ell^1 = \tilde{x}_\ell^2 = \cdots = \tilde{x}_\ell^s$ to the same binary value. We show that $\text{obj}'(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^s)$ is strictly larger than $\text{obj}'(x^1, x^2, \ldots, x^s)$, which contradicts the optimality of $(x^1, x^2, \ldots, x^s)$. First, by definition of M,

$$f(x^1, x^2, \ldots, x^s) - f(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^s) < M.$$

Next, we have

$$\begin{aligned} &p_\ell(x_\ell^1, x_\ell^2, \ldots, x_\ell^s) - p_\ell(\tilde{x}_\ell^1, \tilde{x}_\ell^2, \ldots, \tilde{x}_\ell^s) \leq -1 \\ &p_k(x_k^1, x_k^2, \ldots, x_k^s) - p_k(\tilde{x}_k^1, \tilde{x}_k^2, \ldots, \tilde{x}_k^s) = 0 \qquad \forall k \in V \setminus \{\ell\}. \end{aligned}$$

Hence,

$$\text{obj}'(x^1, x^2, \ldots, x^s) - \text{obj}'(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^s) < M - M = 0.$$

To complete the proof, we write the new instance in the form of Problem F with s as above, $n_j = |V|$ for every $j \in [s]$, and $|\mathcal{I}| \leq |V| + |E| + s$. For every $j \in [s]$, the part of the objective function of the new instance that is linear in variables x^j corresponds to a set $I = \{j\}$ with $c^{I,j}$ the vector of coefficients of the variables x^j. Next, consider a single nonlinear monomial in the objective function of the new instance, which we can write as $c \prod_{j \in I} x^j_{k_j}$, for some $I \subseteq [s]$ and $k_j \in [n_j]$, for $j \in [s]$. This monomial can be written as $\prod_{j \in I} \langle c^{I,j}, x^j \rangle$, where the $c^{I,j}_{k_j}$, for every $j \in I$, are set in any way so that their product is c, and every other component of $c^{I,j}$, for every $j \in I$, is set to zero. □

3.2 Tensor factorizations and rank

In Section 3.1, we saw that Problems E and F can be polynomially reduced into each other. However, in some cases, one formulation can be significantly more succinct than the other. In the next example we see that, starting from an instance of Problem F with $|\mathcal{I}| = 1$, and expanding all products in the objective function, it is possible to obtain an instance of Problem E with $\prod_{j \in [s]} n_j$ monomials with nonzero coefficients. In particular, the hypergraph associated with the obtained instance of Problem E is s-partite, s-uniform, and contains all possible $\prod_{j \in [s]} n_j$ edges. Recall that a hypergraph is *s-uniform* if all its edges have cardinality s.

Example 1. *Consider the special case of Problem F with $\mathcal{I} = \{I\}$, $I = [s]$, and with each vector $c^{I,j} \in \mathbb{Q}^{n_j}$, for every $j \in [s]$, having all nonzero components. We expand all products in the objective function and obtain:*

$$\prod_{j \in [s]} \left(\sum_{k_j \in [n_j]} c^{I,j}_{k_j} x^j_{k_j} \right) = \sum_{k_1 \in [n_1]} \cdots \sum_{k_s \in [n_s]} c^{I,1}_{k_1} c^{I,2}_{k_2} \cdots c^{I,s}_{k_s} \cdot x^1_{k_1} x^2_{k_2} \cdots x^s_{k_s}.$$

The hypergraph associated with the obtained instance of Problem E is s-partite and s-uniform. Since each product $c^{I,1}_{k_1} c^{I,2}_{k_2} \cdots c^{I,s}_{k_s}$, for $k_1 \in [n_1], k_2 \in [n_2], \ldots, k_s \in [n_s]$, is nonzero, and no cancellation is possible, it contains all possible $\prod_{j \in [s]} n_j$ edges.

Example 1 highlights how Problem F can provide a much sparser formulation with respect to Problem E, revealing some inner sparsity of the problem that is hidden in Problem E. Our next goal is to connect this phenomenon to the concept of tensor factorizations and rank of a tensor.

To avoid unnecessary complicated notation, we mainly focus on the special cases of Problems E and F that we implicitly considered in Example 1. For Problem E, we focus on the special case in which the associated hypergraph H is s-partite, with sides V^j, for $j \in [s]$, and s-uniform. As a result, H has at most $\prod_{j \in [s]} n_j$ edges, where $n_j := |V^j|$, for every $j \in [s]$. We can write each edge $e \in E$ in the form $e = (k_1, k_2, \ldots, k_s)$, for some $k_1 \in [n_1]$, $k_2 \in [n_2]$, $\ldots$, $k_s \in [n_s]$. We can then write this special case of Problem E in the form

$$\begin{aligned} \max_{x^1, \ldots, x^s} \quad & \sum_{(k_1, \ldots, k_s) \in E} c_{k_1, \ldots, k_s} \prod_{j \in [s]} x^j_{k_j} \\ \text{s.t.} \quad & x^j \in \{0,1\}^{n_j} \qquad \forall j \in [s]. \end{aligned} \tag{E-U}$$

Note that the degree of each monomial in the objective function is s. Furthermore, we can define the associated $n_1 \times n_2 \times \cdots \times n_s$ tensor of the coefficients of the objective function $C := (c_{k_1, \ldots, k_s})$,

with the understanding that $c_{k_1,\dots,k_s}$ is set to zero, if $(k_1,\dots,k_s) \notin E$. On the other hand, in the special case of Problem F that we consider, we have that all sets $I \in \mathcal{I}$ coincide with the set $[s]$. We can then set $r := |\mathcal{I}|$ in Problem F and write this special case of Problem F in the form:

$$\begin{aligned} \max_{x^1,\dots,x^s} \quad & \sum_{i\in[r]} \prod_{j\in[s]} \langle c^{i,j}, x^j \rangle \\ \text{s.t.} \quad & x^j \in \{0,1\}^{n_j} \qquad \forall j \in [s]. \end{aligned} \tag{F-U}$$

Next, we show that Problem F-U provides a much sparser formulation with respect to Problem E-U, when the tensor C of the coefficients of the objective function of Problem E-U can be written as a sum of few rank-1 tensors:

$$\sum_{i\in[r]} (c^{i,1} \otimes c^{i,2} \otimes \cdots \otimes c^{i,s}), \tag{1}$$

where $\otimes$ denotes the vector outer product, and where $c^{i,j} \in \mathbb{Q}^{n_j}$, for every $i \in [r]$ and $j \in [s]$. In this case, we say that (1) is a *factorization* of C.

Observation 1. *Assume that the tensor C of the coefficients of the objective function of Problem E-U has a factorization of the form* (1)*. Then, Problem E-U can be reformulated as Problem F-U, where r and the $c^{i,j}$ are the ones from* (1)*.*

Proof. Let $C = (c_{k_1,\dots,k_s})$ be the $n_1 \times n_2 \times \cdots \times n_s$ tensor of the coefficients of the objective function of Problem E-U. Since it has a factorization of the form (1), the entries of C are

$$c_{k_1,k_2,\dots,k_s} = \sum_{i\in[r]} \prod_{j\in[s]} c^{i,j}_{k_j} \qquad \forall k_1 \in [n_1],\ k_2 \in [n_2], \dots, k_s \in [n_s].$$

The objective function of Problem E-U can then be written in the form of the objective function Problem F-U as follows:

$$\begin{aligned} \sum_{(k_1,\dots,k_s)\in E} c_{k_1,\dots,k_s} \prod_{j\in[s]} x^j_{k_j} &= \sum_{k_1\in[n_1]} \cdots \sum_{k_s\in[n_s]} c_{k_1,\dots,k_s} \prod_{j\in[s]} x^j_{k_j} \\ &= \sum_{k_1\in[n_1]} \cdots \sum_{k_s\in[n_s]} \left(\sum_{i\in[r]} \prod_{j\in[s]} c^{i,j}_{k_j} \right) \prod_{j\in[s]} x^j_{k_j} \\ &= \sum_{i\in[r]} \sum_{k_1\in[n_1]} \cdots \sum_{k_s\in[n_s]} \prod_{j\in[s]} c^{i,j}_{k_j} x^j_{k_j} \\ &= \sum_{i\in[r]} \prod_{j\in[s]} \left(\sum_{k_j\in[n_j]} c^{i,j}_{k_j} x^j_{k_j} \right) \\ &= \sum_{i\in[r]} \prod_{j\in[s]} \langle c^{i,j}, x^j \rangle . \end{aligned}$$

□

Observation 1 allows us to draw a connection with the concept of tensor rank. The *rank* of an $n_1 \times n_2 \times \cdots \times n_s$ tensor C is the minimum number r such that C has a factorization of the form (1). In this case, we say that (1) is a *rank factorization* of C. It is well-known that the rank of an

$n_1 \times n_2 \times \cdots \times n_s$ tensor C with $s \geq 2$ is less than or equal to the product of the $s-1$ smallest numbers among $n_1, n_2, \ldots, n_s$. In the case $s = 2$, the definition of rank and of rank factorization of a tensor reduce to the well-known definitions for a matrix. In fact, we have

$$\sum_{i \in [r]} (c^{i,1} \otimes c^{i,2}) = AB^{\mathsf{T}},$$

where A is the $n_1 \times r$ matrix with column i equal to $c^{i,1}$, and B is the $n_2 \times r$ matrix with column i equal to $c^{i,2}$, for $i \in [r]$. While it is possible to determine the rank of a matrix in polynomial time, for example via Gaussian elimination, determining the rank of a tensor is NP-hard, even for a tensor of order 3. This follows from [30], which details how to encode any given 3SAT Boolean formula in n variables and m clauses as an $(n+2m+2) \times 3n \times (3n+m)$ tensor C with the property that the 3SAT formula is satisfiable if and only if the rank of C is at most $4n + 2m$.

The next result characterizes the best possible sparsity of Problem F-U in terms of the rank of the tensor of the coefficients of the objective function of Problem E-U.

Observation 2. *Problem E-U can be reformulated as Problem F-U, where the number r in Problem F-U equals the rank of the tensor C of the coefficients of the objective function of Problem E-U. This is best possible, in the sense that it is not possible to write the objective function of Problem E-U in the form of the objective function of Problem F-U with a strictly smaller r.*

Proof. The first part of the statement follows directly from the definition of rank and Observation 1. In the remainder of the proof we prove the second part of the statement. Assume that the objective function of Problem E-U can be written in the form of the objective function of Problem F-U:

$$\sum_{i \in [r]} \prod_{j \in [s]} \left\langle c^{i,j}, x^j \right\rangle,$$

where $c^{i,j} \in \mathbb{Q}^{n_j}$, for every $i \in [r]$ and $j \in [s]$. Let $C = (c_{k_1,\ldots,k_s})$ be the $n_1 \times n_2 \times \cdots \times n_s$ tensor of the coefficients of the objective function of Problem E-U. Following the same derivation in the proof of Observation 1, but in the reverse order, we obtain that the entries of C are

$$c_{k_1,k_2,\ldots,k_s} = \sum_{i \in [r]} \prod_{j \in [s]} c^{i,j}_{k_j} \qquad \forall k_1 \in [n_1],\ k_2 \in [n_2], \ldots, k_s \in [n_s].$$

Hence, C has a factorization of the form (1). From the definition of rank, we have that r is greater than or equal to the rank of C. □

The discussion above about the complexity of determining the rank of a tensor, highlights a key difference in the complexity of the reductions in Proposition 2 and Observations 1 and 2: The reformulation in Proposition 2 can be constructed in strongly polynomial time. On the other hand, the reformulation in Observation 1 can be obtained in strongly polynomial time if the factorization (1) of C is given. Furthermore, the reformulation in Observation 2 can be obtained in strongly polynomial time if a rank factorization of C is given, and it cannot be obtained in polynomial time, in general, unless P=NP.

Observations 1 and 2 are stated in terms of Problems E-U and F-U. Similar results can be also obtained if we consider, instead of Problem E-U, Problem E in which the associated hypergraph is s-partite with sides V^j, for $j \in I$. In other words, Problem E-U without the assumption that the hypergraph is s-uniform. A key difference, which results in a significantly more complicated notation, is that, in this more general problem, we can no longer encode the coefficients of the

objective function with only one tensor. Instead, we need at most $2^s - 1$ tensors: one for each subset I of $[s]$ such that there is an edge that contains one node in each V^j, for $j \in I$. In these results, we then need to replace Problem F-U with Problem F. For this pair of more general problems, one can obtain the following results, which correspond to Observations 1 and 2.

Observation 3. *Consider Problem E on an s-partite hypergraph. Assume that all the tensors of the coefficients of the objective function have a factorization of the form* (1)*. Then, this problem can be reformulated as Problem F, where $|\mathcal{I}|$ equals the sum of all the r in* (1)*, and the $c^{I,j}$ are the ones from* (1)*.*

Observation 4. *Consider Problem E on an s-partite hypergraph. This problem can be reformulated as Problem F, where $|\mathcal{I}|$ in Problem F equals the sum of the ranks of the tensors of the coefficients of the objective function of Problem E. This is best possible, in the sense that it is not possible to write the objective function of Problem E in the form of the objective function of Problem F with a strictly smaller $|\mathcal{I}|$.*

Observations 3 and 4 follow by applying Observations 1 and 2, respectively, to each tensor of the coefficients of the objective function of the considered Problem E.

4 Binary tensor factorization and Boolean tensor factorization

To formally introduce the binary tensor factorization problem, and the Boolean tensor factorization problem, we first define the binary rank and the Boolean rank of a tensor. The *binary rank* of an $n_1 \times n_2 \times \cdots \times n_s$ tensor C is the minimum number r such that C has a factorization of the form (1), where $c^{i,j} \in \{0,1\}^{n_j}$, for every $i \in [r]$ and $j \in [s]$. On the other hand, the *Boolean rank* of an $n_1 \times n_2 \times \cdots \times n_s$ tensor C is the minimum number r such that C has a factorization of the form

$$\bigvee_{i \in [r]} (c^{i,1} \otimes c^{i,2} \otimes \cdots \otimes c^{i,s}), \tag{2}$$

where $\vee$ denotes the component-wise "or" operation, and where $c^{i,j} \in \{0,1\}^{n_j}$, for every $i \in [r]$ and $j \in [s]$. We refer the interested reader to [31] for more informations on the different notions of rank. In the *rank-t binary tensor factorization problem,* we are given an $n_1 \times n_2 \times \cdots \times n_s$ tensor $A = (a_{k_1,\ldots,k_s})$ and an integer t. The goal is to find an $n_1 \times n_2 \times \cdots \times n_s$ tensor $B = (b_{k_1,\ldots,k_s})$ of binary rank t that minimizes the objective function

$$\sum_{k_1 \in [n_1]} \cdots \sum_{k_s \in [n_s]} (a_{k_1,\ldots,k_s} - b_{k_1,\ldots,k_s})^2 . \tag{3}$$

Also in the *rank-t Boolean tensor factorization problem,* we are given an $n_1 \times n_2 \times \cdots \times n_s$ tensor $A = (a_{k_1,\ldots,k_s})$ and an integer t. However, the goal is to find an $n_1 \times n_2 \times \cdots \times n_s$ tensor $B = (b_{k_1,\ldots,k_s})$ of Boolean rank t that minimizes the objective function (3). In the special case $t = 1$ the two problems coincide. The case $s = 2$, $t = 1$ is the *rank-1 Boolean matrix factorization problem.* It is shown in [40] that the latter problem can be formulated as Problem Q with $s = 2$. Next, we extend this result to tensors of any order, and to the more general rank-t binary tensor factorization problem.

Proposition 3. *The rank-t binary tensor factorization problem on an $n_1 \times n_2 \times \cdots \times n_s$ tensor $A = (a_{k_1,\ldots,k_s})$ can be formulated as the following Problem E on an st-partite hypergraph:*

$$\begin{aligned} \max_{x^{i,j}} \quad & \sum_{k_1 \in [n_1]} \cdots \sum_{k_s \in [n_s]} \left[(2a_{k_1,\ldots,k_s} - 1) \sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} - 2 \sum_{i,i' \in [t],\ i<i'} \prod_{j \in [s]} x_{k_j}^{i,j} x_{k_j}^{i',j} \right] \\ \text{s.t.} \quad & x^{i,j} \in \{0,1\}^{n_j} \qquad \forall i \in [t],\ j \in [s]. \end{aligned} \tag{BTF}$$

Proof. Consider the binary tensor factorization problem. For every $k_1 \in [n_1], k_2 \in [n_2], \ldots, k_s \in [n_s]$ we have

$$b_{k_1,\ldots,k_s} = \sum_{i \in [t]} x_{k_1}^{i,1} x_{k_2}^{i,2} \cdots x_{k_s}^{i,s}.$$

Since $x^{i,j}$ is binary, for every $i \in [t]$, $j \in [s]$, we have, for every $k_1 \in [n_1], k_2 \in [n_2], \ldots, k_s \in [n_s]$:

$$\begin{aligned} (a_{k_1,\ldots,k_s} - b_{k_1,\ldots,k_s})^2 &= \left(a_{k_1,\ldots,k_s} - \sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} \right)^2 \\ &= a_{k_1,\ldots,k_s}^2 - 2a_{k_1,\ldots,k_s} \sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} + \left(\sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} \right)^2 \\ &= a_{k_1,\ldots,k_s}^2 - 2a_{k_1,\ldots,k_s} \sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} + \sum_{i \in [t]} \prod_{j \in [s]} \left(x_{k_j}^{i,j} \right)^2 + 2 \sum_{i,i' \in [t],\ i<i'} \prod_{j \in [s]} x_{k_j}^{i,j} x_{k_j}^{i',j} \\ &= a_{k_1,\ldots,k_s}^2 - 2a_{k_1,\ldots,k_s} \sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} + \sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} + 2 \sum_{i,i' \in [t],\ i<i'} \prod_{j \in [s]} x_{k_j}^{i,j} x_{k_j}^{i',j} \\ &= a_{k_1,\ldots,k_s}^2 + (1 - 2a_{k_1,\ldots,k_s}) \sum_{i \in [t]} \prod_{j \in [s]} x_{k_j}^{i,j} + 2 \sum_{i,i' \in [t],\ i<i'} \prod_{j \in [s]} x_{k_j}^{i,j} x_{k_j}^{i',j}. \end{aligned}$$

Our binary tensor factorization problem can then be formulated as Problem E. Note that, in each monomial in the objective function, no two variables from the same vector $x^{i,j}$, for $i \in [t]$, $j \in [s]$, are ever multiplied together in the same monomial. Therefore, this optimization problem is Problem E on an st-partite hypergraph, with $t(n_1 + n_2 + \cdots + n_s)$ variables, and with an objective function of degree $2s$. □

Corollary 5. *The rank-1 Boolean tensor factorization problem on an $n_1 \times n_2 \times \cdots \times n_s$ tensor $A = (a_{k_1,\ldots,k_s})$ can be formulated as the following Problem E-U on an s-partite hypergraph:*

$$\begin{aligned} \max_{x^1,\ldots,x^s} \quad & \sum_{k_1 \in [n_1]} \cdots \sum_{k_s \in [n_s]} (2a_{k_1,\ldots,k_s} - 1) \prod_{j \in [s]} x_{k_j}^j \\ \text{s.t.} \quad & x^j \in \{0,1\}^{n_j} \qquad \forall j \in [s]. \end{aligned} \tag{BTF$_1$}$$

Proof. In the special case $t = 1$, the rank-1 Boolean tensor factorization problem coincides with the rank-1 binary tensor factorization problem. It then follows from Proposition 3 that the rank-1 Boolean tensor factorization problem on an $n_1 \times n_2 \times \cdots \times n_s$ tensor $A = (a_{k_1,\ldots,k_s})$ can be formulated as Problem BTF$_1$, which is of the form of Problem E on an s-partite hypergraph. Note that, in each monomial in the objective function, precisely one variable from each vector x^j, for $j \in [s]$, is present. Therefore, this optimization problem is of the form of Problem E-U. □

Next, we see how the binary tensor factorization problem and the rank-1 Boolean tensor factorization problem can be formulated succinctly as Problem F and Problem F-U, respectively, if a factorization of the input tensor A is given. This result will allow us to show our tractability results for the binary tensor factorization problem and its special cases, Corollaries 1 to 3, as consequences of our main result.

Proposition 4. *Consider the rank-t binary tensor factorization problem on an $n_1 \times n_2 \times \cdots \times n_s$ tensor*

$$A = \sum_{p \in [q]} \left(a^{p,1} \otimes a^{p,2} \otimes \cdots \otimes a^{p,s} \right),$$

where $a^{p,j} \in \mathbb{Q}^{n_j}$, for every $p \in [q]$, $j \in [s]$. Then, the problem can be reformulated, in strongly polynomial time, as Problem F with $s := st$ and with $|\mathcal{I}| := tq + (t^2 + t)/2$.

Proof. Let $A = (a_{k_1,\ldots,k_s})$. From Proposition 3, the rank-t binary tensor factorization problem can be formulated as Problem BTF. In this problem, the coefficients of the objective function can be encoded with one tensor for each sum inside the square brackets in the objective function.

First, we consider the first sum, over all $i \in [t]$, and we consider now one specific such i. The corresponding tensor of coefficients is an $n_1 \times n_2 \times \cdots \times n_s$ tensor, and the entry in position $(k_1, k_2, \ldots, k_s)$ is $2a_{k_1,\ldots,k_s} - 1$. Clearly, we have

$$2A = \sum_{p \in [q]} \left(a^{p,1} \otimes a^{p,2} \otimes \cdots \otimes 2a^{p,s} \right).$$

For every $j \in [s]$, let e^j be the vector in $\mathbb{Q}^{n_j}$ with all components equal to one. Then, the $n_1 \times n_2 \times \cdots \times n_s$ tensor with all entries equal to one can be written as

$$e^1 \otimes e^2 \otimes \cdots \otimes (-e^s).$$

We can now write the above tensor of coefficients in the form

$$\sum_{p \in [q]} \left(a^{p,1} \otimes a^{p,2} \otimes \cdots \otimes 2a^{p,s} \right) + \left(e^1 \otimes e^2 \otimes \cdots \otimes (-e^s) \right).$$

Next, we consider the second sum inside the square brackets in the objective function. This sum is over all $i, i' \in [t]$ with $i < i'$, and we consider now one specific such pair i, i'. The corresponding tensor of coefficients is an $n_1 \times n_1 \times n_2 \times n_2 \times \cdots \times n_s \times n_s$ tensor, and all the entries are equal to -2. We can then write the above tensor of coefficients as

$$e^1 \otimes e^2 \otimes \cdots \otimes (-2e^s).$$

Problem BTF is a special case of Problem E. It then follows from Observation 3 that the rank-t binary tensor factorization problem can be formulated as Problem F, with $s := st$ and $|\mathcal{I}| = t(q+1) + (t^2 - t)/2 = tq + (t^2 + t)/2$. □

Corollary 6. *Consider the rank-1 Boolean tensor factorization problem on an $n_1 \times n_2 \times \cdots \times n_s$ tensor*

$$A = \sum_{p \in [q]} \left(a^{p,1} \otimes a^{p,2} \otimes \cdots \otimes a^{p,s} \right),$$

where $a^{p,j} \in \mathbb{Q}^{n_j}$, for every $p \in [q]$, $j \in [s]$. Then, the problem can be reformulated, in strongly polynomial time, as Problem F-U with the same s, n_j, for $j \in [s]$, and with $r := q + 1$.

More applications. We conclude this section by mentioning some additional applications for which Problem E, with an associated s-partite hypergraph, and Problem F provide a better fit than Problem E. In [40] the author provides a number of applications of Problem Q with $s = 2$. These are the *maximum weight biclique problem*, the *maximal sum submatrix problem*, and the problem of finding the *cut-norm of a matrix*. Problem E, with an associated s-partite hypergraph, allows us to directly formulate problems that contain the ones above as special cases. In the maximum weight biclique problem, instead of a bipartite graph, we can consider a multipartite graph with more than two sides, or even a multipartite hypergraph. In the maximal sum submatrix problem, and in the problem of finding the cut-norm of a matrix, instead of a matrix in input, we can have a tensor of any order. If this input tensor is given through a factorization of the form (1), then these problems can be naturally formulated in the form of Problem F-U, due to Observation 1. We leave the details of these formulations to the reader.

5 Proofs of main results

In this section, we prove our main results. We begin by stating an extended version of Theorem 1, presented in Section 1. In Problem F, we denote by m_j, for $j \in [s]$, the number of sets $I \in \mathcal{I}$ that contain index j and at least one index strictly smaller than j.

Theorem 2. *Problem F can be solved with*

$$s\theta^{s-1} n_2^{m_2} n_3^{m_3} \cdots n_s^{m_s} \operatorname{poly}(n_1, n_2, \ldots, n_s, |\mathcal{I}|)$$

arithmetic operations, where θ is a constant. Furthermore, the size of the numbers produced in the course of the execution of the algorithm is polynomial in the size of the input. In particular, Problem F can be solved in strongly polynomial time, provided that $s, m_2, \ldots, m_s$ are fixed.

Clearly, Theorem 2 implies Theorem 1, since $|\mathcal{I}|$ in Problem F is an upper bound on m_j, for every $j \in [s]$. It is interesting to note that the parameter m_1, unlike all other m_j, for $j \in [s] \setminus \{1\}$, never appears as an exponent in the running time in Theorem 2, so it does not need to be fixed to obtain a polynomial time algorithm. Furthermore, due to the symmetry of the problem, we can assume without loss of generality that m_1 is the largest of the m_j, for $j \in [s]$. This detail is lost in Theorem 1. The proof of Theorem 2 is presented in the next section.

5.1 Proof of Theorem 2

Our algorithm is based on the construction of hyperplane arrangements. We now introduce the proper terminology. A finite family of hyperplanes in $\mathbb{R}^d$ defines a dissection of $\mathbb{R}^d$ into connected sets of various dimensions. We call this dissection the *arrangement* of these hyperplanes. The connected sets of dimension d are called *cells* of the arrangement. We refer the reader to [21, 22] for more information about hyperplane arrangements.

5.1.1 The algorithm

Case $s = 1$. We set $\bar{x}^1 \in \{0,1\}^{n_1}$ so that:

$$\bar{x}^1_k = \begin{cases} 1 & \text{if } \sum_{I \in \mathcal{I}} c^{I,1}_k > 0 \\ 0 & \text{if } \sum_{I \in \mathcal{I}} c^{I,1}_k < 0 \end{cases} \qquad \forall k \in [n_1], \tag{4}$$

meaning that in the case $\sum_{I\in\mathcal{I}} c_k^{I,1} = 0$, we can set arbitrarily $\bar{x}_k^1$ to either 0 or 1. Return the solution $(\bar{x}^1)$.

Case $s \geq 2$**.** We partition the family $\mathcal{I}$ into $\mathcal{I}_\alpha$, $\mathcal{I}_\beta$, and $\mathcal{I}_\gamma$ as follows: $\mathcal{I}_\alpha$ contains the sets $I \in \mathcal{I}$ that do not contain index s; $\mathcal{I}_\beta$ contains the sets $I \in \mathcal{I}$ that strictly contain $\{s\}$; $\mathcal{I}_\gamma$ contains the sets $I \in \mathcal{I}$ that coincide with $\{s\}$. Note that $|\mathcal{I}_\beta| = m_s$. For every $I \in \mathcal{I}_\beta$, define variable λ_I. For every $k \in [n_s]$, define the affine function

$$\begin{aligned} h_k : \quad & \mathbb{R}^{\mathcal{I}_\beta} \to \mathbb{R} \\ & \lambda \mapsto \sum_{I\in\mathcal{I}_\beta} \lambda_I c_k^{I,s} + \sum_{I\in\mathcal{I}_\gamma} c_k^{I,s}, \end{aligned} \tag{5}$$

and the corresponding hyperplane in $\mathbb{R}^{\mathcal{I}_\beta}$:

$$H_k := \left\{\lambda \in \mathbb{R}^{\mathcal{I}_\beta} : h_k(\lambda) = 0\right\} \qquad \forall k \in [n_s]. \tag{6}$$

Construct the arrangement of these hyperplanes in $\mathbb{R}^{\mathcal{I}_\beta}$ with the algorithm in [21, 22], and denote by $\mathcal{A}$ the set of cells of the arrangement.

The remainder of the algorithm should be applied, separately, to each cell $C \in \mathcal{A}$, so we now fix one cell $C \in \mathcal{A}$. The cell C induces a signing of the hyperplanes H_k, for every $k \in [n_s]$. Namely, for every $k \in [n_s]$, we know which inequality among $h_k(\lambda) \geq 0$ and $h_k(\lambda) \leq 0$ is valid for C, and note that precisely one of the two inequalities is valid for C, since C is full-dimensional. We then define the *partial solution* corresponding to cell C as $\bar{x}^s \in \{0,1\}^{n_s}$ as follows:

$$\bar{x}_k^s := \begin{cases} 1 & \text{if } h_k(\lambda) \geq 0 \text{ is valid for } C \\ 0 & \text{if } h_k(\lambda) \leq 0 \text{ is valid for } C \end{cases} \qquad \forall k \in [n_s]. \tag{7}$$

Define the *child instance* of Problem F corresponding to cell C as the optimization problem obtained, from Problem F, by fixing x^s to $\bar{x}^s$:

$$\begin{aligned} \sum_{I\in\mathcal{I}_\gamma} \left\langle c^{I,s}, \bar{x}^s \right\rangle + \max_{x^1,\ldots,x^{s-1}} \quad & \sum_{I\in\mathcal{I}_\alpha} \prod_{j\in I} \left\langle c^{I,j}, x^j \right\rangle + \sum_{I\in\mathcal{I}_\beta} \left\langle c^{I,s}, \bar{x}^s \right\rangle \prod_{j\in I\setminus\{s\}} \left\langle c^{I,j}, x^j \right\rangle \\ \text{s.t.} \quad & x^j \in \{0,1\}^{n_j} \qquad \forall j \in [s-1]. \end{aligned} \tag{F_C}$$

Problem F_C is essentially a new instance of Problem F with the parameter s decreased by one. However, there are two minor differences that we should point out. First, there is a constant term $\sum_{I\in\mathcal{I}_\gamma} \left\langle c^{I,s}, \bar{x}^s \right\rangle$ in the objective function, that we brought outside of the max. Second, there is a constant factor $\left\langle c^{I,s}, \bar{x}^s \right\rangle$ in the sum over $I \in \mathcal{I}_\beta$. This second issue can be easily remedied by absorbing the scalar $\left\langle c^{I,s}, \bar{x}^s \right\rangle$ in precisely one of the subsequent inner products $\left\langle c^{I,j}, x^j \right\rangle$, say the one corresponding to the smallest $j \in I \setminus \{s\}$. If we denote this index by $\hat{j}$, this is accomplished by redefining $c^{I,\hat{j}}$ as $\left\langle c^{I,s}, \bar{x}^s \right\rangle c^{I,\hat{j}}$ in the new instance.

We then apply the algorithm recursively to Problem F_C. Let $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1})$ be the solution returned by the algorithm applied to Problem F_C. The algorithm then returns the solution $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1}, \bar{x}^s)$ to Problem F.

5.1.2 Correctness

In this section, we show that the algorithm presented in Section 5.1.1 is correct, that is, that it returns an optimal solution to Problem F.

The proof is by induction on s. In the base case we assume $s = 1$, and so all sets $I \in \mathcal{I}$ are equal to $\{1\}$. Problem F takes the form

$$\begin{aligned} \max_{x^1} \quad & \sum_{I \in \mathcal{I}} \langle c^{I,1}, x^1 \rangle \\ \text{s.t.} \quad & x^1 \in \{0,1\}^{n_1} . \end{aligned}$$

The objective function is linear in x^1 and can be written in the form

$$\sum_{I \in \mathcal{I}} \langle c^{I,1}, x^1 \rangle = \left\langle \sum_{I \in \mathcal{I}} c^{I,1}, x^1 \right\rangle = \sum_{k \in [n_1]} \left(\sum_{I \in \mathcal{I}} c^{I,1}_k \right) x^1_k.$$

It is then simple to check that the solution $(\bar{x}^1)$ defined in (4) and returned by the algorithm is optimal to Problem F.

Next, we consider the induction step, and we assume $s \geq 2$. In our first claim, below, we write the objective function of Problem F as a linear function in x^s, if we think to all other variables x^j, for $j \in [s-1]$, as being fixed. Let $\mathcal{I}_\alpha$, $\mathcal{I}_\beta$, and $\mathcal{I}_\gamma$ be the partition of the family $\mathcal{I}$ defined in the algorithm.

Claim 1. *The objective function of Problem F can be written in the form*

$$\sum_{I \in \mathcal{I}_\alpha} \prod_{j \in I} \langle c^{I,j}, x^j \rangle + \sum_{k \in [n_s]} h_k(\lambda) x^s_k, \tag{8}$$

where, for every $k \in [n_s]$, $h_k(\lambda)$ is the affine function defined by the algorithm in (5)*, and where*

$$\lambda_I := \prod_{j \in I \setminus \{s\}} \langle c^{I,j}, x^j \rangle \qquad \forall I \in \mathcal{I}_\beta.$$

Proof. The objective function of Problem F can be written in the form

$$\sum_{I \in \mathcal{I}} \prod_{j \in I} \langle c^{I,j}, x^j \rangle = \sum_{I \in \mathcal{I}_\alpha} \prod_{j \in I} \langle c^{I,j}, x^j \rangle + \sum_{I \in \mathcal{I}_\beta} \prod_{j \in I} \langle c^{I,j}, x^j \rangle + \sum_{I \in \mathcal{I}_\gamma} \prod_{j \in I} \langle c^{I,j}, x^j \rangle . \tag{9}$$

Since each $I \in \mathcal{I}_\beta$ strictly contains $\{s\}$, we can write the second summand in (9) as follows:

$$\begin{aligned} \sum_{I \in \mathcal{I}_\beta} \prod_{j \in I} \langle c^{I,j}, x^j \rangle &= \sum_{I \in \mathcal{I}_\beta} \left(\prod_{j \in I \setminus \{s\}} \langle c^{I,j}, x^j \rangle \right) \langle c^{I,s}, x^s \rangle \\ &= \left\langle \sum_{I \in \mathcal{I}_\beta} \left(\prod_{j \in I \setminus \{s\}} \langle c^{I,j}, x^j \rangle \right) c^{I,s}, x^s \right\rangle \\ &= \sum_{k \in [n_s]} \left(\sum_{I \in \mathcal{I}_\beta} \prod_{j \in I \setminus \{s\}} \langle c^{I,j}, x^j \rangle \, c^{I,s}_k \right) x^s_k. \end{aligned}$$

Since each $I \in \mathcal{I}_\gamma$ coincides with $\{s\}$, the third summand in (9) can be written as follows:

$$\begin{aligned}\sum_{I\in\mathcal{I}_\gamma}\prod_{j\in I}\langle c^{I,j},x^j\rangle &= \sum_{I\in\mathcal{I}_\gamma}\langle c^{I,s},x^s\rangle\\ &= \left\langle \sum_{I\in\mathcal{I}_\gamma} c^{I,s},x^s\right\rangle\\ &= \sum_{k\in[n_s]}\left(\sum_{I\in\mathcal{I}_\gamma} c_k^{I,s}\right)x_k^s.\end{aligned}$$

From (9), we can then write the objective function of Problem F in the form

$$\begin{aligned}&\sum_{I\in\mathcal{I}_\alpha}\prod_{j\in I}\langle c^{I,j},x^j\rangle + \sum_{k\in[n_s]}\left(\sum_{I\in\mathcal{I}_\beta}\prod_{j\in I\setminus\{s\}}\langle c^{I,j},x^j\rangle\, c_k^{I,s}\right)x_k^s + \sum_{k\in[n_s]}\left(\sum_{I\in\mathcal{I}_\gamma} c_k^{I,s}\right)x_k^s =\\ &= \sum_{I\in\mathcal{I}_\alpha}\prod_{j\in I}\langle c^{I,j},x^j\rangle + \sum_{k\in[n_s]}\left[\left(\sum_{I\in\mathcal{I}_\beta}\prod_{j\in I\setminus\{s\}}\langle c^{I,j},x^j\rangle\, c_k^{I,s}\right) + \left(\sum_{I\in\mathcal{I}_\gamma} c_k^{I,s}\right)\right]x_k^s\\ &= \sum_{I\in\mathcal{I}_\alpha}\prod_{j\in I}\langle c^{I,j},x^j\rangle + \sum_{k\in[n_s]}\left[\sum_{I\in\mathcal{I}_\beta}\lambda_I c_k^{I,s} + \sum_{I\in\mathcal{I}_\gamma} c_k^{I,s}\right]x_k^s\\ &= \sum_{I\in\mathcal{I}_\alpha}\prod_{j\in I}\langle c^{I,j},x^j\rangle + \sum_{k\in[n_s]} h_k(\lambda)x_k^s.\end{aligned}$$

□

Note that, in (8), the first sum and each $h_k(\lambda)$, for $k \in [n_s]$, generally depend on x^j, for $j \in [s-1]$, but do not depend on x^s. Claim 1 allows us to characterize the vector x^s of all optimal solutions to Problem F, as we see in the next claim.

Claim 2. *Let $\breve{x}$ be an optimal solution to Problem F and let*

$$\breve{\lambda}_I := \prod_{j\in I\setminus\{s\}}\langle c^{I,j},\breve{x}^j\rangle \qquad \forall I \in \mathcal{I}_\beta.$$

We have

$$\breve{x}_k^s = \begin{cases}1 & \text{if } h_k(\breve{\lambda}) > 0\\ 0 & \text{if } h_k(\breve{\lambda}) < 0.\end{cases} \qquad \forall k \in [n_s]. \tag{10}$$

Furthermore, each solution obtained from $\breve{x}$ by flipping the value of any number of variables $\breve{x}_k^s$, $k \in [n_s]$, such that $h_k(\breve{\lambda}) = 0$, is optimal to Problem F as well.

Proof. For a contradiction, assume there is $k \in [n_s]$ such that $\breve{x}_k^s$ does not satisfy the corresponding condition in (10). Consider now the solution obtained from $\breve{x}$ by flipping the value of variable $\breve{x}_k^s$, from 0 to 1, or from 1 to 0. It follows from Claim 1 that the objective value of this new solution is strictly larger than the objective value of $\breve{x}$. This contradicts the optimality of $\breve{x}$, thus $\breve{x}$ must satisfy conditions (10).

Next, let $k \in [n_s]$ such that $h_k(\breve{\lambda}) = 0$. Consider the solution obtained from $\breve{x}$ by flipping the value of variable $\breve{x}_k^s$. It follows from Claim 1 that this new solution has the same objective value as $\breve{x}$, so it is optimal too. □

While there are 2^{n_s} possible binary vectors $x^s \in \{0,1\}^{n_s}$, our goal is to exploit Claim 2 to identify a polynomial number of possible candidates. The idea is to use hyperplane arrangements to consider all possible vectors x^s that are compatible with (10). However, the function $h_k(\lambda)$, for every $k \in [n_s]$ depends on $x^1, x^2, \ldots, x^{s-1}$ which amounts to $n_1 + n_2 + \cdots + n_{s-1}$ variables, and this would result in an exponential bound using hyperplane arrangements. The key is to observe that $h_k(\lambda)$, in fact, depends only on the λ_I, for $I \in \mathcal{I}_\beta$, as suggested by our purposely chosen notation.

Consider the hyperplanes H_k, for every $k \in [n_s]$, defined by the algorithm in (6), and let $\mathcal{A}$ be the set of cells of the arrangement of these hyperplanes, as in the algorithm. In the next claim, we show that there is at least one cell in $\mathcal{A}$ which provides a partial solution from an actual optimal solution to Problem F.

Claim 3. *There exists an optimal solution $\mathring{x}$ to Problem F and a cell $C \in \mathcal{A}$ such that, if we denote by $\bar{x}^s \in \{0,1\}^{n_s}$ the partial solution corresponding to cell C defined in (7), we have $\mathring{x}^s = \bar{x}^s$.*

Proof. Let $\breve{x}$ be an optimal solution to Problem F and let $\breve{\lambda}_I$, for every $I \in \mathcal{I}_\beta$, be defined as in Claim 2. Note that $\breve{\lambda}$ is in at least one cell in $\mathcal{A}$, and it may be contained in more than one. Let C be any cell in $\mathcal{A}$ containing $\breve{\lambda}$, and let $\bar{x}^s \in \{0,1\}^{n_s}$ be the partial solution corresponding to C defined in (7). Now we compare $\breve{x}^s$ with $\bar{x}^s$ and show

$$\breve{x}^s_k = \bar{x}^s_k \qquad \text{if } h_k(\breve{\lambda}) \neq 0 \qquad \forall k \in [n_s]. \tag{11}$$

To prove (11), consider separately the cases $h_k(\breve{\lambda}) > 0$ and $h_k(\breve{\lambda}) < 0$. In the first case we have $\breve{x}^s_k = 1$ from (10). Since $\breve{\lambda} \in C$, then $h_k(\lambda) \geq 0$ is valid for C, therefore we have $\bar{x}^s_k = 1$ from (7). In the second case we have $\breve{x}^s_k = 0$ from (10). Since $\breve{\lambda} \in C$, then $h_k(\lambda) \leq 0$ is valid for C, therefore we have $\bar{x}^s_k = 0$ from (7). This concludes the proof of (11).

Let $\mathring{x}$ be the solution to Problem F defined as follows:

$$\mathring{x}^j := \breve{x}^j \qquad \forall j \in [s-1]$$
$$\mathring{x}^s_k := \begin{cases} \breve{x}^s_k & \text{if } h_k(\breve{\lambda}) \neq 0 \\ \bar{x}^s_k & \text{if } h_k(\breve{\lambda}) = 0 \end{cases} \qquad \forall k \in [n_s].$$

From Claim 2, $\mathring{x}$ is also an optimal solution to Problem F. Due to (11) and the definition of $\mathring{x}$, we have $\mathring{x}^s = \bar{x}^s$. □

Consider now the child instance Problem F_C of Problem F corresponding to a cell C from Claim 3. The next simple claim will be useful to conclude the proof of the correctness of our algorithm.

Claim 4. *Let $x^j \in \{0,1\}^{n_j}$ for every $j \in [s-1]$. The objective value of $(x^1, x^2, \ldots, x^{s-1})$ in Problem F_C equals the objective value of $(x^1, x^2, \ldots, x^{s-1}, \bar{x}^s)$ in Problem F.*

Proof. Follows directly from the fact that Problem F_C is obtained from Problem F by fixing x^s to $\bar{x}^s$. □

Let $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1})$ be the solution returned by the algorithm applied to Problem F_C.

Claim 5. *The solution $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1}, \bar{x}^s)$ is optimal to Problem F.*

Proof. By induction, the solution $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1})$ is optimal to Problem F_C. Let $\mathring{x}$ be an optimal solution to Problem F as in Claim 3. Since $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1})$ is optimal to Problem F_C, its objective value in Problem F_C is greater than or equal to the objective value of $(\mathring{x}^1, \mathring{x}^2, \ldots, \mathring{x}^{s-1})$

in Problem F_C. From Claim 4, the objective value of $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1})$ in Problem F_C equals the objective value of $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1}, \bar{x}^s)$ in Problem F. Claim 4 also implies that the objective value of $(\mathring{x}^1, \mathring{x}^2, \ldots, \mathring{x}^{s-1})$ in Problem F_C equals the objective value of $(\mathring{x}^1, \mathring{x}^2, \ldots, \mathring{x}^{s-1}, \bar{x}^s)$ in Problem F. Hence, in Problem F, the objective value of $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1}, \bar{x}^s)$ is greater than or equal to the objective value of $(\mathring{x}^1, \mathring{x}^2, \ldots, \mathring{x}^{s-1}, \bar{x}^s)$. From Claim 3, we have $(\mathring{x}^1, \mathring{x}^2, \ldots, \mathring{x}^{s-1}, \bar{x}^s) = (\mathring{x}^1, \mathring{x}^2, \ldots, \mathring{x}^{s-1}, \mathring{x}^s)$, hence this solution is optimal to Problem F. We then obtain that also $(\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^{s-1}, \bar{x}^s)$ is optimal to Problem F. □

5.1.3 Running time

In this section, we prove the bound on the running time of our algorithm stated in Theorem 2.

Claim 6. *The algorithm presented in Section 5.1.1 performs at most*

$$s\theta^{s-1} n_2^{m_2} n_3^{m_3} \cdots n_s^{m_s} \operatorname{poly}(n_1, n_2, \ldots, n_s, |\mathcal{I}|)$$

arithmetic operations, where θ is a constant.

Proof. In the first iteration, the algorithm constructs at most $\theta n_s^{m_s}$ child instances of the form of Problem F_C in at most $\theta n_s^{m_s}$ arithmetic operations, where θ is a constant [21, 22]. The parameter s associated with each child instance is decreased by one, and the dimensions of the vectors of variables in the new instances are unchanged: $n_1, n_2, \ldots, n_{s-1}$. It follows from the construction of Problem F_C that also the values m_j, for $j \in [s-1]$, associated with the child instances, are unchanged. The total number of problems of the form Problem F considered in the recursive algorithm, and the number of arithmetic operations to construct them, is then at most

$$1 + \theta n_s^{m_s} + \theta^2 n_s^{m_s} n_{s-1}^{m_{s-1}} + \cdots + \theta^{s-1} n_s^{m_s} n_{s-1}^{m_{s-1}} \cdots n_2^{m_2} \leq s\theta^{s-1} n_s^{m_s} n_{s-1}^{m_{s-1}} \cdots n_2^{m_2},$$

where the inequality holds because we can assume, without loss of generality, that $\theta \geq 1$. For each instance considered, the number of remaining arithmetic operations performed by the algorithm is

$$\operatorname{poly}(n_1, n_2, \ldots, n_s, |\mathcal{I}|).$$

□

To conclude the proof of Theorem 2, it suffices to show the following result.

Claim 7. *The size of the numbers produced in the course of the execution of the algorithm presented in Section 5.1.1 is polynomial in the size of the input.*

Proof. In each iteration of the algorithm, the size of the numbers produced by the algorithm is polynomial in the size of the data of the instance of Problem F considered at the beginning of that iteration. This includes the numbers produced by the algorithm in [21, 22] to construct the arrangement of the hyperplanes.

On the other hand, we show that the size of the data of each child instance Problem F_C constructed throughout the execution of the algorithm is polynomial in the size of the data of the original instance of Problem F. In fact, from the construction of Problem F_C, each child instance is obtained from the original instance of Problem F by fixing each component of $x^t, x^{t+1}, \ldots, x^s$, for some $t \in \{2, 3, \ldots, s\}$, to zero or one. As a result, each number appearing in a child instance is obtaining by summing or multiplying polynomially many numbers from the original instance. □

5.2 Corollaries

In this section, we discuss some consequences of Theorem 1, for different optimization problems.

5.2.1 An extension of factorized binary polynomial optimization

First, we remark that our algorithm can be used, with minor modifications, to prove the same result for the slightly more general optimization problem obtained from Problem F by replacing each linear function in the objective with an affine function:

$$\begin{aligned} \max_{x^1,\dots,x^s} \quad & \sum_{I\in\mathcal{I}'} \prod_{j\in I} \left(\langle c^{I,j}, x^j\rangle + d^{I,j}\right) \\ \text{s.t.} \quad & x^j \in \{0,1\}^{n_j} \qquad \forall j \in [s]. \end{aligned} \tag{F$'$}$$

The data of the problem consists of positive integers s and n_j, for $j \in [s]$, a nonempty family $\mathcal{I}'$ of nonempty subsets of $[s]$, vectors $c^{I,j} \in \mathbb{Q}^{n_j}$, and scalars $d^{I,j} \in \mathbb{Q}$, for $I \in \mathcal{I}'$, $j \in I$. We decided to give our algorithm only for Problem F, to avoid overloading the notation, which is already quite heavy. Furthermore, the tractability of Problem F$'$ can be obtained directly from the tractability of Problem F, as we see next.

Corollary 7. *Problem F$'$ can be solved in strongly polynomial time, for any fixed s and $|\mathcal{I}'|$.*

Proof. For every $I \in \mathcal{I}'$, the corresponding product in the objective function, $\prod_{j\in I}\left(\langle c^{I,j}, x^j\rangle + d^{I,j}\right)$, can be expanded to a sum

$$\sum_{I'\in S(I)} \prod_{j\in I} \left\langle c'^{I,j}, x^j \right\rangle + d^I,$$

where $S(I)$ is a set of nonempty subsets of I, $c'^{I,j} \in \mathbb{Q}^{n_j}$ for $j \in I$, and $d^I \in \mathbb{Q}$. Therefore, Problem F$'$ can be written in the form of Problem F with the same parameter s and with $|\mathcal{I}| \le 2^s|\mathcal{I}'|$. The result then follows from Theorem 1. □

5.2.2 Explicit binary polynomial optimization on an s-partite hypergraph

Using Observation 3, we obtain the following corollaries to Theorem 1 for Problem E on an s-partite hypergraph.

Corollary 8. *Consider Problem E on an s-partite hypergraph. This problem can be solved in strongly polynomial time if s is fixed, and all the tensors of the coefficients of the objective function are given through a factorization of the form (1), with r fixed.*

Corollary 9. *Consider Problem E on an s-partite hypergraph. This problem can be solved in strongly polynomial time if s is fixed, and all the tensors of the coefficients of the objective function have fixed rank and rank factorizations are given.*

5.2.3 Binary tensor factorization and rank-1 Boolean tensor factorization

Next, we discuss consequences of Theorem 1 for the binary tensor factorization problem and the rank-1 Boolean tensor factorization problem. Theorem 1, Proposition 4, and Corollary 6 directly imply the following results.

Corollary 10. *The rank-t binary tensor factorization problem on an $n_1 \times n_2 \times \cdots \times n_s$ tensor*

$$A = \sum_{p \in [q]} \left(a^{p,1} \otimes a^{p,2} \otimes \cdots \otimes a^{p,s} \right),$$

where $a^{p,j} \in \mathbb{Q}^{n_j}$, for every $p \in [q]$, $j \in [s]$, can be solved in strongly polynomial time for any fixed q, s, t.

Corollary 11. *The rank-1 Boolean tensor factorization problem on an $n_1 \times n_2 \times \cdots \times n_s$ tensor*

$$A = \sum_{p \in [q]} \left(a^{p,1} \otimes a^{p,2} \otimes \cdots \otimes a^{p,s} \right),$$

where $a^{p,j} \in \mathbb{Q}^{n_j}$, for every $p \in [q]$, $j \in [s]$, can be solved in strongly polynomial time for any fixed q, s.

When the input tensor A has fixed rank and a rank factorization is given, from Corollaries 10 and 11, we obtain Corollaries 1 and 2, stated in Section 1. In the special case $s = 2$, Corollary 2 implies Corollary 3, since a rank factorization of a matrix can be computed in strongly polynomial time via Gaussian elimination [23].

5.2.4 The quadratic case

Regarding quadratic objective functions, Corollary 4 can be obtained easily from Theorem 1 and Observation 3. We give a proof that does not use Observation 3, since the algebra in this special case is significantly simpler.

Proof of Corollary 4. Consider a matrix $Q \in \mathbb{Q}^{m \times n}$. We can compute in strongly polynomial time, via Gaussian elimination [23], a rank factorization of Q, that is, matrices $A \in \mathbb{Q}^{m \times r}$, $B \in \mathbb{Q}^{n \times r}$ such that $Q = AB^{\mathsf{T}}$, where r is the rank of Q [46]. If we denote column i of A by a^i, and column i of B by b^i, for every $i \in [r]$, we can write

$$\begin{aligned}
x^{\mathsf{T}} Q y &= x^{\mathsf{T}} A B^{\mathsf{T}} y \\
&= x^{\mathsf{T}} \left(\sum_{i \in [r]} \left(a^i \otimes b^i \right) \right) y \\
&= x^{\mathsf{T}} \left(\sum_{i \in [r]} \left(a^i b^{i\mathsf{T}} \right) \right) y \\
&= \sum_{i \in [r]} \left(x^{\mathsf{T}} a^i b^{i\mathsf{T}} y \right) \\
&= \sum_{i \in [r]} \left(\langle a^i, x \rangle \langle b^i, y \rangle \right).
\end{aligned}$$

We apply the above argument to each matrix $Q^{i,j} \in \mathbb{Q}^{n_i \times n_j}$, for $i, j \in [s]$ with $i < j$, in the objective function of Problem Q. There are $(s^2 - s)/2$ of these matrices, and we denote by r the maximum of their ranks. Problem Q can then be written in the form of Problem F, where $|\mathcal{I}|$ is at most $r(s^2 - s)/2 + s$. This number is fixed, since by assumption s and r are fixed. The result then follows from Theorem 1. □

6 Comparison with known tractability results

Every instance of Problem F can be reformulated as an instance of Problem E by expanding all products in the objective function; thus we can employ known algorithms for Problem E to solve Problem F. In this section we show that our Theorem 1 does not follow from known tractability results of Problem E. To the best of our knowledge, five main polynomially solvable classes of Problem E on a hypergraph $H = (V, E)$ have been identified so far. These are instances such that:
C1 The objective function is supermodular (see Chapter 45 in [44]);
C2 H is a β-acyclic hypergraph [12, 13, 19] (see also [5, 15, 18]);
C3 H is a cycle hypergraph [11];
C4 The primal treewidth of H is bounded by $\log(\text{poly}(|V|, |E|))$ [3, 8, 36];
C5 The incidence treewidth of H is bounded by $\log(\text{poly}(|V|, |E|))$ [6].

It will suffice to consider the special case of Problem F in Example 1: Problem F-U with $r = 1$, and with each vector $c^{1,j} \in \mathbb{Q}^{n_j}$, for every $j \in [s]$, having all nonzero components. It is also sufficient to consider the case $n_1 = n_2 = \cdots = n_s$, and we set $n := n_1$. Theorem 1 implies that this problem can be solved in strongly polynomial time for any fixed s. As we saw in Example 1, expanding all products in the objective function, the problem takes the form of Problem E-U, where the corresponding hypergraph is s-partite, with sides V^j, for $j \in [s]$, of cardinality n, is s-uniform, and contains all possible n^s edges; We denote this hypergraph by H^s_n.

The following facts imply that even the polynomial solvability of this special case of Problem F, for every $s \geq 2$ fixed, does not follow from any of the known tractable classes C1–C5 above.

Fact 1. *The objective functions of the problems considered in Example 1 are not generally supermodular, for every $s \geq 2$, $n \geq 1$.*

Fact 2. *The hypergraph H^s_n is not β-acyclic, for every $s \geq 2$, $n \geq 2$.*

Fact 3. *The hypergraph H^s_n is not a cycle hypergraph, for every $s \geq 2$, $n \geq 3$.*

Fact 4. *The primal treewidth of H^s_n is $(s-1)n$, for every $s \geq 2$, $n \geq 1$.*

Fact 5. *The incidence treewidth of H^s_n is at least n, for every $s \geq 2$, $n \geq 1$.*

Facts 1 to 3 are easy to verify.

Proof of Fact 1. It suffices to partition the set $[s]$ into nonempty sets S_1, S_2, and construct one solution $\bar{x} = (\bar{x}^1, \bar{x}^2, \ldots, \bar{x}^s)$ with all $\bar{x}^j$, $j \in S^1$, with all components one, and all x^j, $j \in S^2$, with all components zero. The objective values of $\bar{x}$ and $1 - \bar{x}$ are zero, and so is the objective value of the componentwise minimum $\bar{x} \downarrow (1 - \bar{x})$. On the other hand, the objective value of the componentwise maximum $\bar{x} \uparrow (1 - \bar{x})$ is the sum of all edge costs, which can easily be made negative. □

Proof of Fact 2. Let v^j_1, v^j_2 be two distinct nodes in V^j, for every $j \in [s]$. A β-cycle of length four is

$$v^1_1, \left\{v^1_1, v^2_1, \ldots, v^s_1\right\}, v^s_1, \left\{v^1_2, v^2_1, v^3_1, \ldots, v^s_1\right\}, v^1_2, \left\{v^1_2, v^2_2, \ldots, v^s_2\right\}, v^s_2, \left\{v^1_1, v^2_2, v^3_2, \ldots, v^s_2\right\}, v^1_1.$$

□

Proof of Fact 3. For every $s \geq 2$ and $n \geq 3$, in the hypergraph H^s_n, each node is contained in at least three edges. However, in a cycle hypergraph, each node is contained in at most two edges. □

In the remainder of the section, we prove Facts 4 and 5. We start by defining the primal treewidth and the incidence treewidth of a hypergraph $H = (V, E)$. First, we associate two graphs to H. The *primal graph* $G_{\text{prim}}(H)$ of H is defined as the graph whose node set is V and edge set is $\{\{u, v\} : u \neq v,\ \exists e \in E,\ \{u, v\} \in e\}$. Intuitively, the primal graph is obtained by replacing every edge of H by a clique. The *incidence graph* $G_{\text{inc}}(H)$ of H is defined as the bipartite graph whose node set is $V \cup E$ and the edge set is $\{\{v, e\} : v \in V,\ e \in E,\ v \in e\}$. The *primal treewidth* $\text{ptw}(H)$ of H is the treewidth of its primal graph, that is $\text{ptw}(H) = \text{tw}(G_{\text{prim}}(H))$, while the *incidence treewidth* $\text{itw}(H)$ is the treewidth of its incidence graph, that is $\text{itw}(H) = \text{tw}(G_{\text{inc}}(H))$. In the next result, we characterize the treewidth of a *complete s-partite graph,* which is an s-partite graph that contains all possible edges (of cardinality two).

Lemma 1. *Let K be the complete s-partite graph, with $s \geq 2$, and with sides V^j of cardinality n, for $j \in [s]$. Then $\text{tw}(K) = (s-1)n$.*

Proof. It is well-known that the treewidth of K is the minimum size of the largest clique minus one, in a chordal completion of K.

Let $h \in [s]$. Let G be the graph obtained from K by adding, for each $j \in [s] \setminus \{h\}$, all edges between all pairs of nodes in V^j. The graph G is chordal because it has the following perfect elimination ordering: First we list all nodes in V^h in any order, and next all nodes in $V \setminus V^h$ in any order. Since any maximal clique of G consists of all nodes in $V \setminus V^h$ and one node in V^h, the size of the largest clique in G is $(s-1)n + 1$.

Now let $G' = (V, F)$ be a chordal graph containing K. We claim that there is $h \in [s]$ such that, for every $j \in [s] \setminus \{h\}$, F contains all edges between all pairs of nodes in V^j. Assume by contradiction that this does not hold. Then there exist two distinct indices $s, t \in [s]$, two nodes $u_1, u_2 \in V^s$ and two nodes $v_1, v_2 \in V^t$ such that $\{u_1, u_2\}, \{v_1, v_2\} \notin F$. But then u_1, v_1, u_2, v_2, u_1 is a chordless cycle of G' of length 4, a contradiction. This completes the proof of our claim. Hence, G' contains as a subgraph the graph G from the previous paragraph. So the size of the largest clique in G' is at least $(s-1)n + 1$. The treewidth of K is then $(s-1)n$. □

We are now ready to prove Facts 4 and 5.

Proof of Fact 4. The graph $G_{\text{prim}}(H_n^s)$ is the complete s-partite graph with sides V^j of cardinality n, for $j \in [s]$. From Lemma 1, $\text{tw}(G_{\text{prim}}(H)) = (s-1)n$. We obtain $\text{ptw}(H) = \text{tw}(G_{\text{prim}}(H)) = (s-1)n$. □

Proof of Fact 5. Consider the graph $G_{\text{inc}}(H_n^s)$, whose node set is $V \cup E$. Let G' be the minor of $G_{\text{inc}}(H_n^s)$ obtained by deleting all nodes in $V^3 \cup V^4 \cup \cdots \cup V^s$, and then contracting the edges with one node in E and the other node in V^2. G' is then the complete 2-partite graph with sides V^1, V^2. From Lemma 1, $\text{tw}(G') = n$. We obtain $\text{itw}(H_n^s) = \text{tw}(G_{\text{inc}}(H_n^s)) \geq \text{tw}(G') = n$, where the inequality holds because G' is a minor of $G_{\text{inc}}(H_n^s)$ (see, e.g., lemma 14 in [4]). □

The quadratic case. The special case $s = 2$ of Problem Q is the *binary bipartite quadratic optimization* (BQO) problem studied in [40–42]:

$$\begin{aligned} \max_{x^1, x^2} \quad & {x^1}^{\mathsf{T}} Q x^2 + {c^1}^{\mathsf{T}} x^1 + {c^2}^{\mathsf{T}} x^2 \\ \text{s.t.} \quad & x^1 \in \{0,1\}^{n_1},\ x^2 \in \{0,1\}^{n_2}, \end{aligned} \tag{BQO}$$

where $Q \in \mathbb{Q}^{n_1 \times n_2}$, $c^1 \in \mathbb{Q}^{n_1}$, and $c^2 \in \mathbb{Q}^{n_2}$. A polynomial time algorithm for Problem BQO, under the assumption that the rank of Q is fixed, follows by combining the algorithm for the

continuous relaxation of Problem BQO in [32] with the rounding procedure in [41], as observed in [40]. This algorithm is not strongly polynomial, since it needs to solve linear optimization problems. An algorithm that solves Problem BQO in strongly polynomial time if Q has rank one, is given in [40,42]. In [42], the authors also present an algorithm for Problem BQO that is strongly polynomial if the rank of Q is fixed and some "dual non-degeneracy assumptions" are satisfied. As mentioned by the authors, it seems possible to lift these dual non-degeneracy assumptions by constructing an appropriate perturbation of the objective function.

The special case $s = 2$ of our Corollary 4, implies that Problem BQO can be solved in strongly polynomial time if the rank of Q is fixed. Therefore, our result, even in the very special setting, significantly expands previously known results, both in terms of generality, since our graph is s-partite and not just bipartite, and in terms of computational complexity, since our algorithm is strongly polynomial rather than just weakly polynomial.

Funding: Alberto Del Pia is partially funded by AFOSR grant FA9550-23-1-0433. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Air Force Office of Scientific Research.

References

[1] Kim Allemand, Komei Fukuda, Thomas M. Liebling, and Erich Steiner. A polynomial case of unconstrained zero-one quadratic optimization. *Mathematical Programming*, 91:49–52, 2001.

[2] R. Belohlavek, C. Glodeanu, and V. Vychodil. Optimal factorization of three-way binary data using triadic concepts. *Order*, 30(2):437–454, 2013.

[3] Daniel Bienstock and Gonzalo Muñoz. LP formulations for polynomial optimization problems. *SIAM Journal on Optimization*, 28(2):1121–1150, 2018.

[4] Hans L. Bodlaender and Arie M.C.A. Koster. Treewidth computations II. Lower bounds. *Information and Computation*, 209:1103–1119, 2011.

[5] Christoph Buchheim, Yves Crama, and Elisabeth Rodríguez-Heck. Berge-acyclic multilinear $0-1$ optimization problems. *European Journal of Operational Research*, 2018.

[6] Florent Capelli, Alberto Del Pia, and Silvia Di Gregorio. A knowledge compilation take on binary polynomial optimization. *arXiv:2311.00149*, 2024.

[7] Eranda Çela, Bettina Klinz, and Christophe Meyer. Polynomially solvable cases of the constant rank unconstrained quadratic $0-1$ programming problem. *Journal of Combinatorial Optimization*, 12:187–215, 2006.

[8] Yves Crama, Pierre Hansen, and Brigitte Jaumard. The basic algorithm for pseudo-Boolean programming revisited. *Discrete Applied Mathematics*, 29(2–3):171–185, 1990.

[9] Alberto Del Pia. Sparse PCA on fixed-rank matrices. *Mathematical Programming, Series A*, 198:139–157, 2023.

[10] Alberto Del Pia, Santanu S. Dey, and Robert Weismantel. Subset selection in sparse matrices. *SIAM Journal on Optimization*, 30(2):1173–1190, 2020.

[11] Alberto Del Pia and Silvia Di Gregorio. Chvátal rank in binary polynomial optimization. *INFORMS Journal on Optimization*, 3(4):315–349, 2021.

[12] Alberto Del Pia and Silvia Di Gregorio. On the complexity of binary polynomial optimization over acyclic hypergraphs. In *Proceedings of SODA 2022*, pages 2684–2699, 2022.

[13] Alberto Del Pia and Silvia Di Gregorio. On the complexity of binary polynomial optimization over acyclic hypergraphs. *Algorithmica*, 85:2189–2213, 2023.

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

[15] Alberto Del Pia and Aida Khajavirad. The multilinear polytope for acyclic hypergraphs. *SIAM Journal on Optimization*, 28(2):1049–1076, 2018.

[16] Alberto Del Pia and Aida Khajavirad. On decomposability of multilinear sets. *Mathematical Programming, Series A*, 170(2):387–415, 2018.

[17] Alberto Del Pia and Aida Khajavirad. On decomposability of the multilinear polytope and its implications in mixed-integer nonlinear optimization. *INFORMS OS Today*, 8(1):3–10, 2018.

[18] Alberto Del Pia and Aida Khajavirad. The running intersection relaxation of the multilinear polytope. *Mathematics of Operations Research*, 46(3):1008–1037, 2021.

[19] Alberto Del Pia and Aida Khajavirad. A polynomial-size extended formulation for the multilinear polytope of beta-acyclic hypergraphs. *Mathematical Programming, Series A*, 2023.

[20] Alberto Del Pia, Aida Khajavirad, and Nikolaos V. Sahinidis. On the impact of running-intersection inequalities for globally solving polynomial optimization problems. *Mathematical Programming Computation*, 12:165–191, 2020.

[21] Herbert Edelsbrunner. *Algorithms in combinatorial geometry*. Springer, Berlin, 1987.

[22] Herbert Edelsbrunner, J. O'Rourke, and R. Seidel. Constructing arrangements of lines and hyperplanes with applications. *SIAM Journal on Computing*, 15(2):341–363, 1986.

[23] Jack Edmonds. Systems of distinct representatives and linear algebra. *Journal of Research of the National Bureau of Standards – B. Mathematics and Mathematical Physics*, 71B(4):241–245, 1967.

[24] D. Erdos and P. Miettinen. Walk 'n' merge: A scalable algorithm for boolean tensor factorization. *IEEE 13th International Conference on Data Mining*, pages 1037–1042, 2013.

[25] Jean-Albert Ferrez, Komei Fukuda, and Thomas M. Liebling. Solving the fixed rank convex quadratic maximization in binary variables by a parallel zonotope construction algorithm. *European Journal of Operational Research*, 2003.

[26] Michael R. Garey, David S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1(3):237–267, 1976.

[27] Nicolas Gillis and François Glineur. Low-rank matrix approximation with weights or missing data is NP-hard. *SIAM Journal on Matrix Analysis and Applications*, 32:1149–1165, 2011.

[28] Nicolas Gillis and Stephen A. Vavasis. On the complexity of robust pca and ℓ_1-norm low-rank matrix approximation. *Mathematics of Operations Research*, 43(4):1072–1084, 2018.

[29] Peter L. Hammer, Pierre Hansen, Panos M. Pardalos, and David J. Rader. Maximizing the product of two linear functions in $0-1$ variables. *Optimization*, 51:511–537, 2002.

[30] Johan Håstad. Tensor rank is NP-complete. *Journal of Algorithms*, 11(4):644–654, 1990.

[31] Ishay Haviv and Michal Parnas. On the binary and boolean rank of regular matrices. In *47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022)*, 2022.

[32] Milan Hladík, Michal Černý, and Miroslav Rada. A new polynomially solvable class of quadratic optimization problems with box constraints. *Optimization Letters*, 15:2331–2341, 2021.

[33] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, 28:58–81, 2014.

[34] Mehmet Koyutürk, Ananth Grama, and Naren Ramakrishnan. Compression, clustering, and pattern discovery in very high-dimensional discrete-attribute data sets. *IEEE Transactions on Knowledge and Data Engineering*, 17:447–461, 2005.

[35] Mehmet Koyutürk, Ananth Grama, and Naren Ramakrishnan. Nonorthogonal decomposition of binary matrices for bounded error data compression and analysis. *BMC Bioinformatics*, 32:1–9, 2006.

[36] Monique Laurent. Sums of squares, moment matrices and optimization over polynomials. In *Emerging Applications of Algebraic Geometry*, volume 149 of *The IMA Volumes in Mathematics and its Applications*, pages 157–270. Springer, 2009.

[37] Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, Heechang Shin, and Lili Jiang. Weighted rank-one binary matrix factorization. In *Proceedings of the 2011 SIAM International Conference on Data Mining (SDM)*, 2011.

[38] P. Miettinen. Boolean tensor factorizations. *2011 IEEE 11th International Conference on Data Mining*, pages 447–456, 2011.

[39] N. Park, S. Oh, and U. Kang. Fast and scalable distributed boolean tensor factorization. *IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1071–1082, 2017.

[40] Abraham P. Punnen. The bipartite QUBO. In *The Quadratic Unconstrained Binary Optimization Problem*, chapter 10, pages 261–300. Springer, 2022.

[41] Abraham P. Punnen, Piyashat Sripratak, and Daniel Karapetyan. Average value of solutions for the bipartite Boolean quadratic programs and rounding algorithms. *Theoretical Computer Science*, 565:77–89, 2015.

[42] Abraham P. Punnen, Piyashat Sripratak, and Daniel Karapetyan. The bipartite unconstrained $0-1$ quadratic programming problem: Polynomially solvable cases. *Discrete Applied Mathematics*, 193:1–10, 2015.

[43] T. Rukat, C. Holmes, and Yau C. Probabilistic boolean tensor decomposition. *Proceedings of the 35th International Conference on Machine Learning (PMLR)*, 80:4413–4422, 2018.

[44] Alexander Schrijver. *Combinatorial Optimization. Polyhedra and Efficiency.* Springer-Verlag, Berlin, 2003.

[45] Bao-Hong Shen, Shuiwang Ji, and Jieping Ye. Mining discrete patterns via binary matrix factorization. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 757–765, 2009.

[46] G.W. Stewart. *Matrix algorithms: volume 1: basic decompositions.* SIAM, 1998.

[47] C. Wan, W. Chang, T. Zhao, M. Li, S. Cao, and C. Zhang. Fast and efficient boolean matrix factorization by geometric segmentation. *AAAI 2020*, 2020.