

On the impact of running intersection inequalities for globally solving polynomial optimization problems *

Alberto Del Pia [†] Aida Khajavirad [‡] Nikolaos Sahinidis [§]

June 29, 2018

Abstract

We consider global optimization of nonconvex problems whose factorable reformulations contain a collection of multilinear equations of the form $z_e = \prod_{v \in e} z_v$, $e \in E$, where E denotes a set of subsets of cardinality at least two of a ground set. Important special cases include multilinear and polynomial optimization problems. The multilinear polytope is the convex hull of a set of binary points z satisfying a number of multilinear equations. In [16], the authors introduce running intersection inequalities, a family of facet-defining inequalities for the multilinear polytope. In this paper we address the separation problem for this class of inequalities. We first prove that separating flower inequalities, a subclass of running intersection inequalities, is NP-hard. Subsequently, for multilinear polytopes of fixed degree, we devise an efficient polynomial-time algorithm for separating running intersection inequalities and embed the proposed cutting-plane generation scheme at every node of the branch-and-reduce global solver **BARON**. We carry out an extensive computational study on multilinear and polynomial optimization problems of degree three and four. Results show that running intersection cuts significantly improve the performance of **BARON** and lead to an average 50% CPU time reduction.

Key words: branch-and-cut; polynomial optimization; running-intersection inequalities; multilinear polytope; separation algorithm; mixed-integer nonlinear optimization

1 Introduction

Constructing strong and cheap to compute convex relaxations is key to the efficiency of global optimization algorithms. Factorable programming is a widely used

*The first two authors have been partially supported 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

[§]Department of Chemical Engineering, Carnegie Mellon University. E-mail: sahinidis@cmu.edu

methodology in global optimization of mixed-integer nonlinear optimization problems (MINLPs) for bounding general nonconvex functions [26]. These techniques iteratively decompose a factorable function, through the introduction of auxiliary variables and constraints for intermediate nonlinear expressions, until each intermediate expression can be convexified effectively. Current general-purpose global solvers such as BARON [24], COUENNE [5], LindoGlobal [25], SCIP [6] and ANTIGONE [29], rely on factorable programming bounds. Hence, improving the quality of these relaxations has an immediate impact on our ability to solve nonconvex problems to global optimality.

Multilinear sets and polytopes. Factorable reformulations of many types of MINLPs, such as mixed-integer polynomial optimization problems, contain a collection of multilinear equations of the form $z_e = \prod_{v \in e} z_v$, $e \in E$, where E denotes a set of subsets of cardinality at least two of a ground set V . Let us define the set of points satisfying all multilinear equations present in a factorable reformulation of a MINLP as

$$\tilde{\mathcal{S}} = \{z \in \mathbb{R}^{V+E} : z_e = \prod_{v \in e} z_v \forall e \in E, z_v \in [0, 1] \forall v \in V_1, z_v \in \{0, 1\} \forall v \in V_2\}, \quad (1)$$

where V_1 and V_2 form a partition of V and are the index sets corresponding to continuous and binary variables, respectively. Without loss of generality, we can assume that the continuous variables are defined over the unit hypercube; otherwise, one can obtain a set of the form $\tilde{\mathcal{S}}$ after some scaling and shifting of variables. It then follows that the convex hull of $\tilde{\mathcal{S}}$ is a polytope and the projection of its vertices onto the space of the variables z_v , $v \in V$, is given by $\{0, 1\}^V$ (see for example [39]). Consequently, the facial structure of the convex hull of $\tilde{\mathcal{S}}$ can be equivalently studied by considering the following binary set:

$$\left\{ z \in \{0, 1\}^{V+E} : z_e = \prod_{v \in e} z_v \forall e \in E \right\}. \quad (2)$$

There is a one-to-one correspondence between sets of form (2) and hypergraphs $G = (V, E)$ [13]. Henceforth we refer to (2) as the *multilinear set* of G and denote it by \mathcal{S}_G , and refer to its convex hull as the *multilinear polytope* of G and denote it by MP_G . If all multilinear equations defining \mathcal{S}_G are bilinear, the multilinear polytope coincides with the Boolean quadric polytope defined by Padberg [31] in the context of 0–1 quadratic optimization. Over the past three decades, a significant amount of research has been devoted to studying the facial structure of the Boolean quadric polytope and these theoretical findings have had a significant impact on the performance of branch-and-cut based algorithms for mixed-integer quadratic optimization problems (MIQCPs) [3, 46, 21, 7]. In contrast, as we detail next, for higher degree multilinear polytopes, similar polyhedral studies are rather scarce.

The explicit characterization of the multilinear polytope of the complete hypergraph, that is, the hypergraph whose edge set consists of all subsets of nodes of cardinality at least two, has been presented in several studies (see for example [37]). In [13] the authors study the facial structure of the multilinear polytope of general hypergraphs and develop the theory of a number of lifting operations, giving rise to various types of facet-defining inequalities. In [14], the authors study the decomposability properties of the multilinear polytope. Namely, they derive necessary and sufficient conditions under which we have $\text{MP}_G = \bigcap_{j \in J} \text{MP}_{G_j}$, where each hypergraph G_j has a much simpler structure than the original hypergraph G . In [15, 16], the authors study the complexity of the facet-description of the multilinear polytope in conjunction with the acyclicity degree of the underlying hypergraph. Explicit characterizations of the multilinear polytope of Berge-acyclic and γ -acyclic hypergraphs are provided in [15]. Moreover, as a byproduct, [15] introduces flower inequalities, a family of valid inequalities for the multilinear polytope, which generalize 2-link inequalities defined in [12]. Subsequently, in [16], the authors introduce running intersection inequalities, a significant generalization of flower inequalities and identify sufficient conditions under which the proposed inequalities are facet-defining. Moreover, they show that the polytope obtained by adding all running intersection inequalities to a well-known relaxation of the multilinear set coincides with the multilinear polytope of a large subclass of β -acyclic hypergraphs.

Quadratization approaches. In order to capitalize on existing algorithms for 0–1 quadratic optimization, various techniques have been developed to reduce higher degree binary polynomial optimization problems to quadratic ones, at the cost of an increase in the number of variables. These methods are often referred to as quadratization approaches. In [1], the authors perform a systematic study of the existing quadratization techniques. They provide tight lower and upper bounds on the number of added variables in the worst case. In particular, they show that, for each fixed degree d , there are polynomials of n variables and of degree d for which every quadratization must involve at least $\Omega(n^{d/2})$ added variables. In [8], the authors demonstrate the usefulness of quadratization in some computer vision applications. In [9], the authors propose a different type of quadratization scheme that is most effective when the original multilinear set is reducible; that is, every set $e \in E$ is a union of two other sets in E . Otherwise, auxiliary variables are added to the model to make \mathcal{S}_G reducible.

Convex and concave envelopes of multilinear functions. There has been a stream of research in the global optimization literature on characterizing the convex/concave envelope of a multilinear function defined as $L(z) = \sum_{e \in E} c_e \prod_{v \in e} z_v$, where $c_e \in \mathbb{R} \setminus \{0\}$ for all $e \in E$ and $z_v \in [l_v, u_v]$ such that $-\infty < l_v < u_v < \infty$ for all $v \in V$. Since multilinear functions are closed under scaling and shifting of variables, without loss of generality one can assume that $l_v = 0$ and $u_v = 1$ for all $v \in V$.

It then follows that an extended formulation for the convex hull of a multilinear function (and hence its convex and concave envelopes) is given by the image of the multilinear polytope MP_G under the linear mapping $(z_v, z_e) \rightarrow (z_v, \sum_{e \in E} c_e z_e)$. For a few classes of structured multilinear functions, explicit characterizations of the envelopes are available. These results address trilinear terms over a box [27, 28], special forms of bilinear functions over the unit hypercube [33], special forms of multilinear functions over the unit hypercube and some discrete sets [36], bilinear and polynomial covering sets with certain sign restrictions and no upper bounds on variables [40], and submodular functions over various polyhedral subdivisions of a box [41].

For bounding a general multilinear function, however, a common practice is to utilize a termwise scheme in which each multilinear term is relaxed by a recursive application of bilinear envelopes [26, 34, 42]. This termwise factorable scheme is simple to implement and has been deployed in all general-purpose global solvers; however, in general, it leads to very poor bounds. As a notable exception, in [2], the authors propose a cutting plane generation framework in which certain facets corresponding to the convex hull of a multilinear function are generated by solving a linear optimization problem (LP) and are added at every node of the global solver **BARON**. However, the size of this LP grows exponentially with the number of variables in the multilinear function. To control the size of the LP, the authors present a decomposition algorithm to break down a multilinear function into a collection of lower-dimensional multilinear functions. In addition, they devise a customized simplex algorithm for solving the separation problem that outperforms the state-of-the-art LP solvers by several orders of magnitude. Their computational study on various sets of multilinear and polynomial optimization problems show that the proposed cuts lead to significant reductions in CPU time and enable **BARON** to solve many more problems to global optimality.

Our contribution. In this paper, we study the separation problem for running intersection inequalities [16], a family of valid inequalities for the multilinear polytope. In [15] the authors consider flower inequalities, a subclass of running intersection inequalities, and show that, over γ -acyclic hypergraphs, the separation problem can be solved in strongly polynomial-time; i.e., in a number of iterations bounded by a polynomial in $|V|$ and $|E|$. They also mention that, over balanced hypergraphs, a generalization of γ -acyclic hypergraphs, separation of flower inequalities can be done in polynomial-time by solving an LP. However, in almost all MINLPs that appear in applications, the corresponding hypergraphs are not balanced. In this paper, we prove that the separation problem for flower inequalities over general hypergraphs is NP-hard. Subsequently, we consider hypergraphs with a fixed rank, where the rank r of G is defined as the maximum cardinality of an edge in E . This assumption is based on the observation that, for the multilinear sets that appear in MINLPs, we often have $r \ll |V|$. In fact, for all practical purposes we can assume that $r < 10$ and

therefore it is reasonable to assume that r is a fixed parameter. We then show that if G has a fixed rank, the separation problem for running intersection inequalities can be solved in $O(|E|^2)$ operations.

To exploit the local information regarding bounds on variables for cut generation, as well as to utilize running intersection inequalities for local feasibility-based and optimality-based range reductions, we devise an efficient implementation of the separation algorithm within a branch-and-cut framework. Our proposed cut generation algorithm is embedded at every node of **BARON**'s branch-and-reduce algorithm. Extensive computational results are presented for globally solving multilinear and polynomial optimization problems of degree three and four. Results show that the incorporation of running intersection cuts in **BARON** reduces the average CPU time and number of nodes in the search tree by 50% and 75%, respectively.

Organization. In Section 2, we provide a brief overview of running intersection and flower inequalities. In Section 3, we show that the separation problem for flower inequalities over general hypergraphs is NP-hard. Subsequently, in Section 4, we consider fixed-rank hypergraphs and present an efficient polynomial-time separation algorithm for running intersection inequalities. With the objective of embedding the proposed separation algorithm in a branch-and-cut framework, in Section 5, we present an enhanced implementation that conducts most of the expensive computations at the root node. Finally, computational results on a variety of multilinear and polynomial optimization problems are presented in Section 6.

2 Running intersection inequalities

In [16] the authors introduce running intersection inequalities, a class of valid inequalities for the multilinear polytope. For a self-contained exposition, in the following we briefly review some of the key concepts related to these inequalities which will be used for the subsequent developments.

To define running intersection inequalities, we make use of the notion of running intersection property, which was first introduced in the database community to study acyclic databases [4]. A set F of subsets of a finite set V has the *running intersection property* if there exists an ordering p_1, p_2, \dots, p_m of the sets in F such that

$$\text{for each } k = 2, \dots, m, \text{ there exists } j < k \text{ such that } p_k \cap \left(\bigcup_{i < k} p_i \right) \subseteq p_j. \quad (3)$$

Throughout the paper, we refer to an ordering p_1, p_2, \dots, p_m satisfying (3) as a *running intersection ordering* of F . Each running intersection ordering p_1, p_2, \dots, p_m of F induces a collection of sets

$$N(p_1) := \emptyset, \quad N(p_k) := p_k \cap \left(\bigcup_{i < k} p_i \right) \text{ for } k = 2, \dots, m. \quad (4)$$

Definition 1. Consider a hypergraph $G = (V, E)$. Let $e_0 \in E$ and let $e_k, k \in K$, be a collection of edges in E satisfying the following conditions:

- (i) $|e_0 \cap e_k| \geq 2$ for all $k \in K$,
- (ii) $e_0 \cap e_k \not\subseteq e_0 \cap e_{k'}$ for any $k, k' \in K$,
- (iii) the set $\tilde{E} := \{e_0 \cap e_k : k \in K\}$ has the running intersection property.

Consider a running intersection ordering of \tilde{E} with the corresponding sets $N(e_0 \cap e_k)$, for $k \in K$, as defined in (4). For each $k \in K$ with $N(e_0 \cap e_k) \neq \emptyset$, let w_k be a node in $N(e_0 \cap e_k)$. We define a running intersection inequality as

$$- \sum_{k \in K: N(e_0 \cap e_k) \neq \emptyset} z_{w_k} + \sum_{v \in e_0 \setminus \bigcup_{k \in K} e_k} z_v + \sum_{k \in K} z_{e_k} - z_{e_0} \leq \omega - 1, \quad (5)$$

where

$$\omega = \left| e_0 \setminus \bigcup_{k \in K} e_k \right| + \left| k \in K : N(e_0 \cap e_k) = \emptyset \right|.$$

We refer to e_0 as the center and to $e_k, k \in K$, as the neighbors.

The above definition for running intersection inequalities depends on a running intersection ordering of the set \tilde{E} defined in condition (iii). It can be shown that such an ordering is, in general, not unique. However, in [13], the authors prove that the system of all running intersection inequalities centered at e_0 with neighbors $e_k, k \in K$, is independent of the running intersection ordering. Moreover, they present sufficient conditions under which running intersection inequalities define facets of the multilinear polytope MP_G .

In the special case where $N(e_0 \cap e_k) = \emptyset$ for all $k \in K$; i.e., $e_0 \cap e_k \cap e_{k'} = \emptyset$ for all distinct $k, k' \in K$, running intersection inequalities simplify to flower inequalities introduced in [15]. The *flower inequality* centered at e_0 with the neighbors $e_k, k \in K$ is given by:

$$\sum_{v \in e_0 \setminus \bigcup_{k \in K} e_k} z_v + \sum_{k \in K} z_{e_k} - z_{e_0} \leq |e_0 \setminus \bigcup_{k \in K} e_k| + |K| - 1. \quad (6)$$

In this paper, we are interested in utilizing running intersection inequalities in a branch-and-cut framework. As we detailed in Section 1, given a collection of multilinear terms corresponding to a factorable reformulation of a MINLP, one can obtain a set of the form (1) after possibly scaling and shifting some of the continuous variables. In a branch-and-cut framework, the variables bounds change in different parts of the tree. Clearly, running intersection cuts corresponding to the variable bounds at the root node remain valid throughout the search tree. However, for cut generation, it is highly beneficial to exploit tighter variable bounds at each node of the tree. To this end, at each node, one can construct the set $\tilde{\mathcal{S}}$ defined by (1) using

local bounds on variables and generate cutting planes accordingly. However, such an approach is fairly expensive as the structure of the hypergraph may change as a result of each branching or range reduction. Hence, to address this trade-off, in this paper, we pursue a different alternative. Define

$$\mathcal{S}_G^b = \left\{ z \in \mathbb{R}^{V+E} : z_e = \prod_{v \in e} z_v \ \forall e \in E, \ z_v \in [l_v, u_v], \ \forall v \in V \right\},$$

where $0 \leq l_v < u_v < \infty$ for all $v \in V$. Define $\tilde{z}_v = z_v/u_v$ for all $v \in V$ and $\tilde{z}_e = \prod_{v \in e} \tilde{z}_v$ for all $e \in E$. Let $u_e = \prod_{v \in e} u_v$. It then follows that $\tilde{z}_e = z_e/u_e$ for all $e \in E$. Hence, the following inequality is valid for \mathcal{S}_G^b :

$$- \sum_{k \in K: N(e_0 \cap e_k) \neq \emptyset} \frac{\tilde{z}_{w_k}}{u_{w_k}} + \sum_{v \in e_0 \setminus \bigcup_{k \in K} e_k} \frac{\tilde{z}_v}{u_v} + \sum_{k \in K} \frac{\tilde{z}_{e_k}}{u_{e_k}} - \frac{\tilde{z}_{e_0}}{u_{e_0}} \leq \omega - 1, \quad (7)$$

where ω is as defined in (5).

3 Separation of flower inequalities over general hypergraphs

In [15] the authors consider flower inequalities as defined in (6) and show that, over γ -acyclic hypergraphs, the separation problem can be solved in strongly polynomial-time. They also mention that over balanced hypergraphs, a generalization of γ -acyclic hypergraphs, separation of flower inequalities can be done in polynomial-time by solving an LP. However, in almost all MINLPs that appear in applications, the corresponding hypergraphs are not balanced. In this section, we prove that separating flower inequalities over general hypergraphs is NP-hard. First, we formally define the separation problem for a general family of inequalities (see [35] for more details).

The separation problem: Given a hypergraph $G = (V, E)$, a family of inequalities valid for MP_G , and a vector $\bar{z} \in [0, 1]^{V+E}$, decide whether \bar{z} satisfies all the inequalities in the family or not, and in the latter case, find an inequality in the family that is violated by \bar{z} .

Before stating our main result, let us recall a widely-used polyhedral relaxation of \mathcal{S}_G that is obtained by replacing each multilinear term $z_e = \prod_{v \in e} z_v$, by its convex hull over the unit hypercube:

$$\begin{aligned} \text{MP}_G^{\text{LP}} = \left\{ z : \right. & z_v \leq 1, \ \forall v \in V, \\ & z_e \geq 0, \ z_e \geq \sum_{v \in e} z_v - |e| + 1, \ \forall e \in E, \\ & \left. z_e \leq z_v, \ \forall v \in e, \ \forall e \in E \right\}. \end{aligned} \quad (8)$$

The above relaxation is often referred to as the *standard linearization* of the multi-linear set [11]. For a rank- r hypergraph $G = (V, E)$, the system defining MP_G^{LP} has at most $|V| + r|E|$ inequalities. Hence, for our computational complexity analysis, it is reasonable to assume that the point \bar{z} is present in MP_G^{LP} , as otherwise, one can easily identify an inequality in system (8) that is violated by \bar{z} .

Theorem 1. *Given a hypergraph \bar{G} and a vector $\bar{z} \in \text{MP}_{\bar{G}}^{\text{LP}}$, the following problems are NP-hard:*

(P1) *Is there a flower inequality for \bar{G} centered at an edge e_0 of $E(\bar{G})$ violated by \bar{z} ?*

(P2) *Is there a flower inequality for \bar{G} violated by \bar{z} ?*

Proof. To prove the theorem we need to give some definitions. A *matching* in a hypergraph $G = (V, E)$ is a subset M of E with the property that $e \cap f = \emptyset$ for all $e, f \in M$ with $e \neq f$. A matching in G is called *perfect* if each node in V is contained in exactly one edge of the matching. We say that a hypergraph G is *3-uniform* if each edge contains exactly three nodes.

In order to show that Problems (P1) and (P2) are NP-hard, we give polynomial reductions from the 3-Dimensional Matching Problem (3-DMP). We refer the reader to [23] for the definition and NP-hardness of the 3-DMP. The 3-DMP is a special case of the following problem, which is therefore NP-hard as well.

(P3) Given a 3-uniform hypergraph $G = (V, E)$, is there a matching M of G with $|M| = \frac{|V|}{3}$?

Let $G = (V, E)$ be a 3-uniform hypergraph with a number of nodes divisible by 3 and let $n := |V|$.

Reduction to perfect matching with costs. We define the set of loops $L := \{\{v\} : v \in V\}$, and let $\tilde{G} := (V, E \cup L)$. In the following, for ease of notation, we often identify a loop $\{v\}$ with the node v . Let $\alpha := \frac{3}{n+1}$. Since $n \geq 3$, α satisfies $0 < \alpha < 1$. We define the *cost* of each element of $E(\tilde{G})$ as

$$\begin{aligned} w_e &:= \alpha & \forall e \in E \\ w_v &:= \alpha & \forall v \in L. \end{aligned}$$

Accordingly, we define the *cost* of each subset \tilde{M} of $E \cup L$ as

$$c(\tilde{M}) := \sum_{v \in \tilde{M} \cap L} c_v + \sum_{e \in \tilde{M} \cap E} c_e = \alpha |\tilde{M}|.$$

We now show that Problem (P3) is equivalent to the following problem:

(P4) Is there a perfect matching \tilde{M} of \tilde{G} of cost $c(\tilde{M}) < 1$?

There is a natural bijection between matchings in G and perfect matchings in \tilde{G} . Namely, given a perfect matching \tilde{M} of \tilde{G} , the set M obtained from \tilde{M} by deleting all loops is a matching of G . Conversely, given a matching M of G , the set $\tilde{M} := M \cup \{\{v\} : v \notin \cup_{e \in M} e\}$ is a perfect matching of \tilde{G} . Since all edges of G contain three nodes, for a matching M of G and its corresponding perfect matching \tilde{M} of \tilde{G} we have

$$|\tilde{M}| = |M| + (n - 3|M|) = n - 2|M|.$$

Hence,

$$c(\tilde{M}) = \alpha|\tilde{M}| = \alpha(n - 2|M|).$$

This implies that $|M| = \frac{n}{3}$ if and only if $c(\tilde{M}) = \alpha(n - 2\frac{n}{3}) = \frac{n}{n+1}$, where we used the definition of α . Now consider a matching M of G that satisfies $|M| < \frac{n}{3}$. Then $|M| \leq \frac{n-3}{3}$ and this happens if and only if $c(\tilde{M}) \geq \alpha(n - 2\frac{n-3}{3}) = \frac{n+6}{n+1}$. This shows that there is a matching M of G with $|M| = \frac{n}{3}$ if and only if there is a perfect matching \tilde{M} of \tilde{G} of cost $c(\tilde{M}) < 1$.

Reduction to Problem (P1). Let $\tilde{G} := (V, E \cup e_0)$, where $e_0 := V$. We define the vector $\bar{z} \in [0, 1]^{V \cup E(\tilde{G})}$ as follows:

$$\begin{aligned} \bar{z}_{e_0} &:= 0 \\ \bar{z}_e &:= 1 - \alpha && \forall e \in E \\ \bar{z}_v &:= 1 - \alpha && \forall v \in V. \end{aligned}$$

First, we show that the vector \bar{z} is in the standard linearization $\text{MP}_{\tilde{G}}^{\text{LP}}$. Using the definition of \bar{z} , we can rewrite the inequalities in the definition of $\text{MP}_{\tilde{G}}^{\text{LP}}$ with $z = \bar{z}$. The inequalities corresponding to the nodes of \tilde{G} are:

$$\alpha \geq 0 \quad \forall v \in V. \quad (9)$$

The inequalities corresponding to an edge $e \in E$ are:

$$\alpha \leq 1 \quad \forall e \in E \quad (10)$$

$$\alpha \leq \sum_{v \in e} \alpha \quad \forall e \in E \quad (11)$$

$$\alpha \geq \alpha \quad \forall v \in e, \forall e \in E. \quad (12)$$

Finally, the inequalities corresponding to edge e_0 are:

$$1 \leq 1 \quad (13)$$

$$1 \leq \sum_{v \in V} \alpha \quad (14)$$

$$1 \geq \alpha \quad \forall v \in V. \quad (15)$$

We now show that all inequalities (9)–(15) are satisfied. Clearly, inequalities (12) and (13) are satisfied. Since $0 < \alpha < 1$, it follows that (9), (10), and (15) are also satisfied. Since each edge $e \in E$ contains three nodes, we have $\sum_{v \in e} \alpha = 3\alpha > \alpha$, implying (11) is satisfied. Finally, $\sum_{v \in V} \alpha = n\alpha = \frac{3n}{n+1} > 1$, and thus (14) also is satisfied.

We now prove that Problem (P4) is equivalent to Problem (P1) with the defined \bar{G} , e_0 , and \bar{z} . To this end, we show that there exists a flower inequality for \bar{G} centered at e_0 violated by \bar{z} if and only if there is a perfect matching \tilde{M} of \bar{G} of cost $c(\tilde{M}) < 1$.

Every edge $e \in E$ is adjacent to e_0 and satisfies $|e_0 \cap e| = 3 \geq 2$. Therefore, there exists a flower inequality for \bar{G} centered at e_0 violated by \bar{z} if and only if there exist a nonempty collection of edges e_k , $k \in K$, in E with $e_k \cap e_{k'} = \emptyset$ for all distinct $k, k' \in K$, such that

$$\sum_{v \in e_0 \setminus \cup_{k \in K} e_k} \bar{z}_v + \sum_{k \in K} \bar{z}_{e_k} - \bar{z}_{e_0} > |e_0 \setminus \cup_{k \in K} e_k| + |K| - 1,$$

or, equivalently,

$$\sum_{v \in e_0 \setminus \cup_{k \in K} e_k} (1 - \bar{z}_v) + \sum_{k \in K} (1 - \bar{z}_{e_k}) < 1 - \bar{z}_{e_0}.$$

Using the definition of \bar{z} and c , the latter reduces to

$$\sum_{v \in e_0 \setminus \cup_{e \in K} e} c_v + \sum_{e \in K} c_e < 1. \quad (16)$$

Assume now that there exists a nonempty collection of edges e_k , $k \in K$, with $e_k \cap e_{k'} = \emptyset$ for all distinct $k, k' \in K$, that satisfies (16). Define $\tilde{M} := K \cup \{\{v\} : v \notin \cup_{k \in K} e_k\}$. Then \tilde{M} is a perfect matching of \bar{G} with $c(\tilde{M}) < 1$. Conversely, assume that \tilde{M} is a perfect matching of \bar{G} with $c(\tilde{M}) < 1$. Then, the set e_k , $k \in K$, obtained from \tilde{M} by deleting all loops is a nonempty collection of edges with $e_k \cap e_{k'} = \emptyset$ for all distinct $k, k' \in K$, that satisfies (16).

This completes the proof that Problem (P1) is NP-hard.

Reduction to Problem (P2). Consider a flower inequality for \bar{G} centered at an edge $e \in E$. We show that this flower inequality is always valid at $z = \bar{z}$. Note that we must have $|K| = 1$, thus we denote by k the only index in K . There are two possibilities: either e_k is an edge in E with $|e \cap e_k| = 2$, or $e_k = e_0$. In the first case, the corresponding flower inequality is

$$z_v + z_{e_k} - z_e \leq 1,$$

where v is the unique node in $e \setminus e_k$. Such an inequality is always satisfied by \bar{z} because $\bar{z}_v = \bar{z}_{e_k} = \bar{z}_e = \alpha$, and $\alpha < 1$. In the second case, the corresponding flower inequality is

$$z_{e_0} - z_e \leq 0.$$

This inequality is always satisfied at \bar{z} since $\bar{z}_{e_0} = 0$, $\bar{z}_e = \alpha$, and $\alpha > 0$.

This implies that there exists a flower inequality for \bar{G} violated by \bar{z} if and only if there exists a flower inequality for \bar{G} centered at e_0 violated by \bar{z} . This shows that Problem (P2) is NP-hard as well. \square

Theorem 1 and the polynomial-time equivalence of separation and optimization directly imply the following corollary.

Corollary 1. *Given a hypergraph \bar{G} and a vector $\bar{z} \in MP_{\bar{G}}^{LP}$, the following two problems are NP-hard:*

- (i) *The separation problem over all flower inequalities centered at an edge e_0 of \bar{G} ,*
- (ii) *The separation problem over all the flower inequalities of \bar{G} .*

4 Separation of running intersection inequalities over fixed-rank hypergraphs

The NP-hardness results of Section 3 are based on the assumption that, given a multilinear set \mathcal{S}_G , the rank r of the hypergraph $G = (V, E)$ is a problem input. However, for multilinear sets that appear in MINLPs we often have $r \ll |V|$. In fact, for all practical purposes, we can assume that $r < 10$ and therefore it is reasonable to assume that r is a fixed parameter. In this section, we consider the separation problem over all running intersection inequalities. As we detail in the following, if r is fixed, this separation problem can be solved in strongly polynomial-time, i.e., in a number of iterations bounded by a polynomial in $|V|$ and $|E|$.

For a general hypergraph $G = (V, E)$, the number of all running intersection inequalities is often very large. In the following, we describe a preprocessing step for our separation algorithm that enables us to solve the separation problem over a small subset of running intersection inequalities. In the sequel, we refer to such inequalities as the *separation system*. As we detail later, this step is an essential part of our separation algorithm as it significantly reduces both time and memory requirements. Let $\bar{z} \in [0, 1]^{V+E}$. Let e_0 denote an edge in E , denote by E_{e_0} the subset of E containing, for every $f \subseteq e_0$ with $|f| \geq 2$, among all the edges $e \in E \setminus \{e_0\}$ with $e_0 \cap e = f$, only one that maximizes \bar{z}_e . The set \mathcal{I}_{e_0} is then defined as the set of running intersection inequalities with center e_0 , and neighbors $e_k \in E_{e_0}$, for $k \in K$, such that for each $k \in K$, the node w_k is chosen as one in $N(e_0 \cap e_k)$ that minimizes

\bar{z}_{w_k} . Finally, the separation system \mathcal{I} is defined as $\mathcal{I} := \cup_{e_0 \in E} \mathcal{I}_{e_0}$. Given a running intersection inequality (5) and a vector $\bar{z} \in [0, 1]^{V+E}$, we define the *violation of the inequality by \bar{z}* as the difference between the left-hand side of (5) evaluated in \bar{z} and its right-hand side. The following proposition establishes the key property of running intersection inequalities that are present in the separation system.

Proposition 1. *Let $G = (V, E)$ be a hypergraph and let $\bar{z} \in [0, 1]^{V+E}$. Consider a running intersection inequality centered at e_0 with neighbors $e_k, k \in K$. Then \mathcal{I}_{e_0} contains a running intersection inequality centered at e_0 with neighbors $e'_k, k \in K$, such that $e_0 \cap e_k = e_0 \cap e'_k$ for all $k \in K$ whose violation by \bar{z} is at least as large as the violation of the original inequality by \bar{z} .*

Proof. Consider a running intersection inequality (5) centered at e_0 with neighbors $e_k, k \in K$. For each $k \in K$ with $N(e_0 \cap e_k) \neq \emptyset$, let w_k be the node selected from $N(e_0 \cap e_k)$ to construct this inequality. For each $k \in K$, we define the new neighbor e'_k as an edge e that maximizes \bar{z}_e among all edges in $e \in E$ with $e_0 \cap e = e_0 \cap e_k$. By construction, such an edge e'_k is always present in E_{e_0} . For each $k \in K$ with $N(e_0 \cap e_k) \neq \emptyset$, we define the node u'_k as a node u in $N(e_0 \cap e_k)$ that minimizes \bar{z}_u . We now obtain a new running intersection inequality from the original one by replacing each neighbor e_k with the corresponding e'_k and by replacing each node w_k with the corresponding u'_k . The new inequality is a running intersection inequality as for each edge $k \in K$ we have $e_0 \cap e_k = e_0 \cap e'_k$, implying $N(e_0 \cap e_k) = N(e_0 \cap e'_k)$. Moreover, from the definition of the separation system it follows that this inequality is present in \mathcal{I}_{e_0} . Clearly, the left-hand side of the new inequality evaluated at \bar{z} is not smaller than the left-hand side of the original running intersection inequality evaluated at \bar{z} . Since the two inequalities have the same right-hand side, it follows that the violation of the new inequality by \bar{z} is at least as large as the violation of the original inequality by \bar{z} . \square

Proposition 1 implies that the separation problem over all running intersection inequalities can be equivalently solved over the separation system. More precisely, if there exists a running intersection inequality that is violated by \bar{z} in the original system, then there also exists an inequality in the separation system that is violated by \bar{z} . We say that a running intersection inequality is *maximally violated by \bar{z}* if the violation of the inequality by \bar{z} is maximum among all running intersection inequalities. Then, Proposition 1 implies that the separation system contains a running intersection inequality that is maximally violated by \bar{z} . As we detail later, the computational cost of the separation problem over all running intersection inequalities depends on the maximum number of neighbours present in running intersection inequalities. Hence, to control the cost of the separation algorithm, we now consider the problem of separating \bar{z} over all running intersection inequalities with at most q neighbors. We refer to this restricted separation problem as *the q -separation problem*.

From Proposition 1, it follows that the q -separation problem can be equivalently solved over inequalities with at most q neighbors that are present in the separation system. We refer to the set of all such inequalities as the q -separation system. Throughout this section, we assume that a rank- r hypergraph $G = (V, E)$ is represented by an incidence-list in which edges are stored as objects, and every edge stores the vertices it contains. In order to use efficient search algorithms, we assume that the vertex list for each edge is sorted. Otherwise, such a sorted data structure can be obtained in $O(r|E|)$ operations by using some integer sorting algorithm such as counting sort.

Proposition 2. *Given a rank- r hypergraph $G = (V, E)$ and a vector $\bar{z} \in [0, 1]^{V+E}$, there exists an algorithm that solves the q -separation problem in $O(|E|(r2^r|E| + 2^{r^q}q^2r))$ operations.*

Proof. Let e_0 be an edge in E . We show how to solve the separation problem over all inequalities in the q -separation system that are centered at e_0 . By applying the algorithm $|E|$ times, we can then solve the q -separation problem.

Let the sets \mathcal{I}_{e_0} and E_{e_0} be as defined before. Denote by $\mathcal{I}_{e_0}^q$ the set of all inequalities in \mathcal{I}_{e_0} with at most q neighbors. By Proposition 1, to solve the q -separation problem, it suffices to consider inequalities in $\mathcal{I}_{e_0}^q$. In our separation algorithm, we explicitly generate all running intersection inequalities in $\mathcal{I}_{e_0}^q$. First we construct the set E_{e_0} . For a rank- r hypergraph G , the number of edges in E_{e_0} is at most $2^r - r - 1$. This follows from condition (i) of Definition 1. By assigning a unique key to each subset of e_0 , and using the fact that the vertex list for each edge of G is sorted, it can be shown that the set E_{e_0} can be constructed in $O(r2^r|E|)$ operations. The total number of possible sets of neighbors of cardinality at most q is given by $N_q = \sum_{i=1}^q \binom{2^r - r - 1}{i}$ and it is well-known that $N_q \leq (2^r - r)^q$. For each possible set of t neighbors for some $t \leq q$, we can check the validity of condition (ii) of Definition 1, in $O(t^2r)$ operations. Moreover, we can check if the set \tilde{E} defined in condition (iii) has the running intersection property in $O(r + t)$ operations [38]. Finally, for each valid set of neighbors, we generate the corresponding unique running intersection inequality in $\mathcal{I}_{e_0}^q$ which can be done in $O(tr)$ operations. Therefore, the total running time of the separation algorithm is given by $O(|E|(r2^r|E| + 2^{r^q}q^2r))$. \square

To analyze the worst-case running time of the separation problem over all running intersection inequalities, an upper bound on the number of neighbours is needed. The following lemma provides such a bound.

Lemma 1. *Let G be a hypergraph and let $e_0 \in E(G)$. Then a running intersection inequality centered at e_0 has at most $|e_0| - 1$ neighbors.*

Proof. To prove the statement, we show that each running intersection inequality centered at e_0 has at most $t - 1$ neighbors, where t is the number of nodes in e_0 that are present in at least one neighbor. The lemma then follows since $t \leq |e_0|$.

By definition of a running intersection inequality, we have $t \geq 2$. Thus, the base case is $t = 2$. If $t = 2$, by conditions (i) and (ii) in definition of running intersection inequalities, we can have only one neighbor and thus the claim follows. For the induction step, we assume that $t \geq 3$. Consider a running intersection inequality centered at e_0 and let e_1, \dots, e_k be a running intersection ordering of the neighbors. If $k = 1$, then the proof is trivial. Henceforth, suppose that $k \geq 2$. We define a new running intersection inequality centered at e_0 and with neighbors e_1, \dots, e_{k-1} . There exists at least one node in $e_0 \cap e_k$ that is not contained in any other neighbor e_1, \dots, e_{k-1} , as otherwise there would be an edge $e_{k'}$ among them with $e_0 \cap e_k \subseteq e_0 \cap e_{k'}$ and this contradicts the definition of a running intersection inequality. It then follows that $e_0 \cap (\cup_{i=1}^{k-1} e_i) \subset e_0 \cap (\cup_{i=1}^k e_i)$, and hence by the induction hypothesis the new running intersection inequality has at most $t - 2$ neighbors. This implies that our original running intersection inequality has at most $t - 1$ neighbors. \square

The next result follows from Proposition 2 and Lemma 1.

Corollary 2. *Given a rank- r hypergraph $G = (V, E)$ and a vector $\bar{z} \in [0, 1]^{V+E}$, there exists an algorithm that solves the separation problem over all running intersection inequalities in $O(|E|(r2^r|E| + 2^{r^2}r^3))$ operations.*

Corollary 2 implies that, for a fixed-rank hypergraph $G = (V, E)$, there exists an algorithm that solves the separation problem over all running intersection inequalities in $O(|E|^2)$ operations. Before proceeding further, let us comment on the significance of the separation system by considering the use of explicit enumeration instead. Employing a similar line of analysis as in the proof of Proposition 2, we obtain an algorithm whose running time is $O(r^3|E|^r)$. It is important to note that, in almost all multilinear sets that appear in MINLPs, we have $|E| \gg 2^r$; in fact, in many problems we have $|E| = \Theta(n^r)$. Hence, in such cases, by employing the separation system, we reduce the complexity of the separation algorithms from $O(n^{r^2})$ to $O(n^{2r})$. In addition, as we detail in the next section, the costly step associated with constructing the separation system is independent of \bar{z} , and hence when incorporated in a branch-and-bound algorithm, can be performed only once at the root node. Therefore, all subsequent cut generations at each node of the search tree can be carried out very efficiently.

5 Implementation in a branch-and-cut algorithm

In this section, we describe an efficient implementation of running intersection inequalities in a branch-and-cut framework. Our implementation is incorporated within the global solver BARON [24]. Our starting point is the separation algorithm described in Section 4. We enhance the performance of this algorithm for a branch-and-cut based solver by performing most of the expensive computations only once

at the root node. More precisely, we employ the following two-step approach to generate running intersection inequalities at each node in the branch-and-cut tree:

Recognition. Prior to the initialization of the branch-and-cut tree, we mark and classify all collections of multilinear equations for which running intersection inequalities can be generated later in the tree. To this end, we first identify all multilinear terms that are present in the original problem, as well as intermediate multilinear terms that are introduced via a factorable reformulation. See [2] for a formal description of the factorable reformulator in BARON. Let us demonstrate the reformulation approach by a simple example. Consider the polynomial

$$f(z) = z_1^2 z_2 z_3 + z_1^2 z_2^2 z_3 + z_1 z_2^2 z_3.$$

Obviously, this function does not contain any multilinear terms. However, a factorable reformulation of $f(z)$ is given by $f(z) = z_{234} + z_{345} + z_{135}$, where $z_{234} = z_4 z_2 z_3$, $z_{345} = z_4 z_5 z_3$, $z_{135} = z_1 z_5 z_3$, $z_4 = z_1^2$ and, $z_5 = z_2^2$. Hence, in this lifted space, we can identify a multilinear set of the hypergraph $G = (V, E)$ with $V = \{v_1, \dots, v_5\}$ and $E = \{\{v_2, v_3, v_4\}, \{v_3, v_4, v_5\}, \{v_1, v_3, v_5\}\}$. Since factorable relaxations are already built in the lifted space, generating strong cuts for intermediate relations can significantly enhance the convergence rate of a branch-and-cut algorithm. Indeed, as we demonstrate in the next section, running intersection cuts for these lifted multilinear terms are quite effective for polynomial optimization problems.

Now suppose that the multilinear set \mathcal{S}_G associated with a factorable reformulation of the MINLP is identified. As we detail next, at each node of the branch-and-cut tree, we solve a variant of the q -separation problem for some $q \in \{1, \dots, r-1\}$. In order to efficiently generate the q -separation system at each node of the search tree, we construct the following data structure at the root node. Let $e_0 \in E$. For each $f \subseteq e_0$ with $|f| \geq 2$, we store the list of edges e in $E \setminus \{e_0\}$ with $e \cap e_0 = f$ and denote this list by $\mathcal{L}_{e_0}^f$. Define $\mathcal{L}_{e_0} = \{(f, \mathcal{L}_{e_0}^f) : \mathcal{L}_{e_0}^f \neq \emptyset\}$. From the proof of Proposition 2, it follows that, for a rank- r hypergraph, \mathcal{L}_{e_0} can be generated in $O(r2^r|E|)$ operations. For each set of subsets of e_0 with $(f, \mathcal{L}_{e_0}^f) \in \mathcal{L}_{e_0}$ and of cardinality at most q , we check the validity of condition (ii) in Definition 1. We store all such sets of subsets in \mathcal{N}_{e_0} . It can be checked that \mathcal{N}_{e_0} can be constructed in $O(rq^22^{rq})$ operations. Notice that it suffices to check condition (ii) only once at the root of the branch-and-cut tree. Now consider a collection of edges $e_0, e_k, k \in K$, that do not satisfy condition (iii) at the root node. Suppose that, later in the tree, a number of variables corresponding to a subset V' of nodes of G get fixed due to branching or range reduction operations; i.e., $\tilde{z}_v = 1$ for all $v \in V'$, where $\tilde{z}_v = z_v/u_v$ as defined in inequality (7). It is then possible that the collection $e_0 \setminus V', e_k \setminus V', k \in K$ satisfy condition (iii). Hence, to minimize the overall computational cost, we employ the following approach. At the root node, for each set F in \mathcal{N}_{e_0} , we check whether it has the running intersection property and, if so, we mark it so that no such a check is performed later in the tree. Otherwise, as we detail in the following, at certain

nodes in the search tree, we check if F has the running intersection property. Again, by the proof of Proposition 2, this running intersection check at the root node can be performed in $O((r+q)2^{rq})$ operations. Hence, the total cost of the recognition step is given by $O(|E|(r2^r|E| + rq^22^{rq}))$.

Cut generation. At each node in the branch-and-cut tree, we first construct and solve a crude outer-approximation of the problem based on the conventional factorable reformulation. Subsequently, various classes of cutting planes are generated and added to the current relaxation at multiple rounds, only if they violate the relaxation solution (see [43] for details). We do not add any of the running intersection inequalities to the initial outer-approximation but utilize them in the iterative cut generation scheme.

Let \bar{z} denote the relaxation solution at a given cut generation round. For each $e_0 \in E$ and for each set of subsets F in \mathcal{N}_{e_0} , if needed, we first test if F has the running intersection property in $O(|e_0| + t)$ operations [38]. That is, this test is performed only if (i) F does not have running intersection property at the root node and (ii) at the current node, at least one variable corresponding to a node contained in an edge in F gets fixed as a result of branching or range reduction operations. Subsequently, we generate the corresponding running intersection inequality that is present in the separation system. Namely, for each $f \in F$, among all edges $e \in \mathcal{L}_{e_0}^f$, we select an edge \bar{e} that minimizes \bar{z}_e/u_e , where u_e is the upper bound on z_e at the current node. This step can be performed in $O(|E|)$ operations. Furthermore, among all nodes $v \in N(e \cap e_0)$, we select the node that minimizes \bar{z}_v/u_v , where again u_v is an upper bound on z_v at the current node; this can be done in $O(q|e_0|)$ operations. We then compute the violation of this inequality by \bar{z} , and if the violation is larger than a preselected positive threshold, the inequality is added to BARON's cut pool. The total cost of cut generation at each round is $O(2^{rq}(|E| + rq))$.

To control the size of the LP relaxation, BARON's cut pool maintains a certain percentage of running intersection cuts with largest violation values and adds them to the relaxation of the problem at the end of each round of cut generation.

For a fixed-rank hypergraph, by employing the above two-step approach, we obtain a separation algorithm with the running time of $O(|E|^2)$ at the root node and $O(|E|)$ at each round of cut generation. As the number of cut generation rounds is often large even for moderate-sized problems, this approach leads to significant speed ups in comparison to applying the algorithm from scratch at each node of the search tree.

6 Numerical Experiments

In this section, we demonstrate the computational benefits of incorporating running intersection cuts at every node of the branch-and-reduce global solver BARON [43,

24]. To this end, we consider randomly generated test sets containing multilinear and polynomial optimization problems of degree three and four. Throughout this section, all experiments are performed with `GAMS 25.1.1` on a 64-bit Intel Xeon X5650 2.66Ghz processor; all implementations are single-threaded. In addition, all problems are solved with relative/absolute optimality tolerance of 10^{-6} , and a CPU time limit of 500 seconds. Other algorithmic parameters are set to the default settings of the `GAMS` distribution for all solvers. When comparing the performance of different algorithms, we call a problem trivial if all algorithms take less than half a second to solve it to optimality. All tests were performed with a development version of `BARON 18.6.19` that uses `CPLEX 12.8` [22] as the default solver for LP and MIP subproblems and performs dynamic local search using `CONOPT 3.171` [18], `IPOPT 3.12.8` [45], `MINOS 5.5` [30], `SNOPT 7.2-12.1` [20], `FILTERSD 2` [10], and `FILTERSQP` [32].

6.1 The test set

We consider a polynomial optimization problem of the form:

$$\begin{aligned}
 \text{(PL)} \quad & \min f_0(x) \\
 & \text{s.t. } f_i(x) \leq b_i, \forall i = 1, \dots, q \\
 & x \in [0, 1]^n,
 \end{aligned}$$

where for each $i \in \{0, 1, \dots, q\}$, $b_i \in \mathbb{R}^q$ and $f_i(x)$ is a multivariate polynomial

$$f_i(x) = \sum_{T \in \Omega_i} c_T \prod_{j \in T} x_j^{a_j},$$

Ω_i is a collection of subsets of $\{1, \dots, n\}$, while c_T , $T \in \Omega_i$ are nonzero real-valued coefficients, and the exponents a_j , $j \in T$ are positive integers. By convention, the degree of a monomial $\prod_{j \in T} x_j^{a_j}$ is the sum of its exponents, and the degree a polynomial function $f_i(x)$ is the largest degree of its monomials, i.e., $d_i = \max_T \sum_{j \in T} a_j$. Similarly, the degree of the polynomial optimization problem (PL) is defined as the largest degree of its polynomials $d = \max_i d_i$, $i = \{0, 1, \dots, q\}$. By letting $q = 0$, we obtain a box-constrained optimization problem. In the following, we assume that $d_i > 1$ for all $i \in \{0, 1, \dots, q\}$.

For a polynomial f_i with n variables and degree d_i , it can be shown that the maximum number of monomial terms is given by $W_i = \sum_{k=1}^{d_i} \binom{d_i}{k} \binom{n}{k}$, where $\binom{n}{k}$ is zero if $n < k$. Denote by w_i the number of nonlinear monomials in f_i . We associate a density with each polynomial f_i defined as $\nu_i = w_i / (W_i - n)$. Accordingly, when $\nu_i = \nu$ for all $i \in \{0, 1, \dots, q\}$, we say that the density of Problem (PL) is given by ν . Throughout this section, we characterize a polynomial optimization problem by its degree (d), number of variables (n), number of constraints (q), and density (ν). We also consider multilinear optimization problems that can be obtained by replacing

the polynomial functions of Problem (PL) by multilinear functions; i.e., $a_j = 1$ for all j . For multilinear problems, we adapt the terminology defined above for polynomials, by noting that in this case we have $W_i = \sum_{k=2}^{d_i} \binom{n}{k}$. For the numerical experiments, we consider the following two test sets:

Set 1. Multilinear optimization problems of degree three with

$$(n, \nu) \in \{(10, 1.0), (15, 0.5), (20, 0.15), (20, 0.1), (25, 0.05), (30, 0.02)\},$$

and polynomial optimization problems of degree three with

$$(n, \nu) \in \{(10, 0.75), (15, 0.25), (15, 0.15), (20, 0.1), (20, 0.05)\}.$$

Set 2. Multilinear optimization problems of degree four with

$$(n, \nu) \in \{(10, 1.0), (15, 0.15), (20, 0.02), (20, 0.01), (25, 0.01), (25, 0.005)\},$$

and polynomial optimization problems of degree four with

$$(n, \nu) \in \{(10, 0.25), (10, 0.15), (15, 0.05), (15, 0.02), (20, 0.01)\}.$$

In both test sets, we let $q \in \{0, n/5, n/2, n\}$. For each combination of $\{d, n, q, \nu\}$, we generated five problem instances, where the problem data were randomly generated from uniform distributions: the polynomial coefficients c_T were generated in the range $[-1, 1]$, while the righthand side values b_i were generated in the range $[0, 100]$. Overall, our test set contains 220 multilinear and polynomial optimization problems of degree three, and 220 multilinear and polynomial optimization problems of degree four. This collection is obtained by adding new test problems to the test set considered in [2]. The new instances are problems with more variables and/or lower density. This augmented test set is designed to examine the effect of running intersection cuts on problems with different sparsity characteristics, ranging from boxed-constrained problem to those that are highly constrained. The test problems can be obtained from [44].

6.2 Comparisons with the existing relaxations in BARON

We solve the two test sets described in Section 6.1 to global optimality using BARON, with and without running intersection cuts. For nonconvex problems of form (PL), BARON's factorable bounds consist of supporting hyperplanes and affine envelopes for univariate monomials, along with a recursive application of bilinear envelopes to bound multilinear terms. In addition, BARON generates *multilinear cutting planes* [2] corresponding to certain facets of the convex hull of a multilinear function defined as $L(z) = \sum_{e \in E'} c_e \prod_{v \in e} z_v$, where $c_e \in \mathbb{R} \setminus \{0\}$ for all $e \in E'$ and $z_v \in [l_v, u_v]$ such that $-\infty < l_v < u_v < \infty$ for all $v \in V'$. The corresponding separation

problem is an LP whose size grows exponentially with the number of variables of the multilinear function. To cope with this growth, the authors of [2] propose a decomposition technique that exploits the structure of a multilinear function to decompose it to lower-dimensional components, for which the aforementioned LP can be solved efficiently by employing a customized simplex algorithm.

To further strengthen BARON’s relaxation constructor, we employ the cut generation scheme of Section 5 to incorporate running intersection inequalities at every node of BARON’s search tree. For multilinear optimization problems, these cuts correspond to multilinear terms present in the original formulation. For polynomial optimization problems, running intersection cuts are generated for intermediate multilinear terms in the lifted space. In our implementation, we solve the q -separation problem described in the previous section by letting $q = \min\{r, 4\}$. Clearly, for our test sets this implies that we are solving the separation problem over all running intersection inequalities, as we have $r = 3$ and $r = 4$, in Set 1 and Set 2, respectively.

To compare the performance of the two algorithms, we consider the following factors: (i) execution time, (ii) total number of nodes in the branch-and-bound tree (iterations), (iii) maximum number of nodes stored in memory (nodes), and (iv) largest amount of memory utilized by BARON (memory). The memory peaks are reported in megabytes and account for all arrays allocated by BARON during the run. We first consider multilinear and polynomial optimization problems of degree three. Computational results are depicted in Figure 1. For this test set, the number of problems that are solved to global optimality without and with running intersection cuts are 208 and 216, respectively. For a meaningful comparison, we eliminated trivial problems from the test set (35 instances). For those nontrivial problems that are solvable within 500 seconds by at least one of the two algorithms (181 instances), incorporating running intersection cuts in BARON results in average reductions of 60% in CPU time, 78% in number of iterations, and 70% in maximum number of nodes in memory. Finally, due to the storage of the running-intersection data structure at the recognition step, there is a modest increase in memory requirements.

The results of Figure 1 are analyzed in Table 1(a) to further quantify the effect of incorporating the running intersection cuts. The first line of Table 1(a) provides the percentage of nontrivial problems for which running intersection cuts lead to at least a factor of two improvement with respect to CPU time, number of iterations, and number of nodes in memory. The subsequent lines of the table provides similar statistics for problems for which the algorithm was improved by at least 30% but no more than 50%, problems for which there was no significant performance change after addition of cutting planes, and problems for which there was some deterioration in performance because of cutting planes. As can be seen from Table 1(a), for about two third of the test problems, running intersection cuts reduce the CPU time of the global solver by at least a factor of two. Similar improvements are observed with respect to the total number and maximum number of nodes in the branch-and-cut tree. We observe minor performance degradations for about 3% of the test

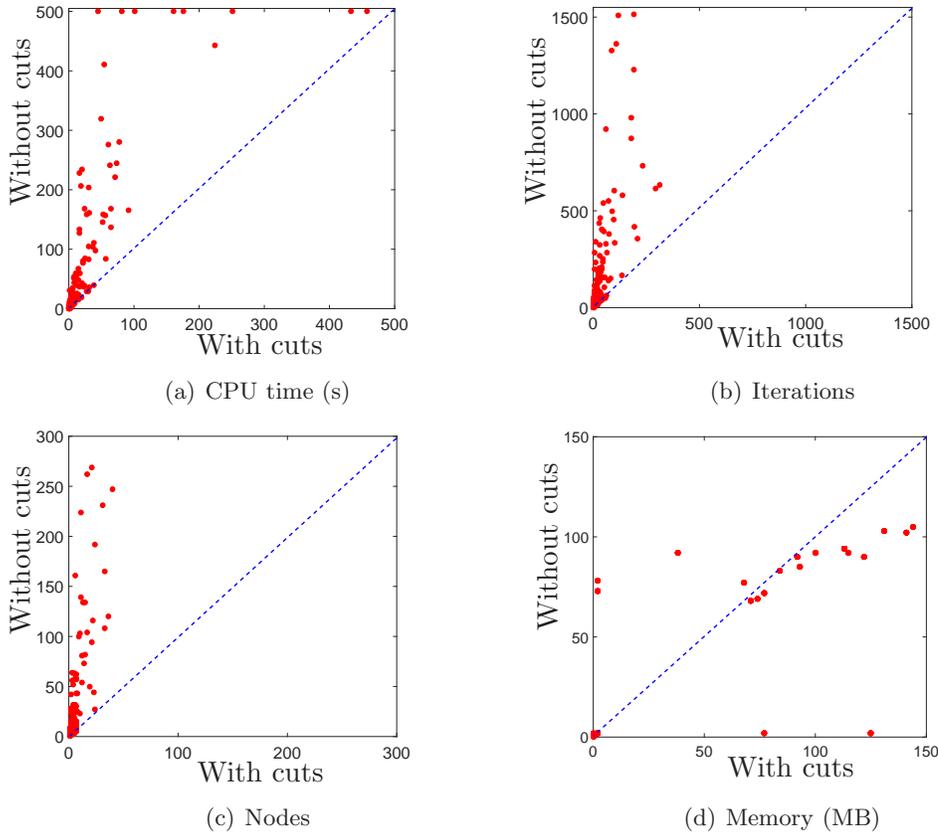


Figure 1: Performance of BARON with and without running intersection cuts for 220 third-degree multilinear and polynomial optimization problems. In these figures, nontrivial problems that are solved in less than 500 seconds are compared with respect to (a) CPU time, (b) number of iterations, (c) maximum number of nodes in memory and, (d) largest amount of memory allocated.

Table 1: Effect of adding running intersection cuts to BARON

(a) Third-degree multilinear and polynomial optimization problems

Effect of adding cuts	CPU time	Iterations	Nodes
Better by a factor at least 2	113 (62%)	130 (72 %)	129 (71 %)
Between 30% and 100% better	21 (12%)	18 (10%)	14 (8 %)
Difference smaller than 30%	42 (23%)	31 (17%)	36 (20%)
Between 30% and 100% worse	5 (3%)	2 (1%)	2 (1%)
Worse by a factor of at least 2	0 (0%)	0 (0%)	0 (0%)

(b) Fourth-degree multilinear and polynomial optimization problems

Effect of adding cuts	CPU time	Iterations	Nodes
Better by a factor at least 2	55 (35%)	114 (72%)	113 (71%)
Between 30% and 100% better	19 (12%)	16 (10%)	16 (10%)
Difference smaller than 30%	59 (37%)	25 (16%)	27 (17%)
Between 30% and 100% worse	16 (10%)	0 (0%)	0 (0%)
Worse by a factor of at least 2	10 (6%)	4 (2%)	3 (2%)

problems, most of which are due to solving more expensive LP relaxations obtained as a result of the addition of running intersection cuts.

Next, we examine the impact of running intersection cuts on multilinear and polynomial optimization problems of degree four. Out of 220 problems, 181 instances are solved to global optimality within 500 seconds by at least one of the two algorithms, 22 of which are trivial. The number of solvable problems to global optimality without and with running intersection cuts are 168 and 180, respectively. As can be seen from Figure 2 and Table 1(b), the proposed cuts significantly improve the performance of BARON for this test set. Specifically, we obtain average reductions of 43% in CPU time, 76% in number of iterations, and 72% in maximum number of nodes in memory. For this test set, the increase in the largest amount of allocated memory is more significant as the size of the data structure allocated in the recognition step is increasing in the degree of the multilinear set. Table 1(b) indicates that, thanks to the running intersection cuts, for more than one third of the problems, the execution time is improved by at least a factor of two. However, we also observe some notable performance degradations for about 6% of the problems. Again, this adverse effect is due to the increase in the size of the LP relaxations. For higher degree problems, many more running intersection inequalities can be generated and while these cuts improve the quality of the lower bounds in general, they may increase the overall cost of the branch-and-bound tree as the resulting LPs are often much more expensive to solve.

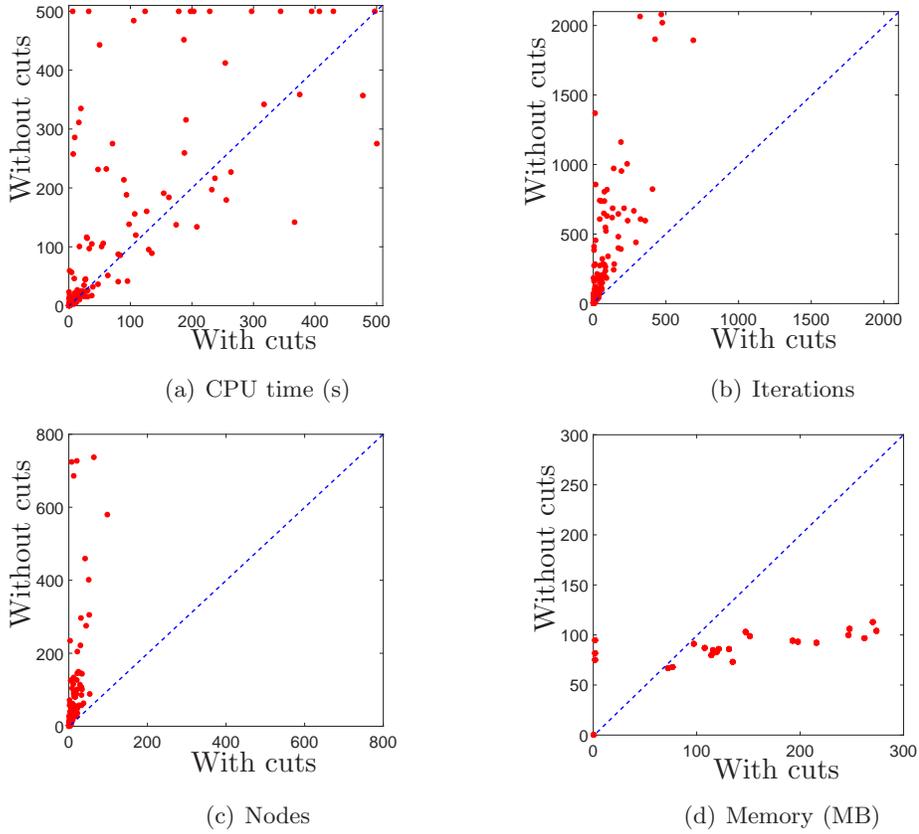


Figure 2: Performance of BARON with and without running intersection cuts for 220 fourth-degree multilinear and polynomial optimization problems. In these figures, nontrivial problems that are solved in less than 500 seconds are compared with respect to (a) CPU time, (b) number of iterations, (c) maximum number of nodes in memory and, (d) largest amount of memory allocated.

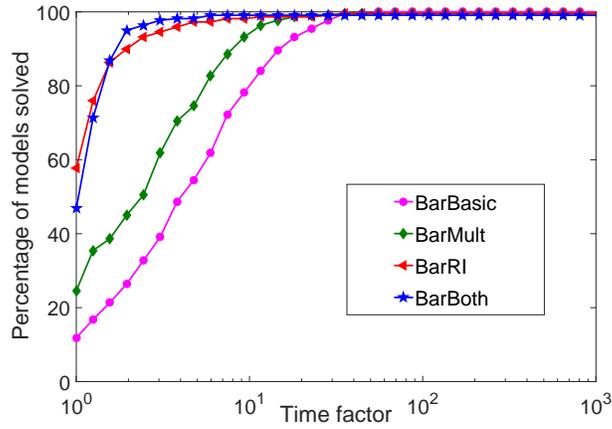
6.3 Relative impact of multilinear cuts versus running intersection cuts

In this section, we examine the relative impact of multilinear cutting planes as detailed in [2] and the proposed running intersection cuts. Results in [2] indicate that multilinear cuts significantly enhance the performance of **BARON**. In the previous section, we demonstrated that the addition of running intersection cuts leads to further improvements. The purpose of this section is to understand the relative impact of the two cut generation strategies on different problem types. To this end, we consider the following four versions of **BARON**:

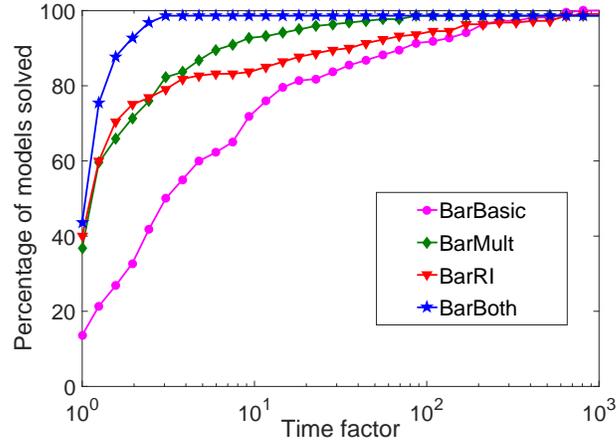
1. **BarBasic**: No multilinear cuts or running intersection cuts are generated; in this version, for our test sets, **BARON**'s relaxation constructor is essentially a conventional term-wise factorable scheme [43].
2. **BarMult**: Multilinear cuts are generated but no running intersection cut is generated.
3. **BarRI**: Running intersection cuts are generated but no multilinear cuts are generated.
4. **BarBoth**: Both multilinear cuts and running intersection cuts are generated.

To evaluate the performance of **BARON** with different cut generation strategies, we make use of performance profiles [17]. The performance profile of a solver is a (cumulative) distribution function for a performance metric. Throughout this section, we use the ratio of the time that an algorithm takes to solve a problem versus the best time of all algorithms as the performance metric. For the purpose of constructing performance profiles, by *solve* we imply that the algorithm finds the best solution among all solvers for a given problem within the time limit. We utilize the **GAMS** performance tools [19] to construct all performance profiles. To account for trivial problems as we described in the previous section, we set `resmin`= 0.5 seconds. As defined by the **GAMS** performance tools manual, `resmin` is the minimum resource time threshold. That is, if a solver reports a resource time below `resmin`, then it is increased to `resmin` before being used in ratio calculations.

Figure 3 shows the performance profiles of the four variants of **BARON** for both test sets. As can be seen from this figure, for both test sets **BarBoth** is the best solver whereas **BarBasic** is the worst. The relative performance of **BarMult** and **BarRI** however changes in the two test sets. In Table 2, for each cut generation strategy, we report the geometric mean of the CPU time taken over each test; for both test tests, **BarBoth** is on average the fastest solver. Hence, we conclude that the combination of multilinear cuts and running intersection cuts outperforms either of the two cut generation schemes in isolation.



(a) Third degree problems



(b) Fourth degree problems

Figure 3: Impact of different cut generation strategies on BARON for (a) 220 randomly generated multilinear and polynomial optimization problems of degree three and (b) 220 randomly generated multilinear and polynomial optimization problems of degree four.

Table 2: Comparison of CPU times (geometric means) using different cut generation schemes.

Test set	BarBasic	BarMult	BarRI	BarBoth
Third degree	12.75	6.60	3.42	3.02
Fourth degree	77.49	23.36	30.55	15.25

References

- [1] M. Anthony, E. Boros, Y. Crama, and A. Gruber. Quadratic reformulations of nonlinear binary optimization problems. *Mathematical Programming*, 162(1):115–144, 2017.
- [2] X. Bao, A. Khajavirad, N. V. Sahinidis, and M. Tawarmalani. Global optimization of nonconvex problems with multilinear intermediates. *Mathematical Programming Computation*, 7:1–37, 2015.
- [3] F. Barahona, M. Jünger, and G. Reinelt. Experiments in quadratic 0 – 1 programming. *Mathematical Programming*, 44:127–137, 1989.
- [4] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *Journal of the ACM*, 30:479–513, 1983.
- [5] P. Belotti, J. Lee, L. Liberti, F. Margot, and A. Wächter. Branching and bounds tightening techniques for non-convex MINLP. *Optimization Methods and Software*, 24:597–634, 2009.
- [6] T. Berthold, G. Gamrath, G. Hendel, S. Heinz, T. Koch, M. Pfetsch, S. Vigerske, R. Waniek, M. Winkler, and K. Wolter. *SCIP 3.2*, User’s Manual, 2016.
- [7] P. Bonami, O. Günlük, and J. Linderoth. Globally solving nonconvex quadratic programming problems with box constraints via integer programming methods. *Mathematical Programming Computation*, 2018.
- [8] E. Boros and A. Gruber. On quadratization of pseudo-boolean functions. *Preprint*, 2014.
- [9] C. Buchheim and G. Rinaldi. Efficient reduction of polynomial zero-one optimization to the quadratic case. *SIAM Journal on Optimization*, 18:1398–1413, 2007.
- [10] COIN-OR Project. FilterSD 2 Coin FilterSD solver. <https://projects.coin-or.org/filterSD>.
- [11] Y. Crama. Concave extensions for non-linear 0–1 maximization problems. *Mathematical Programming*, 61:53–60, 1993.
- [12] Y. Crama and E. Rodríguez-Heck. A class of valid inequalities for multilinear 0 – 1 optimization problems. *Discrete Optimization*, 25:28–47, 2017.
- [13] A. Del Pia and A. Khajavirad. A polyhedral study of binary polynomial programs. *Mathematics of Operations Research*, 42(2):389–410, 2017.

- [14] A. Del Pia and A. Khajavirad. On decomposability of multilinear sets. *Mathematical Programming*, 2017, accepted.
- [15] A. Del Pia and A. Khajavirad. The multilinear polytope for acyclic hypergraphs. *SIAM Journal on Optimization*, 28:1049–1076, 2018.
- [16] A. Del Pia and A. Khajavirad. The running intersection relaxation of the multilinear polytope. *Optimization Online manuscript 2018/05/6618*, 2018.
- [17] E. Dolan and J. More. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91:201–213, 2002.
- [18] A. Drud. *CONOPT 3.17A*, User’s Manual. ARKI Consulting and Development A/S, Bagsvaerd, Denmark, 2016.
- [19] GAMS Performance tools. Available at <http://www.gams.com/help/topic/gams.doc/solvers/allsolvers.pdf>.
- [20] P. E. Gill, W. Murray, and M. A. Saunders. User’s Guide for SNOPT 7: A FORTRAN Package for Large-Scale Nonlinear Programming. Technical report, University of California, San Diego and Stanford University, CA, 2008.
- [21] C. Helmberg and F. Rendl. Solving quadratic 0–1 problems by semidefinite programs and cutting planes. *Mathematical Programming*, 82:291–315, 1998.
- [22] IBM. *CPLEX Optimizer*, 2016.
<http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/>.
- [23] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Millera, R.E. and Thatcher, J.W., New York, 1972.
- [24] A. Khajavirad and N. V. Sahinidis. A hybrid LP/NLP paradigm for global optimization relaxations. *Mathematical Programming Computation*, May 2018.
- [25] Y. Lin and L. Schrage. The global solver in the LINDO API. *Optimization Methods and Software*, 24:657–668, 2009.
- [26] G. P. McCormick. Computability of global solutions to factorable nonconvex programs: Part I—Convex underestimating problems. *Mathematical Programming*, 10:147–175, 1976.
- [27] C. A. Meyer and Ch. A. Floudas. Trilinear monomials with positive or negative domains: Facets of the convex and concave envelopes. *Frontiers in Global Optimization*, C.A. Floudas and P.M. Pardalos (eds.), Kluwer Academic Publishers, 103:327–352, 2003.

- [28] C. A. Meyer and Ch. A. Floudas. Trilinear monomials with mixed sign domains: Facets of the convex and concave envelopes. *Journal of Global Optimization*, 29:125–155, 2004.
- [29] R. Misener and Ch. A. Floudas. ANTIGONE: Algorithms for coNTinuous/Integer Global Optimization of Nonlinear Equations. *Journal of Global Optimization*, 59:503–526, 2014.
- [30] B. A. Murtagh and M. A. Saunders. MINOS 5.5 User’s Guide. Technical Report SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, CA, 1995.
- [31] M. Padberg. The boolean quadric polytope: Some characteristics, facets and relatives. *Mathematical Programming*, 45:139–172, 1989.
- [32] R. Fletcher and S. Leyffer. User manual for filterSQP. https://wiki.mcs.anl.gov/leyffer/images/5/58/SQP_manual.pdf.
- [33] A. D. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10:425–437, 1997.
- [34] H. S. Ryoo and N. V. Sahinidis. Analysis of bounds for multilinear functions. *Journal of Global Optimization*, 19:403–424, 2001.
- [35] A. Schrijver. *Theory of Linear and Integer Programming*. Wiley, Chichester, 1986.
- [36] H. D. Sherali. Convex envelopes of multilinear functions over a unit hypercube and over special discrete sets. *Acta Mathematica Vietnamica*, 22:245–270, 1997.
- [37] 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.
- [38] R. E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566–579, 1984.
- [39] M. Tawarmalani. Inclusion certificates and simultaneous convexification of functions. *Working paper*, 2010.
- [40] M. Tawarmalani, J.-P. Richard, and K. Chung. Strong valid inequalities for orthogonal disjunctions and bilinear covering sets. *Mathematical Programming*, 124:481–512, 2010.
- [41] M. Tawarmalani, J.-P. Richard, and C. Xiong. Explicit convex and concave envelopes through polyhedral subdivisions. *Mathematical Programming*, 138:531–577, 2013.

- [42] M. Tawarmalani and N. V. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Mathematical Programming*, 99:563–591, 2004.
- [43] M. Tawarmalani and N. V. Sahinidis. A polyhedral branch-and-cut approach to global optimization. *Mathematical Programming*, 103:225–249, 2005.
- [44] The Optimization Firm, LLC. NLP and MINLP test problems. <https://minlp.com/nlp-and-minlp-test-problems>.
- [45] A. Wächter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106:25–57, 2006.
- [46] Y. Yajima and T. Fujie. A polyhedral approach for nonconvex quadratic programming problems with box constraints. *Journal of Global Optimization*, 13:151–170, 1998.