The Bipartite Boolean Quadric Polytope with Multiple-Choice Constraints

Andreas Bärmann[1], Alexander Martin[1] and Oskar Schneider[2]

[1] Andreas.Baermann@math.uni-erlangen.de
Alexander.Martin@math.uni-erlangen.de
Lehrstuhl für Wirtschaftsmathematik,
Department Mathematik,
Friedrich-Alexander-Universität Erlangen-Nürnberg,
Cauerstraße 11, 91058 Erlangen, Germany

[2] Oskar.Schneider@fau.de
Gruppe Optimization
Fraunhofer Arbeitsgruppe für Supply-Chain Services SCS,
Fraunhofer Institut für Integrierte Schaltungen IIS,
Nordostpark 93, 90411 Nürnberg, Germany

First Draft Online: 24 September 2020

Abstract

We consider the bipartite boolean quadric polytope (BQP) with multiple-choice constraints and analyse its combinatorial properties. The well-studied BQP is defined as the convex hull of all quadric incidence vectors over a bipartite graph. In this work, we study the case where there is a partition on one of the two bipartite node sets such that at most one node per subset of the partition can be chosen. This polytope arises, for instance, in pooling problems with fixed proportions of the inputs at each pool. We show that it inherits many characteristics from BQP, among them a wide range of facet classes and operations which are facet preserving. Moreover, we characterize various cases in which the polytope is completely described via the relaxation-linearization inequalities. The special structure induced by the additional multiple-choice constraints also allows for new facet-preserving symmetries as well as lifting operations. Furthermore, it leads to several novel facet classes as well as extensions of these via lifting. We additionally give computationally tractable exact separation algorithms, most of which run in polynomial time. Finally, we demonstrate the strength of both the inherited and the new facet classes in computational experiments on random as well as real-world problem instances. It turns out that in many cases we can close the optimality gap almost completely via cutting planes alone, and, consequently, solution times can be reduced significantly.

Keywords: Boolean Quadric Polytope, Multiple-Choice Constraints, Convex-Hull Description, Lifting, Pooling Problem

Mathematics Subject Classification: 90C20 - 90C27 - 90C26 - 90C57 - 90C90

1 Introduction

The *boolean quadric polytope* of an undirected graph $G = (V, E)$ is defined as

$$QP(G) := \operatorname{conv} \left\{ (x, z) \in \{0, 1\}^{V \cup E} \;\middle|\; x_i x_j = z_{ij} \; \forall \{i, j\} \in E \right\}$$

and was introduced by Padberg in [Pad89]. Due to its fundamental role in the field of polyhedral combinatorics and its frequent occurrence in practical applications, it has been extensively studied, and many facet classes, corresponding separation algorithms, symmetries and further geometric properties have been found, see e.g. [Pad89, SLA95, BM86, BH93, LS14]. In [DS90], it has been shown that QP is the image of the *cut polytope* over an appropriate graph under an affine transformation called *covariance mapping*. It has also been investigated under the name *correlation polytope* in [Pit91]. For an extensive compilation of the above results, and many more, we refer the interested reader to [DLW97].

In the special case where $G = (X \cup Y, E)$ is a bipartite graph, the polytope is called the *bipartite boolean quadric polytope*:

$$BQP(G) := \operatorname{conv} \left\{ (x, y, z) \in \{0, 1\}^{X \cup Y \cup E} \;\middle|\; x_i y_j = z_{ij} \; \forall \{i, j\} \in E \right\}.$$

and its geometry and further properties have been studied in [SPS19, Sri14, PSK13, PSK15], among others. In the present article, we consider the bipartite case together with an additional *multiple-choice* (or *set-packing*) structure on the set X. Let $\mathcal{I}$ be a partition of X, and let $X^{\mathcal{I}} := \left\{ x \in \{0, 1\}^X \;\middle|\; \sum_{i \in I} x_i \leq 1 \; \forall I \in \mathcal{I} \right\}$ be the set of incidence vectors for which at most one entry per subset in the partition is set to one. A substructure like this is prevalent in many applications when competing choices or compatibilities between decision are involved. This includes the knapsack problem with multiple-choice constraints ([Nau87, KPP04]), with applications in investment planning, among others, or the clique problem with multiple-choice constraints ([BGMS18, BGM20]), which arises, for example, in scheduling problems ([BMS20]) and flow problems with piecewise linear routing costs ([LM16]). In the case of the boolean quadric polytope, the multiple-choice structure models given, fixed proportions between the multiplied quantities. The underlying feasible set, our polytope of interest, can then be stated as

$$P(G, \mathcal{I}) := \operatorname{conv} \left\{ (x, y, z) \in \{0, 1\}^{X \cup Y \cup E} \;\middle|\; x_i y_j = z_{ij} \; \forall \{i, j\} \in E, x \in X^{\mathcal{I}} \right\}.$$

Clearly, if all subsets in $\mathcal{I}$ contain only one node each, then $P(G, \mathcal{I}) = BQP(G)$, otherwise it is obvious that $P(G, \mathcal{I}) \subseteq BQP(G)$, but the former is not a face of the latter. We will study structural properties of this polytope, most notably symmetries, facet classes and separation routines, some of which are inherited from the original (bipartite) QP, while others arise specifically due to the multiple-choice structure.

The boolean quadric polytope in bilinear programming The boolean quadric polytope and its variants play a major role in the solution of bilinear programs. State-of-the-art solvers typically rely on linear programming (LP) relaxations of the non-convex products of variables involved. Most of them use the *McCormick-relaxation* (see [McC76]), since it is the best possible linear relaxation for the product of two continuous variables over their (finite) bounds. Obviously, valid inequalities for QP lead to improved relaxations. If a bilinear program has further combinatorial substructures, studying these in combination with QP allows for even tighter relaxations. Examples of this are the boolean quadric polytope over the forest sets of a graph (see [LL04]) and the cardinality-constrained boolean quadric polytope (see [Meh97, FT05, LG17]). In [HLL98], the authors examine the boolean quadric packing uncapacitated facility location polytope, which models uncapacitated facility location with bilinear costs terms. Further examples are [Cas15, BDK^{+}17, FL18, GACD13], cf. also [KCG13] and the references therein. In *separable* (or *disjoint*) bilinear programs, the variables are partitioned into two subsets such that there are no products between any two variables in the same subset. Moreover, these two subsets of variables are not coupled via further (linear) constraints. This special case motivates the study of the bipartite version of QP, i.e. BQP, see e.g. [GGL19, GLL12]. In particular, the authors of [Gup16], investigate the polyhedral structure of separable bilinear programs where one of the two variable sets obeys a single multiple-choice constraint. For this case, they can give a complete description of the convex hull of feasible solutions. Furthermore, BQP with multiple-choice constraints is studied in the context of the bipartite quadratic assignment problem (see e.g. [PW16]), and, closely related, the bilinear assignment problem (see e.g. [CSPB17]). Here, multiple-choice constraints can be used to model the allocation of resources with different properties such that precisely (or at most) one asset with a certain property has to be positioned at each location.

Contribution and organization of the paper We are interested in bilinear programs where the variables in one of the subsets are covered by non-overlapping multiple-choice constraints, which leads us to study the polytope P. This work is motivated by a real-world pooling problem (see [MF09, ABH$^+$04] for a survey) arising in the food industry. There the products have to be manufactured according to given recipes with fixed proportions of the ingredients. For each of these ingredients, there are potentially many different batches of different qualities on stock to choose from. The practical solution of pooling problems often relies on strong relaxations of an underlying bilinear model, as investigated in [GADC17, DLL11], for example. In a similar spirit, we use our theoretical results on the polyhedral structure of P to demonstrate their effect when solving the mentioned pooling problem with recipes. We show that many of the original symmetries of BQP and further general characteristics are preserved in P, see Section 2. Moreover, a multitude of new facet-preserving symmetries arises, which is very beneficial for the design of separation and lifting routines. As we show in Section 3, we also inherit large part of the facial structure of BQP. However, the multiple-choice structure gives rise to a variety of novel facet classes as well. For the case of cycle-free dependencies between the two bipartite subdivisions of the underlying graph, we can even give a complete convex-hull description for P. In Section 4, we will devise separation algorithms for several of the inherited and the new facet classes, most of them running in polynomial time. Finally, our computational experiments in Section 5 demonstrate that these routines are sufficient to close the integrality significantly, sometimes completely. Furthermore, we present a real-world computational study for a pooling problem with recipes where we show that the exploitation of the multiple-choice structure outperforms a bilinear solver based on McCormick-relaxations by orders of magnitude. Section 6 rounds the paper off with our conclusions. Finally, we give some of the details on the obtained results in the appendix.

Notation For ease of notation, we denote for a vector $a \in \mathbb{R}^{X \cup Y \cup E}$ by a_i the component corresponding to node $i \in X$, similarly a_j for $j \in Y$ and a_{ij} for $\{i,j\} \in E$, assuming an arbitrary fixed order on $X \cup Y \cup E$. Analogously, e_i, e_j and e_{ij} denote the corresponding unit vectors. For a node $v \in X \cup Y$, we define the *neighbourhood* $N(v) := \{w \in X \cup Y \colon \{v,w\} \in E\}$. Furthermore, we call the graph G *subset-uniform* with respect to the partition $\mathcal{I}$, if any two nodes which are in the same subset of I also have the same neighbourhood, i.e. the same neighbours in Y. For such a graph G, we define the corresponding *dependency graph* $\mathcal{G} = (\mathcal{I} \cup Y, \mathcal{E})$ by merging the nodes in each subset of the partition to a single node, represented by the subset itself. Its edge set $\mathcal{E}$ contains the merged edges $\{I,j\}$ for all $I \in \mathcal{I}$ and all $j \in J$ in the joint neighbourhood of the original nodes in I. Valid inequalities $a^T(x^T, y^T, z^T)^T \leq b$ for $P(G,\mathcal{I})$ will be written as $a^T(x,y,z) \leq b$ for short. For such a valid inequality, we define the *Y-support*, given by $\operatorname{supp}_Y(a) := \{j \in Y \mid a_j \neq 0 \vee (\exists i \in N(j))\, a_{ij} \neq 0\}$, to denote the nodes in Y involved. Finally, let $[n] := \{1, \ldots, n\}$.

2 General properties of $P(G,\mathcal{I})$

We start with some basic properties of $P(G,\mathcal{I})$. In particular, these include symmetries of the polytope under different operations on the coefficients of valid and facet-defining inequalities. Some of these symmetries are inherited from BQP, others are induced especially by our multiple-choice structure.

First, observe that it does not matter whether we define $P(G,\mathcal{I})$ as the convex hull of binary or continuous vectors. Namely, let $\bar{X}^{\mathcal{I}} := \{x \in [0,1]^X \mid \sum_{i\in I} x_i \leq 1 \ \forall I \in \mathcal{I}\}$, and define the set $S(G,\mathcal{I}) := \{(x,y,z) \in [0,1]^{X\cup Y\cup E} \mid x_i y_j = z_{ij} \ \forall \{i,j\} \in E,\ x \in \bar{X}^{\mathcal{I}}\}$. Then it is obvious that the extreme points of $\operatorname{conv}(S(G,\mathcal{I}))$ are of the form $(\bar{x},\bar{y},\bar{z})$, where $\bar{x}$ is a vertex of $\bar{X}^{\mathcal{I}}$, $\bar{y}$ is a vertex of $[0,1]^Y$ and $\bar{z} = \bar{x}\bar{y}$. This implies $P(G,\mathcal{I}) = \operatorname{conv}(S(G,\mathcal{I}))$, and therefore $\operatorname{conv}(S(G,\mathcal{I}))$ is a polytope. For general properties of such polyhedra arising from separable bilinear programs where the two underlying sets of feasible vectors are polytopes, we refer to [GGL19].

In the following, we give two immediate results about $P(G,\mathcal{I})$.

Proposition 2.1 (NP-hardness). *Optimizing a linear objective over $P(G,\mathcal{I})$ is NP-hard, even if each subset in the partition contains only one element.*

Proof. The NP-hardness of optimizing a linear objective over $BQP(G)$, i.e. the case with one element per subset, follows from [Sri14, Theorem 2.1]. □

Proposition 2.2 (Dimensionality). *The polytope $P(G,\mathcal{I})$ is full-dimensional.*

Proof. The polytope contains the 0-vector as well as the vectors e_i for all $i \in X$, e_j for all $j \in Y$ and $e_i + e_j + e_{ij}$ for all $\{i,j\} \in E$. These $(|X| + |Y| + |E| + 1)$-many points are easily seen to be affinely independent. □

We will now focus on graph operations and symmetries which preserve validity of an inequality for $P(G,\mathcal{I})$ as well as the property to be facet-defining. Let $\hat{G} = (\hat{X} \cup \hat{Y}, \hat{E})$ be a subgraph of G, where $\hat{X} \subseteq X$, $\hat{Y} \subseteq Y$ and $\hat{E} \subseteq \hat{X} \times \hat{Y}$. Furthermore, let $\hat{\mathcal{I}}$ be the restriction of the partition $\mathcal{I}$ corresponding to $\hat{X}$. We define the *extension* of a valid inequality for $P(\hat{G},\hat{\mathcal{I}})$ to a valid inequality for $P(G,\mathcal{I})$ by adding a 0-coefficient for each additional node and edge. This procedure is called 0-lifting in the literature. Similarly, we define the *extension* of a point in $P(\hat{G},\hat{\mathcal{I}})$ to a point in $P(G,\mathcal{I})$ by adding a 0-entry for each additional node in X, which also uniquely defines the entries corresponding to the additional edges. The *restriction* of a valid inequality for $P(G,\mathcal{I})$ or a point in $P(G,\mathcal{I})$ to $P(\hat{G},\hat{\mathcal{I}})$ is then defined conversely by discarding all components for which there is no corresponding node or edge in $\hat{G}$. We obtain the following results for extended and restricted inequalities respectively.

Proposition 2.3 (Validity of extension and restriction).

1. *If $\hat{G}$ is a subgraph of G and $\mathcal{I}$ an extension of $\hat{\mathcal{I}}$, then the extension of a valid inequality for $P(\hat{G},\hat{\mathcal{I}})$ is a valid inequality for $P(G,\mathcal{I})$.*
2. *If $\hat{G}$ is an induced subgraph of $\hat{G}$ and $\hat{\mathcal{I}}$ a restriction of $\mathcal{I}$, then the restriction of a valid inequality for $P(G,\mathcal{I})$ is a valid inequality for $P(\hat{G},\hat{\mathcal{I}})$.*

Proof. 1. The restriction of each point in $P(G,\mathcal{I})$ is a point in $P(\hat{G},\hat{\mathcal{I}})$. Therefore, the extended inequality is valid for $P(G,\mathcal{I})$. 2. For each point in $P(\hat{G},\hat{\mathcal{I}})$, the extension is a point in $P(G,\mathcal{I})$. As $\hat{G}$ is an induced subgraph of G, the extension only has additional 0-entries. Thus, the restricted inequality is valid for $P(\hat{G},\hat{\mathcal{I}})$. □

Proposition 2.4 (Facets by extension). *Let $\hat{G} = (\hat{X} \cup \hat{Y}, \hat{E})$ be an induced subgraph of G such that $\hat{X} = X$, and let $\hat{\mathcal{I}} = \mathcal{I}$. Then the extension of any facet-defining inequality for $P(\hat{G},\mathcal{I})$ is also facet-defining for $P(G,\mathcal{I})$.*

Proof. The result follows via the same construction of affinely independent points as in the proof for the original QP (see [Pad89, Theorem 3 (Lifting Theorem)]). $\square$

Theorem 2.4 also holds if we add a new subset to the partition which contains a single node. However, inequality (6), which we introduce in Section 3.1, is a counterexample which shows that increasing the graph through an extension of the partition does generally not preserve facets.

Next, we consider symmetries of $P(G, \mathcal{I})$ and their effects on its facial structure. The following result for permutations of constraint coefficients is straightforward.

Proposition 2.5 (Permutation)**.** *For a subset-uniform graph G, let σ be a permutation on $X \cup Y \cup E$ which permutes elements within a subset $I \in \mathcal{I}$ or elements in Y and permutes the related edges in E accordingly. Then the following two statements are equivalent:*

1. *The inequality $a^T(x, y, z) \leq b$ is valid (resp. facet inducing) for $P(G, \mathcal{I})$.*
2. *The inequality $\sigma(a)^T(x, y, z) \leq b$ is valid (resp. facet inducing) for $P(G, \mathcal{I})$, where $\sigma(a)$ denotes the resorted vector according to σ.*

If G is not subset-uniform, Theorem 2.5 only holds when permuting nodes sharing the same neighbourhood (together with their incident edges). Furthermore, if I contains two subsets of the same size in which the respective nodes share the same neighbourhood, then the statement of Theorem 2.5 also holds for swapping all the nodes between these two subsets.

It is also possible to formulate a variant of the well-known switching transformation (see [Pad89, Theorem 6]) for $P(G, \mathcal{I})$. In contrast to the original transformation on QP, it is, however, only possible to switch on a subset of the y-variables here.

Proposition 2.6 (Switching)**.** *Let the inequality $a^T(x, y, z) \leq b$ be valid (resp. facet-inducing) for $P(G, \mathcal{I})$, let further $\hat{Y} \subseteq Y$ and define*

$$\hat{a}_i := \begin{cases} a_i + \sum_{j \in \hat{Y}} a_{ij} & \text{for } I \in \mathcal{I}, i \in I \end{cases} \qquad \hat{a}_{ij} := \begin{cases} -a_{ij} & \text{for } \{i,j\} \in E \text{ with } j \in \hat{Y} \\ a_{ij} & \text{otherwise} \end{cases}$$

$$\hat{a}_j := \begin{cases} -a_j & \text{for } j \in \hat{Y} \\ a_j & \text{otherwise} \end{cases} \qquad \hat{b} := b - \sum_{j \in \hat{Y}} a_j$$

then $\hat{a}^T(x, y, z) \leq \hat{b}$ is a valid (resp. facet-inducing) inequality for $P(G, \mathcal{I})$ as well.

Proof. Define the one-to-one mapping $\psi \colon P(G, \mathcal{I}) \to P(G, \mathcal{I})$ for which $v = (x, y, z)$ is mapped onto $(\hat{x}, \hat{y}, \hat{z})$ with $\hat{x} = x$ as well as

$$\hat{y}_j := \begin{cases} 1 - y_j & \text{for } j \in \hat{Y} \\ y_j & \text{otherwise} \end{cases} \quad \text{and} \quad \hat{z}_{ij} := \begin{cases} -z_{ij} + x_i & \text{for } \{i,j\} \in E \text{ with } j \in \hat{Y} \\ z_{ij} & \text{otherwise} \end{cases}.$$

One can easily check that $v_1, \ldots, v_m \in P(G, \mathcal{I})$ are affinely independent iff $\psi(v_1), \ldots, \psi(v_m) \in P(G, \mathcal{I})$ are. Thus, ψ maps facets onto facets. $\square$

Note that the Y-support of an inequality is preserved under switching. Moreover, one can separate in polynomial time over all switchings of a given inequality by iteratively checking if the inequality is tightened by switching on each $j \in Y$ separately.

The following symmetric operation on $P(G, \mathcal{I})$, which we call *copying*, is novel in the sense that there is no corresponding operation on either QP or BQP. It arises specifically due to the multiple-choice constraints.

Definition 2.7 (Copying)**.** *Let $a^T(x,y,z) \leq b$ be a valid inequality for $P(G,\mathcal{I})$. For an $I \in \mathcal{I}$ and an $i \in I$ let $r_i \in \mathbb{R}^{1+|N(i)|}$ be the tuple of the coefficients of a corresponding to the node i and the incident edges. Further, let H^I be the set of coefficient tuples for all nodes in a subset I. For a subset-uniform graph, a* copying *of a is obtained by replacing each coefficient tuple r_i for $I \in \mathcal{I}$, $i \in I$, by some tuple in $h \in H^I$. If we further restrict the copying operation such that in the transformed inequality each element in $H^I \setminus \{(0,0,\dots,0)\}$ for each $I \in \mathcal{I}$ needs to be chosen at least once, then we call it a* structure-preserving *copying.*

Proposition 2.8 (Validity of copying)**.** *Let $a^T(x,y,z) \leq b$ be a valid inequality for $P(G,\mathcal{I})$, then any copied inequality is also valid.*

Proof. Due to $x \in X^{\mathcal{I}}$, only one variable per subset in $\mathcal{I}$ can be set to 1 in a feasible solution. The correctness then follows from Theorem 2.5. □

If the graph is not subset-uniform, Theorem 2.8 still holds if the copying is only performed among nodes which share the same neighbourhood.

Similar to switching, copying is able to generate an exponential number of new valid inequalities from some given valid inequality. Note that one can separate over all of these copyings in polynomial time by iteratively checking which coefficient tuple makes the inequality tightest for each element of each subset of the partition separately. Thus, copying provides a very efficient lifting procedure. An interesting question is now if copying maps facets onto facets. We will see in Section 3.1 that in general the answer is no, where the inequality (4) can serve as a counterexample. However, in Section 3.2.1 we will introduce several classes of facet-defining inequalities for which structure-preserving copying maps facets onto other facets. For the remainder of this paper, we will always mean structure-preserving copying when we refer to copying.

3 Facet-defining inequalities

In this section, we will describe several classes of facet-defining inequalities for the polytope $P(G,\mathcal{I})$ and characterize cases in which they are sufficient to completely describe its convex hull.

3.1 Basic and RLT inequalities

The following valid inequalities for $P(G,\mathcal{I})$ are part of its definition, which is why we call them the *basic inequalities*:

$$0 \leq y_j \leq 1 \quad \forall j \in Y, \tag{1}$$

$$x_i \geq 0 \quad \forall I \in \mathcal{I}, i \in I, \tag{2}$$

$$\sum_{i \in I} x_i \leq 1 \quad \forall I \in \mathcal{I}. \tag{3}$$

By applying the well-known *Reformulation-Linearization Technique* (see [SA92]) to the inequalities (1)–(3), we obtain the following system of inequalities:

$$z_{ij} \geq 0 \quad \forall \{i,j\} \in E, \tag{4}$$

$$x_i - z_{ij} \geq 0 \quad \forall \{i,j\} \in E, \tag{5}$$

$$y_j - \sum_{i \in N(j) \cap I} z_{ij} \geq 0 \quad \forall I \in \mathcal{I}, j \in Y, \tag{6}$$

$$y_j + \sum_{i \in N(j) \cap I} \left(x_i - z_{ij}\right) \leq 1 \quad \forall I \in \mathcal{I}, j \in Y. \tag{7}$$

We will call (4)–(7) the *RLT inequalities*. In some cases, the basic inequalities are already facet-defining by themselves; however, they are most of the time dominated by the RLT inequalities, as the following two results show.

Theorem 3.1 (Basic facets)**.**

1. *Iff $N(j) = \emptyset$ for some $j \in Y$, the bounds in* (1) *define facets of $P(G, \mathcal{I})$.*
2. *Iff $N(i) = \emptyset$ for some $i \in X$, inequality* (2) *defines a facet of $P(G, \mathcal{I})$.*
3. *Iff $\bigcap_{i \in I} N(i) = \emptyset$ for some $I \in \mathcal{I}$, inequality* (3) *defines a facet of $P(G, \mathcal{I})$.*

Proof. It easy to see that the stated conditions are sufficient for the corresponding inequalities to be facet-defining. Otherwise, the lower bound in (1) is dominated by (4) and (6), the upper bound in (1) is dominated by (5) and (7), (2) is dominated by (4) and (5), and (3) is dominated by (6) and (7). □

Theorem 3.2 (RLT facets)**.** *The RLT inequalities* (4)–(7) *define facets of the polytope $P(G, \mathcal{I})$, except for the case when $N(j) \cap I = \emptyset$ in* (6) *or* (7)*.*

Proof. The 0-vector as well as e_i for $i \in X$, e_j for $j \in Y$ and $e_i + e_j + e_{ij}$ for $\{i, j\} \in E \setminus \{i_1, j_1\}$ satisfy $z_{i_1 j_1} = 0$ for each $\{i_1, j_1\} \in E$. They are affinely independent, which shows that (4) defines a facet. Inequality (5) induces a facet since it is a switching of (4) for $\hat{Y} := \{j_1\}$. The 0-vector as well as e_i for $i \in X$, e_j for $j \in Y \setminus \{j_1\}$ and $e_i + e_j + e_{ij}$ for $\{i, j\} \in E$ satisfy $y_{j_1} - \sum_{i \in N(j_1) \cap I_1} z_{ij_1} = 0$ for each $I_1 \in \mathcal{I}$ and $j_1 \in Y$. Their affine independence shows that (6) defines a facet as well. Again, (7) is a switching of (6) for $\hat{Y} := \{j_1\}$ and thus also induces a facet. □

3.1.1 Complete description on cycle-free dependency graphs

In the following, we describe sufficient conditions for the graph G and the partition $\mathcal{I}$ which guarantee that the basic and RLT inequalities completely describe $P(G, \mathcal{I})$.

An initial, simple conclusion can be drawn directly from Theorem 1 in [Gup16]: in case there is only one subset in the partition $\mathcal{I}$, i.e. $\mathcal{I} = \{X\}$, and G is a complete bipartite graph, the RLT inequalities are indeed sufficient to describe $P(G, \mathcal{I})$. We will now generalize this finding in two ways: our main result will be that the basic and RLT inequalities are sufficient for subset-uniform graphs G with a cycle-free dependency graph, independent from the number of subsets in the partition. Then we will see that the basic and RLT inequalities together fully describe $P(G, \mathcal{I})$ for arbitrary bipartite graphs G if, as in [Gup16], $\mathcal{I} = \{X\}$ holds.

Our proofs are based on Zuckerberg's method for deriving convex-hull descriptions for combinatorial problems (see [Zuc16, BZ04]). We briefly summarize it here, based on the simplified formulation given in [GKRW20]. Consider a 0/1-polytope $R := \operatorname{conv}(\mathcal{F})$ with vertex set $\mathcal{F} \subseteq \{0, 1\}^n$ and a second polytope $H \subseteq \mathbb{R}^n$ (typically given via an inequality description) for which we would like to show $R = H$. We can prove this by verifying both $\mathcal{F} \subseteq H$ and $H \subseteq R$. To show the latter inclusion, we first need to represent $\mathcal{F}$ as a finite set-theoretic expression consisting of unions, intersections and complements of the sets

$$A_i := \{ a \in \{0, 1\}^n \mid a_i = 1 \}, \quad i = 1, \ldots, n.$$

Let $F(A_1, \ldots, A_n)$ be such a representation. Further, define $U := [0, 1)$, let $\mathcal{L}$ be the set of all unions of finitely many half-open subintervals of U, and let μ be the Lebesgue measure (restricted to $\mathcal{L}$), that is

$$\mathcal{L} := \{ [a_1, b_1) \cup \ldots \cup [a_k, b_k) \mid 0 \leq a_1 < b_1 < a_2 < b_2 < \ldots < a_k < b_k \leq 1, k \in \mathbb{N} \},$$
$$\mu(S) := (b_1 - a_1) + \ldots + (b_k - a_k) \text{ for any } S = [a_1, b_1) \cup \ldots \cup [a_k, b_k) \in \mathcal{L}.$$

For each point $h \in H$, we then have to find a collection of sets $S_1, \ldots, S_n \in \mathcal{L}$ such that $\mu(S_i) = h_i$ for all $i \in [n]$ and $F(S_1, \ldots, S_n) = U$ (where the complement is taken in U instead of $\{0,1\}^n$). With these prerequisites, the following theorem, a slight reformulation of [GKRW20, Theorem 4], gives the desired result.

Theorem 3.3 (Zuckerberg's convex hull characterization)**.** *Let $\mathcal{F} \subseteq \{0,1\}^n$, $h \in [0,1]^n$, and let F be a finite set theoretic expression with $F(A_1, \ldots, A_n) = \mathcal{F}$. Then $h \in \operatorname{conv}(\mathcal{F})$ if and only if there are sets $S_1, \ldots, S_n \in \mathcal{L}$ such that $\mu(S_i) = h_i$ for all $i \in [n]$, and $F(S_1, \ldots, S_n) = U$.*

Proof. Combine Theorem 4 with the arguments in Remark 2, both from [GKRW20]. □

A detailed discussion of this proof technique can be found in the mentioned literature.

It is obvious that every binary set $\mathcal{F}$ can be represented by some formula F by encoding each point $v \in \mathcal{F}$ separately, namely via the set-theoretic subexpressions $F_v(A_1, \ldots, A_n) := \bigcap_{i \in [n]:\, v_i = 1} A_i \cap \bigcap_{i \in [n]:\, v_i = 0} \bar{A}_i = \{v\}$. Forming the union of these subexpressions yields a representation $F(A_1, \ldots, A_n)$ of $\mathcal{F}$ in disjunctive normal form. However, from this form it is usually very hard to deduce a construction rule for the sets $S_1, \ldots, S_n$ fulfilling all requirements. A compact description of F is much more indicative in this respect, such as the one we give for $P(G, \mathcal{I})$ in the following.

Lemma 3.4 (Set characterization for $P(G, \mathcal{I})$)**.** *Let $\mathcal{F} := P(G, \mathcal{I}) \cap \{0,1\}^{X \cup Y \cup E}$ and $h \in [0,1]^{X \cup Y \cup E}$. Then $h \in \operatorname{conv}(\mathcal{F})$ if and only if there are sets $S_i \in \mathcal{L}$ for $i \in X$, $S_j \in \mathcal{L}$ for $j \in Y$ and $S_{ij} \in \mathcal{L}$ for $\{i, j\} \in E$ satisfying all of the following conditions:*

(i) $\mu(S_i) = h_i$ *for all* $i \in X$,

(ii) $\mu(S_j) = h_j$ *for all* $j \in Y$,

(iii) $\mu(S_{ij}) = h_{ij}$ *for all* $\{i, j\} \in E$,

(iv) $S_i \cap S_j = S_{ij}$ *for all* $\{i, j\} \in E$,

(v) $S_{i_1} \cap S_{i_2} = \emptyset$ *for all distinct* $i_1, i_2 \in I$ *with* $I \in \mathcal{I}$.

Proof. Let A_i for $i \in X$, A_j for $j \in J$ and A_{ij} for $\{i, j\} \in E$ be defined analogously to Section 3.1.1. The set $\mathcal{F}$ can then be written as

$$F(A_1, \ldots, A_n) := \bigcap_{\{i,j\} \in E} \underbrace{(A_i \cap A_j \Leftrightarrow A_{ij})}_{(a)\ x_i y_i = z_{ij}} \cap \bigcap_{I \in \mathcal{I}} \underbrace{\overline{\bigcup_{i_1, i_2 \in I : i_1 \neq i_2} \underbrace{A_{i_1} \cap A_{i_2}}_{x_{i_1} + x_{i_2} \leq 1}}}_{(b)\ \sum_{i \in I} x_i \leq 1}.$$

Each intersection in the above expression represents a defining constraint for the vertices of $P(G, \mathcal{I})$. Subexpressions (a) ensure that all resulting vectors are valid for $BQP(G)$ while subexpressions (b) require them to fulfil the multiple-choice constraints. Conditions (iv) and (v) together, stemming from (a) and (b) respectively, are now equivalent to $F(S_1, \ldots, S_n) = U$. The rest follows from Theorem 3.3. □

We readily see that Theorem 3.4 also yields a certificate for $h \in BQP(G)$ (or $h \in QP(G)$ for a non-bipartite graph G) if we omit condition (v). We only need to define F without the intersection with subexpressions (b) in the proof.

We are now ready to prove our main convex-hull result.

Theorem 3.5 (Complete description for cycle-free dependency graphs)**.** *Let H be the polytope defined by the basic and RLT inequalities (1) to (7). If G is a subset-uniform graph and the dependency graph $\mathcal{G}$ is cycle-free, we have $H = P(G, \mathcal{I})$.*

Proof. First, note that $P(G,\mathcal{I}) \subseteq H$ holds, as (1) to (7) are valid constraints for $P(G,\mathcal{I})$. To prove $H \subseteq P(G,\mathcal{I})$, we give an explicit construction of the sets S_i, S_j and S_{ij} for each point $h \in H$ as required in Theorem 3.4. As $\mathcal{G}$ is cycle-free, we can represent it as a finite list of trees. This allows us to define the required sets via the algorithm described in the following, which recursively traverses each tree. It consists of the three routines Define-Sets, Traverse-Tree and Match. The auxiliary routine Match is given in Figure 1. Its inputs are a set $S \in \mathcal{L}$ together with a list of diameters $(w_1, \ldots, w_k)$ for some $k \geq 1$. The output is then a list of pairwise disjoint subsets $(S_1, \ldots, S_k)$ of S with $\mu(S_i) = w_i$ for all $i \in [k]$. This is possible precisely if the stated requirements for the diameters w_i are fulfilled.

Input: $S \in \mathcal{L}, (w_1, \ldots, w_k), w_i \in [0,1), i \in [k]$ with $w_1 + \ldots + w_k \leq \mu(S)$
Output: $(S_1, \ldots, S_k)$ with $S_i \subseteq S$ for $i \in [k]$, $S_i \cap S_j = \emptyset$ for $i, j \in [k]$ with $i \neq j$

function Match$(S, (w_1, \ldots, w_k))$
    $t_0 \leftarrow 0$
    for $r = 1, \ldots, k$ **do**
        $t_r \leftarrow \min\{t \in S \mid \mu(S \cap [t_{r-1}, t]) = w_r\}$
        $S_r \leftarrow S \cap [t_{r-1}, t_r)$
    end for
    return $(S_1, \ldots, S_k)$
end function

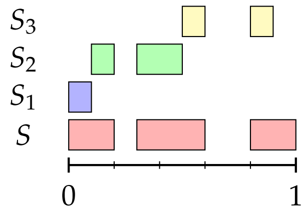


Figure 1: Subroutine Match (left) and exemplary output for defining three subsets of some set S (right)

For the remaining two routines, which are shown in Figure 2, we consider the graph G, its dependency graph $\mathcal{G}$ and the (arbitrary) point $h \in H$ whose membership in $P(G,\mathcal{I})$ shall be verified as global variables, i.e. they are known everywhere. The same holds for the sets required for Theorem 3.4, which become known globally once defined via the symbol :=. In contrast, we define auxiliary sets, which are local to a routine, via the symbol $\leftarrow$.

Function Define-Sets is the main routine of the algorithm. It iterates over each tree in $\mathcal{G}$ and defines the corresponding sets for the nodes and edges independently, which is possible due to Theorem 3.4. For the current tree, the first step is to choose an arbitrary root node $j \in Y$ and to define the corresponding set as $S_j := [0, h_j)$. Then we perform depth-first search by calling Traverse-Tree for all edges $\{j, c\}$ of $\mathcal{G}$ with $c \in N(j)$. This subroutine has two input parameters: a parent node p and a child node c (relative to the chosen root node). It assumes that the sets for p have already been defined and constructs the sets for the node c as well as for the edges between c and p. To define these sets, we need several calls to the function Match and need to argue each time why its input requirements are satisfied for $h \in H$. Clearly, inequalities (4) ensure that the inputs are non-negative. The function starts by differentiating two cases, namely $c \in \mathcal{I}$ or $c \in Y$.

In the first case, it fixes some order of the nodes in c and performs Match M1, which directly defines the sets for the edges between the nodes in c and p. Note that inequalities (6), which hold for h, ensure that the requirements for M1 are satisfied. From Theorem 3.4, condition (iv), we know that for any node $i \in c$ connected to node p the corresponding sets have to fulfil $S_i \cap S_p = S_{ip}$. As S_{ip} has already been defined via M1, part of the set S_i is thereby already known. The other part is given by the auxiliary set $\hat{S}_{i_l}$, which is determined by Match M2 over the complement of S_p, where inequality (7) ensures its requirements. This way, all sets for the nodes in c as well as their connecting edges to p have been defined and we can go further down the tree.

For the second case, $c \in Y$, Traverse-Tree performs Match M3 in a loop to define the sets

function DEFINE-SUBSETS
  for each tree in $\mathcal{G}$ **do**
    Pick some $j \in Y$ as the root node w.l.o.g.
    $S_j := [0, h_j)$.
    for $c \in N(j)$ **do**
      TRAVERSE-TREE(j, c)
    end for
  end for
end function
function TRAVERSE-TREE(p, c)
  if $c \in \mathcal{I}$ **then**
    Let $(i_1, \dots, i_k)$ be any fixed order of the nodes in c
    $(S_{i_1 p}, \dots, S_{i_k p}) := \text{MATCH}(S_p, (h_{i_1 p}, \dots, h_{i_k p}))$ ▷ M1, sets for the edges
    $(\hat{S}_{i_1}, \dots, \hat{S}_{i_k}) \leftarrow \text{MATCH}(\overline{S_p}, (h_{i_1} - h_{i_1 p}, \dots, h_{i_k} - h_{i_k p}))$ ▷ M2
    $(S_{i_1}, \dots, S_{i_k}) := (\hat{S}_{i_1} \cup S_{i_1 p}, \dots, \hat{S}_{i_k} \cup S_{i_k p})$ ▷ sets for the nodes
  else ▷ $c \in Y$
    for $i \in p$ **do**
      $S_{ic} := \text{MATCH}(S_i, (h_{ic}))$ ▷ M3, sets for the edges
    end for
    $\hat{S}_c \leftarrow \text{MATCH}(\overline{\cup_{i \in p} S_i}, (h_c - \sum_{i \in p} h_{ic}))$ ▷ M4
    $S_c := \bigcup_{i \in p} S_{ic} \cup \hat{S}_c$ ▷ set for the node c
  end if
  for $r \in N(c)$ **do**
    TRAVERSE-TREE(c, r)
  end for
end function

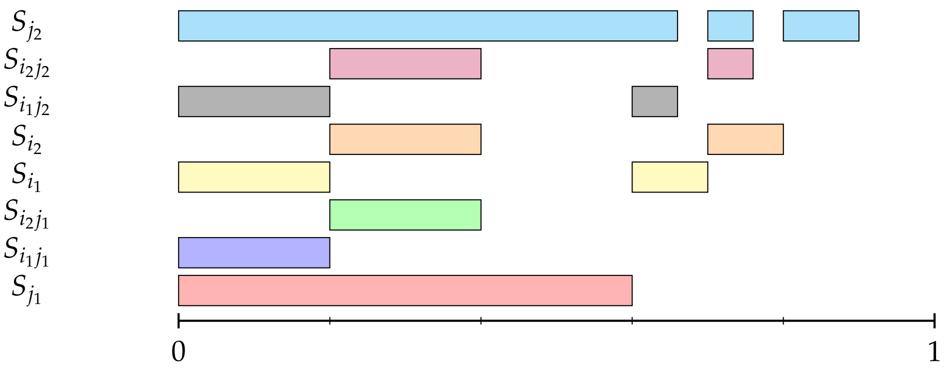


Figure 2: Routines DEFINE-SETS and TRAVERSE-TREE (top) and exemplary construction of the intervals for the complete bipartite graph G with $X = \{i_1, i_2\}$ and $Y = \{j_1, j_2\}$ as well as the partition $\mathcal{I} = X$ (bottom). The dependency graph $\mathcal{G}$ of G has node set $\mathcal{I}$ and edges $\mathcal{E} = \{\{\mathcal{I}, j_1\}, \{\mathcal{I}, j_2\}\}$. The algorithm starts by defining the set S_{j_1}, constructs the intermediate sets for edges as well as the nodes in $\mathcal{I}$ and terminates after the definition of S_{j_2}. The so-defined sets satisfy the conditions in Theorem 3.4, especially (iv) and (v), where in this case the former reads $S_{i_1} \cap S_{i_2} = \emptyset$ while the latter is $S_{i_q} \cap S_{j_p} = S_{i_q j_p}$ for all $q = 1, 2$ and $p = 1, 2$.

for the edges between p and c, by which the set S_c is already partly determined, similar as in the first if-branch. The remaining part of set S_c is finally defined by Match M4 within the complement of $\cup_{i\in p} S_{ic}$. Matches M3 and M4 are possible due to (6) and (7). Again, all required sets corresponding to p, c and the edges in between them have been determined and we can continue recursively with c and its children.

Whenever we come across an isolated node $j \in J$, (1) ensures $S_j \subseteq [0, 1)$. For isolated nodes $i \in I$ for some $I \in \mathcal{I}$, the subset-uniformity of the graph implies that all other nodes in I are isolated as well. Thus, the sets corresponding to the nodes in I can simply be placed next to each other without overlap, and inequalities (2) and (3) together ensure $\sum_{i\in I} \mu(S_i) \leq 1$.

By construction, all the defined sets satisfy the requirements of Theorem 3.4, which finishes the proof. □

The above proof also yields an alternative way to show [Pad89, Proposition 8], which states that the RLT inequalities are sufficient to completely describe QP on cycle-free graphs. Furthermore, the following corollary gives another case where the RLT inequalities suffice to completely describe $P(G, \mathcal{I})$.

Corollary 3.6 (Complete description on graphs with one subset)**.** *Let G be an arbitrary bipartite graph, and let $I = \{X\}$, then the polytope H as defined in Theorem 3.5 fulfils $H = P(G, \mathcal{I})$.*

The proof of Theorem 3.6 is similar to that of Theorem 3.5. The sets for X are chosen adjacent to each other, starting from 0. Then for each $\{i, j\} \in E$, h_{ij} is matched onto the set S_i. Finally, for $j \in Y$ the surplus $h_j - \sum_{\{i,j\}\in E} h_{ij}$ is matched onto $\overline{\cup_{\{i,j\}\in E} S_i}$.

3.2 Lifted facets from BQP facets

In this section, we describe classes of facets of $P(G, \mathcal{I})$ which are inherited from BQP. By applying the switching operation from Theorem 2.3 and the copy operation from Theorem 2.8, we will be able to lift them, which enables us to produce large classes of new facets as well.

3.2.1 Cycle inequalities

A well-known class of facets for QP are the cycle inequalities, which were introduced in [Pad89]. The 0-lifting of a *basic cycle inequality* to $P(G, \mathcal{I})$ can be stated as:

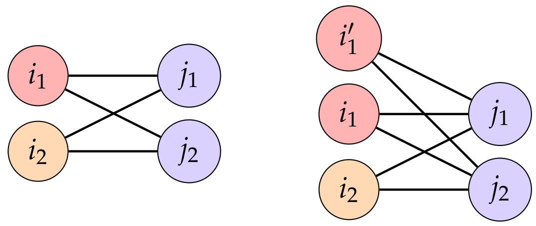
$$-z_{i_1 j_1} + z_{i_1 j_m} + \sum_{p=2}^{m} \left(-y_{j_p} - x_{i_p} + z_{i_p j_{p-1}} + z_{i_p j_p} \right) \leq 0, \tag{8}$$

for each cycle $\{\{i_1, j_1\}, \{j_1, i_2\}, \{i_2, j_2\}, \ldots, \{i_m, j_m\}, \{j_m, i_1\}\} \subseteq E$ of length $2m$ in G for some m, where $i_1, \ldots, i_m$ are from different subsets of the partition $\mathcal{I}$. Note that only even cycles are possible, because G is a bipartite graph. Furthermore, to obtain facets we only need to consider cycles which touch at most one node per subset. Otherwise, (8) is the sum of multiple cycle inequalities fulfilling this property.

The basic cycle inequalities (8) together will all inequalities obtained from them via switchings are commonly subsumed under the name *cycle inequalities*, a notion which we adopt as well. The new copy operation further enlarges this facet class to include those induced by

$$\sum_{i\in S_1} \left(-z_{ij_1} + z_{ij_m} \right) + \sum_{p=2}^{m} \left(-y_{j_p} + \sum_{i\in S_p} \left(-x_i + z_{ij_{p-1}} + z_{ij_p} \right) \right) \leq 0, \tag{9}$$

for each simple, chordless cycle $(\{I_1, j_1\}, \{j_1, I_2\}, \{I_2, j_2\}, \ldots, \{I_m, j_m\}, \{j_m, I_1\}) \subseteq \hat{E}$ in $\mathcal{G}$ of size $2m$ for some m and for all non-empty subsets $S_1 \subseteq I_1, \ldots, S_m \subseteq I_m$. These inequalities define facets if the cycle is chordless, as we will see in Theorem 3.7, otherwise they can be split



$$
\left(\begin{array}{cc|cccc}
 & & y_{j_1} & y_{j_2} & \cdots & y_{j_m} \\
 & & & -1 & \cdots & -1 \\
\hline
x_{i_1} & & -1 & & & 1 \\
x_{i_2} & -1 & 1 & 1 & & \\
\vdots & \vdots & & \ddots & \ddots & \\
x_{i_m} & -1 & & & 1 & 1
\end{array}\right)
$$

Figure 3: Support graph of the shortest possible cycle inequality (8) induced by four nodes $\{i_1, j_1, i_2, j_2\}$ with $i_1, i_2 \in X$ from different subsets in $\mathcal{I}$ and $j_1, j_2 \in Y$ (left). Support graph of the shortest possible cycle+copying inequality (9) induced by the four nodes from before and an additional node $i'_1 \in X$ from the same subset as i_1 (middle). The support of a basic cycle inequality in matrix form (right).

into two shorter cycle inequalities. We refer to inequalities (9) as well as all their switchings as the *cycle+copying inequalities*. In Figure 3, the support of these inequalities is also shown in matrix form, a notation we adopt from [CG04]. The values in the second column and the second row of the matrix are the coefficients of the involved x- and y-variables respectively, and the lower right matrix contains the coefficients of the corresponding z-variables.

Theorem 3.7 (Cycle+copying facets)**.** *The cycle+copying inequalities* (9) *define facets for* $P(G, \mathcal{I})$ *if the underlying cycle is chordless.*

Proof. Let the subset $C := (\{I_1, j_1\}, \{j_1, I_2\}, \{I_2, j_2\}, \ldots, \{I_m, j_m\}, \{j_m, I_1\}) \subseteq \hat{E}$ be a chordless cycle of size $2m$ in $\mathcal{G}$ for some $m \geq 2$, and let the subsets $S_1 \subseteq I_1, \ldots, S_m \subseteq I_m$ be non-empty. If $|S_1| = \ldots = |S_m| = 1$, (9) is the basic cycle inequality for $BQP(G)$, and its validity can be inferred from Theorem 2.3. If $|S_p| > 1$ for some $p \in [m]$, validity follows from Theorem 2.8.

Now, let $a^T(x, y, z) \leq b$ be a facet-defining inequality for $P(G, \mathcal{I})$ which contains the face F induced by (9). We then choose arbitrary representatives $i_1 \in I_1, \ldots, i_m \in I_m$ and show that a and b are multiples of the coefficients of inequality (9) by constructing $|X| + |Y| + |E|$ affinely independent points on F via Algorithm 1.

The copy operation produces different possibilities for the remaining coefficients. Indeed, for each $i \in I_r$, $r = 1, \ldots, m$, there are two possibilities: either all coefficients associated with i (for the x- and z-variables) are 0, then similar steps as 22–34 are necessary, or they are identical to the coefficients of the i_r, which can be seen by replacing i with i_r in Algorithm 1. This proves the claim. □

We remark that if the graph is not subset-uniform, inequality (9) can be defined accordingly in terms of cycles of the original graph, and the sets $S_1, \ldots, S_m$ may only contain nodes with the same neighbourhood.

For the facet classes of $P(G, \mathcal{I})$ we derive in the following, we assume G to be a complete bipartite graph to allow for a simpler presentation.

3.2.2 I_{mm22} Bell inequalities

A second prominent class of valid inequalities for BQP are the I_{mm22} Bell inequalities, see e.g. [AI07, CG04, WW01]. Their 0-lifted version can be stated as

$$
-y_{j_1} - \sum_{k=1}^{m} (m-k) x_{i_k} - \sum_{\substack{2 \leq p,k \leq m: \\ p+k=m+2}} z_{i_p j_k} + \sum_{\substack{1 \leq p,k \leq m: \\ p+k<m+2}} z_{i_p j_k} \leq 0, \tag{10}
$$

for pairwise distinct $I_1, \ldots, I_m \in \mathcal{I}$, $i_1 \in I_1, \ldots, i_m \in I_m$ as well as pairwise distinct $j_1, \ldots, j_m \in Y$ for some $m \geq 2$, cf. Figure 4. These inequalities induce facets as well.

Algorithm 1 Construction of affinely independent points on a cycle facet

From $0 \in F$ follows $b = 0$
From $e_{j_1} \in F$ follows $a_{j_1} = 0$
From $e_{i_1} \in F$ follows $a_{i_1} = 0$
Let $u \leftarrow e_{j_1}$
for $h = 2, \ldots, m$ **do**
  $u \leftarrow u + e_{i_h} + e_{i_h j_{h-1}}$ ▷ u could not be on F if the cycle had a chord
  From $u \in F$ follows $a_{i_h} = -a_{i_h j_{h-1}}$
  $u \leftarrow u + e_{j_h} + e_{i_h j_h}$
  From $u \in F$ follows $a_{j_h} = -a_{i_h j_h}$
end for
Let $v \leftarrow e_{i_1} + e_{j_m} + e_{i_1 j_m}$
From $v \in F$ follows $a_{j_m} = -a_{i_1 j_m}$
for $h = m, \ldots, 2$ **do**
  $v \leftarrow v + e_{i_h} + e_{i_h j_h}$
  From $v \in F$ follows $a_{i_h} = -a_{i_h j_h}$
  $v \leftarrow w + e_{j_{h-1}} + e_{i_h j_{h-1}}$
  if $h > 2$ **then**
    From $v \in F$ follows $a_{j_{h-1}} = -a_{i_h j_{h-1}}$
  end if
end for
From $v + e_{i_1 j_1} \in F$ follows $a_{i_1 j_1} = -a_{i_2 j_1}$
Let $\hat{Y} \leftarrow Y \setminus \{j_1, \ldots, j_m\}$ and $\hat{I} = \mathcal{I} \setminus \{I_1, \ldots, I_m\}$
From $e_j \in F$ follows $a_j = 0$ for all $j \in \hat{Y}$
From $e_i \in F$ follows $a_i = 0$ for all $i \in I$ with $I \in \hat{I}$
From $e_j + e_i + e_{ij} \in F$ follows $a_{ij} = 0$ for all $\{i, j\} \in E$ with $i \in I$, $I \in \hat{I}, j \in \hat{Y}$
From $e_{j_1} + e_i + e_{ij_1} \in F$ follows $a_{ij_1} = 0$ for all $\{i, j_1\} \in E$ with $i \in I$, $I \in \hat{I}$
From $e_j + e_{i_1} + e_{i_1 j} \in F$ follows $a_{i_1 j} = 0$ for all $\{i_1, j\} \in E$ with $j \in \hat{Y}$
Let $w \leftarrow e_{j_1}$
for $h = 2, \ldots, m$ **do**
  $w \leftarrow w + e_{i_h} + e_{i_h j_{h-1}}$
  From $w + e_j + e_{i_h j} \in F$ follows $e_{i_h j} = 0$ for all $\{i_h, j\} \in E$ with $j \in \hat{Y}$
  $w \leftarrow w + e_{j_h} + e_{i_h j_h}$
  From $w + e_i + e_{ij_h} \in F$ follows $e_{ij_h} = 0$ for all $\{i, j_h\} \in E$ with $i \in I$, $I \in \hat{I}$
end for

Theorem 3.8 (I_{mm22} Bell facets)**.** *Inequalities* (10) *define facets of* $P(G, \mathcal{I})$ *if* G *is a complete bipartite graph.*

Proof. From Theorem 3.6, we know that inequalities (10) define facets if all subsets of the partition contain only one element. To prove the claim via induction, we show that if we increase one subset $I \in \mathcal{I}$ by one new node i, the corresponding extension of this inequality still defines a facet. Let F be the face defined by this extended inequality. The extensions of the vertices of the original facet also lie on F. Thus, we only need to construct $(|Y| + 1)$-many additional affinely independent points on F. We can start by choosing $e_i \in F$ and $e_i + e_j + e_{ij} \in F$ for $j \in Y \setminus \{j_1\}$. In case $I \neq I_m$, we can next use $e_i + e_{i_m} + e_{j_1} + e_{ij_1} + e_{i_m j_1} \in F$. Otherwise, we can select $e_i + e_{i_{m-1}} + e_{j_2} + e_{j_1} + e_{ij_1} + e_{i_{m-1} j_1} + e_{ij_2} + e_{i_{m-1} j_2} \in F$, concluding the proof. □

The above proof also works if the support of the I_{mm22} Bell inequality to be lifted is a complete bipartite subgraph of G.

Note that we have only considered one of the two possible valid inequalities for $P(G, \mathcal{I})$

$$
\left(\begin{array}{cc|cccc}
 & & y_{j_1} & y_{j_2} & \cdots & y_{j_m} \\
 & & -1 & & & \\
\hline
x_{i_1} & -(m-1) & 1 & 1 & \dots & 1 \\
x_{i_2} & -(m-2) & 1 & \iddots & \iddots & -1 \\
\vdots & \vdots & \vdots & \iddots & \iddots & \\
x_{i_m} & 0 & 1 & -1 & &
\end{array}\right)
$$

Figure 4: Support of the I_{mm22} Bell inequalities

which could be derived from the original I_{mm22} Bell inequalities for BQP. A similar proof as above also works for swapped coefficients of x and y in inequality (10).

Further facets from BQP We have shown exemplarily for two classes of facet-defining inequalities for BQP that they can be 0-lifted to produce facets for $P(G, \mathcal{I})$. For the Bell inequalities, the corresponding proof requires only a simple extension of the original proof for BQP. Furthermore, we have seen that the copy operation allows us to significantly enlarge a given inequality class. All the resulting inequalities can be facet-defining, as was the case for the cycle inequalities. However, proving this fact was much more involved, as the necessary affinely independent points on the facet needed to be constructed from scratch. Our computational experiments with facet enumeration tools like *polymake* (see [GJ00]) for small instances hint that there may be many more facet classes which $P(G, \mathcal{I})$ inherits from BQP, and their variants obtained via copying could be facet-defining as well.

In [AIIS05], the authors introduce a technique called *triangular elimination*, which transforms a facet of the cut polytope on an appropriate graph to a facet of BQP defined on the complete bipartite graph (see [AII08] for the non-complete case). The cut polytope is very well investigated, and there are many known facet classes for it, like the hypermetric, clique-web and parachute inequalities and many more (see [DLW97]). The triangular eliminations of these inequalities give rise to a rich pool of facet classes for BQP, for which it could be tested as well if they are 0-liftable to facets for $P(G, \mathcal{I})$ and if their copyings define facets, too.

3.3 Novel facets for $P(G, \mathcal{I})$

Besides the facets inherited from BQP, the polytope $P(G, \mathcal{I})$ has a large variety of facets specifically induced by the multiple-choice constraints. In the following, we introduce a very rich template of cutting planes for $P(G, \mathcal{I})$, which we call the $1, m$-inequalities. As two special cases of this inequality, we present the facet-inducing arrow-1 and arrow-2 inequalities, whose switchings and copyings are facet-defining as well.

3.3.1 $1, m$-inequalities

We define the class of *$1, m$-inequalities* as

$$a_{i_1} x_{i_1} - \sum_{p=1}^{k} a_{j_p} y_{j_p} + \sum_{p=1}^{m} a_{i_1 j_p} z_{i_1 j_p} + \sum_{h=2}^{m} \sum_{p=1}^{m} a_{i_h j_p} z_{i_h j_p} \geq 0, \tag{11}$$

with distinct $I_1, I_2 \in \mathcal{I}$, $i_1 \in I_1$, pairwise distinct $i_2, \dots, i_m \in I_2$ as well as pairwise distinct $j_1, \dots, j_m \in Y$ for some $m \geq 3$ and $1 \leq k \leq m$. The following conditions guarantee their validity for $P(G, \mathcal{I})$.

Theorem 3.9. *Inequality* (11) *is valid and supporting for* $P(G, \mathcal{I})$ *if its coefficients satisfy all of the following conditions:*

$$
\left(\begin{array}{cc|cccc}
 & & y_{j_1} & y_{j_2} & \cdots & y_{j_m} \\
 & & 1 & & & \\
\hline
x_{i_1} & m-1 & -1 & -1 & \cdots & -1 \\
x_{i_2} & & -1 & 1 & & \\
\vdots & & \vdots & & \ddots & \\
x_{i_m} & & -1 & & & 1
\end{array}\right)
\qquad
\left(\begin{array}{cc|cccc}
 & & y_{j_1} & y_{j_2} & \cdots & y_{j_m} \\
 & & & 1 & \cdots & 1 \\
\hline
x_{i_1} & 1 & -1 & -1 & \cdots & -1 \\
x_{i_2} & & 1 & -1 & & \\
\vdots & & \vdots & & \ddots & \\
x_{i_m} & & 1 & & & -1
\end{array}\right)
$$

Figure 5: Support of the arrow-1(left) and arrow-2(right) inequalities

(*i*) $a_{i_1 j_p} = -1, \quad \forall p = 1, \ldots, m$

(*ii*) $a_{j_p} = 1, \quad \forall p = 1, \ldots, k$

(*iii*) $a_{i_1} = m - k$

(*iv*) $a_{i_h j_p} \in \{0, -1\}, \quad \forall p = 1, \ldots, k, \quad \forall h = 2, \ldots, m$

(*v*) $a_{i_h j_p} \in \{0, 1\}, \quad \forall p = 1, \ldots, k, \quad \forall h = 2, \ldots, m$

(*vi*) $\sum_{p=1}^{k} a_{i_h j_p} + \sum_{p \in R} \left(a_{i_h j_p} - 1 \right) \geq -(m-k), \quad \forall h = 2, \ldots, m, \quad \forall R \subseteq \{k+1, \ldots, m\}$

Proof. Let r be a vertex of $P(G, \mathcal{I})$. We distinguish the following four exhaustive cases: for $r_{i_1} = 0$ and $r_{i_h} = 0$ for all $h = 2, \ldots, m$, the inequality is trivially satisfied. For $r_{i_1} = 1$ and $r_{i_h} = 0$ for all $h = 2, \ldots, m$, conditions (i)–(iii) ensure validity. For $r_{i_1} = 0$ and $r_{i_h} = 1$ for some $h \in \{2, \ldots, m\}$, it is ensured by conditions (ii), (iv) and (v). For $r_{i_1} = 1$ and $r_{i_h} = 1$ for some $h \in \{2, \ldots, m\}$, the inequality is valid by conditions (i)–(vi). Furthermore, the inequality is always binding for the 0-vector. □

We now present two special cases in which the $1, m$-inequalities define facets.

3.3.2 Arrow-1 inequalities

The first class of facets is defined by what we call the *arrow-1 inequalities*:

$$(m-1)x_{i_1} + y_{j_1} - \sum_{p=1}^{m} z_{i_1 j_p} + \sum_{p=2}^{m} \left(-z_{i_p j_1} + z_{i_p j_m} \right) \geq 0, \tag{12}$$

for distinct $I_1, I_2 \in \mathcal{I}$, $i_1 \in I_1$, pairwise distinct $i_2, \ldots, i_m \in I_2$ and pairwise distinct $j_1, \ldots, j_m \in Y$ with $m \geq 3$. Their support is depicted in Figure 5 on the left. As these inequalities arise from the $1, m$-inequalities by setting $k = 1$, we know from Theorem 3.9 that they are valid for $P(G, \mathcal{I})$. The next result shows that they are even facet-defining.

Theorem 3.10 (Arrow-1 facets)**.** *Inequality* (12) *as well as all its copyings define facets of* $P(G, \mathcal{I})$ *if* G *is a complete bipartite graph.*

Proof. We fix some distinct $I_1, I_2 \in \mathcal{I}$, a $i_1 \in I_1$ as well as pairwise distinct $i_2, \ldots, i_m \in I_2$ and pairwise distinct $j_1, \ldots, j_m \in Y$ for some $m \geq 3$. Let $a^T(x, y, z) \leq b$ be a facet-defining inequality which contains the face F induced by (12). In Algorithm 2, we indicate how to construct $|X| + |Y| + |E|$ affinely independent points on F to show that a and b are multiples of the coefficients of (12).

Copying lifts further variables into the inequality. For each $i \in I_1 \setminus \{i_1\}$, there are two possibilities: either all coefficients associated with i (for the x- and z-variables) are 0, then similar steps as 12–18 need to be done, or they are identical to those of i_1, then the construction can be

Algorithm 2 Construction of affinely independent points on an arrow-1 facet

From $0 \in F$ follows $b = 0$
From $e_{j_p} \in F$ follows $a_{j_p} = 0$ for all $p = 2, \dots, m$
From $e_{i_h} \in F$ follows $a_{i_h} = 0$ for all $h = 2, \dots, m$
From $e_{j_1} + e_{i_h} + e_{i_h j_1} \in F$ follows $a_{i_h j_1} = -a_{j_1}$ for all $h = 2, \dots, m$
From $e_{j_1} + e_{j_p} + e_{i_h} + e_{i_h j_1} + e_{i_h j_p} \in F$ follows $a_{i_h j_p} = 0$ for all $h = 2, \dots, m$ and $p = 2, \dots, m$ with $h \neq p$
From $e_{j_1} + e_{j_p} + e_{i_p} + e_{i_l} + e_{i_p j_1} + e_{i_l j_1} + e_{i_p j_p} + e_{i_l j_p} \in F$ for some arbitrary $l \in \{2, \dots, m\} \setminus p$ follows $a_{i_p j_p} = a_{j_1}$ for all $p = 2, \dots, m$
Let $v \leftarrow e_{j_1} + e_{i_1} + e_{i_1 j_1} + \sum_{h=2}^{m} (e_{i_h} + e_{i_h j_1})$
From $v \in F$ and $v + e_{j_p} + e_{i_1 j_p} + \sum_{r=2}^{m} e_{i_r j_p} \in F$ follows $a_{i_1 j_p} = -a_{j_1}$ for all $p = 2, \dots, m$
From $e_{i_1} + \sum_{h=2}^{m} \left(e_{j_h} + e_{i_1 j_h}\right) \in F$ follows $a_{i_1} = (m-1) a_{j_1}$
From $e_{i_1} + \sum_{h=1}^{m} \left(e_{j_h} + e_{i_1 j_h}\right) \in F$ follows $a_{i_1 j_1} = -a_{j_1}$
Let $\hat{Y} \leftarrow Y \setminus \{j_1, \dots, j_m\}$ and $\hat{I} \leftarrow \mathcal{I} \setminus \{I_1, I_2\}$
From $e_j \in F$ follows $a_j = 0$ for all $j \in \hat{Y}$
From $e_i \in F$ follows $a_i = 0$ for all $i \in I$ with $I \in \hat{I}$
From $e_{j_p} + e_i + e_{i j_p} \in F$ follows $a_{i j_p} = 0$ for all $i \in I$ with $I \in \hat{I}$ and $p = 2, \dots, m$
From $e_j + e_i + e_{ij} \in F$ follows $a_{ij} = 0$ for all $i \in I$ with $I \in \hat{I}$ and $j \in \hat{Y}$
From $e_{j_1} + e_{i_2} + e_{i_2 j_1} + e_i + e_{i j_1} \in F$ follows $a_{i j_1} = 0$ for all $i \in I$ with $I \in \hat{I}$
From $e_{i_1} + \sum_{r=2}^{m} \left(e_{j_r} + e_{i_1 j_r}\right) + e_j + e_{i_h j} \in F$ follows $a_{i_1 j} = 0$ for all $j \in \hat{Y}$
From $e_{i_h} + e_j + e_{i_h j} \in F$ follows $a_{i_h j} = 0$ for all $h = 2, \dots, m$ and $j \in \hat{Y}$

adapted by replacing i with i_1 in Algorithm 2. For $i \in I_2 \setminus \{i_2, \dots i_m\}$, there are m possibilities: again, either all coefficients are 0, or they coincide with those corresponding to some node from $k \in \{i_2, \dots, i_m\}$, in which case we can replace i with k in Algorithm 2. Therefore, the arrow-1 inequalities and all their copyings define facets. □

3.3.3 Arrow-2 inequalities

The second class of facets can be stated as

$$x_{i_1} + \sum_{p=2}^{m} y_{j_p} - \sum_{p=1}^{m} z_{i_1 i_p} + \sum_{p=2}^{m} \left(z_{i_p j_1} - z_{i_p j_p}\right) \geq 0, \tag{13}$$

for distinct $I_1, I_2 \in \mathcal{I}$, $i_1 \in I_1$, pairwise distinct $i_2, \dots, i_m \in I_2$, pairwise distinct $j_1, \dots, j_m \in Y$ and some $m \geq 3$. The support of these *arrow-2 inequalities* is shown in Figure 5 to the right. Their validity follows from Theorem 3.9 by setting $k = m - 1$, and, similar as before, they are facet-defining.

Theorem 3.11 (Arrow-2 facets)**.** *Inequality* (13) *as well as all its copyings define facets of $P(G, \mathcal{I})$ if G is a complete bipartite graph.*

The proof is given in Appendix A.

4 Separation algorithms

For several of the facet classes derived in Section 3, we are able to give efficient separation algorithms. Each of these separation routines consists of an enumeration part over subsets of Y, where in each iteration an auxiliary combinatorial optimization problem has to be solved. We also discuss how these algorithms can be modified in order to separate over all switchings and copyings as well and what this means for their complexity. For ease of exposition, we state all separation routines for the case of a complete bipartite graph G.

4.1 Separation of cycle inequalities

A well-known polynomial-time separation algorithm for the cycle inequalities for QP is based on shortest-path-computations and can be found in [DLW97]. As the cycle inequalities for $QP(G)$ remain valid for $P(G, \mathcal{I})$, this algorithm can still be used for separation; however, not all cycle facets of $QP(G)$ are still facets of $P(G, \mathcal{I})$. For a complete bipartite graph, only cycles of size four define facets and they dominate all longer cycle inequalities. In Algorithm 3, we show how to separate over all cycle inequalities of size four and their copyings.

Algorithm 3 Separation of cycle+copying inequalities

Input: $(x, y, z) \in \mathbb{R}^{X \cup Y \cup E}$
Output: Most violated cycle+copying inequality for each $j_1, j_2 \in Y$ with $j_1 \neq j_2$.
1: **function** CYCLE+COPYING-SEPARATOR((x, y, z))
2:   **for** each distinct $j_1, j_2 \in Y$ **do**
3:     **for** $I \in \mathcal{I}$ **do**
4:       **for** $i \in I$ **do**
5:         $v_i^1 \leftarrow z_{ij_1} - z_{ij_2}$, $v_i^2 \leftarrow -x_i + z_{ij_1} + z_{ij_2}$
6:       **end for**
7:       $S_I^1 \leftarrow \{i \in I : v_i^1 > 0\}$, $S_I^2 \leftarrow \{i \in I : v_i^2 > 0\}$
8:       **if** $S_I^1 = \emptyset$ **then**
9:         $p \leftarrow \operatorname{argmax}\{v_i^1 \mid i \in I\}$
10:         $S_I^1 \leftarrow \{p\}$
11:       **end if**
12:       **if** $S_I^2 = \emptyset$ **then**
13:         $p \leftarrow \operatorname{argmax}\{v_i^2 \mid i \in I\}$
14:         $S_I^2 \leftarrow \{p\}$
15:       **end if**
16:       $\phi_I^1 \leftarrow \sum_{i \in S_I^1} v_i^1$, $\phi_I^2 \leftarrow \sum_{i \in S_I^2} v_i^2$
17:     **end for**
18:     Select distinct $I_1, I_2 \in \mathcal{I}$ which maximize $\phi_{I_1}^1 + \phi_{I_2}^2$
19:     **if** $\phi_{I_1}^1 + \phi_{I_2}^2 - y_{j_1} > 0$ **then**
20:       **return** violated cycle inequality based on $(j_1, j_2, I_1, I_2, S_{I_1}^1, S_{I_2}^2)$
21:     **end if**
22:   **end for**
23: **end function**

A cycle+copying inequality is uniquely defined by choosing distinct $j_1, j_2 \in Y$ distinct $I_1, I_2 \in \mathcal{I}$, an $S^1 \subseteq I_1$ and an $S^2 \subseteq I_2$. Thus, there are exponentially many cycle inequalities with copying. However, once j_1, j_2 are fixed (step 2), one can calculate for each $i \in I$ with $I \in \mathcal{I}$ separately the contribution to the left-hand side of the constraint for the two cases $i \in S^1$ and $i \in S^2$ (steps 3–6). Then one can select greedily the optimal subsets $S_{I_1}^1$ and $S_{I_1}^2$, one for each $I \in \mathcal{I}$ (steps 7–9) and choose the two subsets which lead to the maximal combined violation (step 12). In order to just separate over cycles with size four (without copying), step 7 in the algorithm needs to be left out and we choose the then-branch in steps 8 and 9; its complexity, however, does not decrease. The running time of the algorithm is of the order $\mathcal{O}(|Y|^2|X|)$, including the greedy suboptimization.

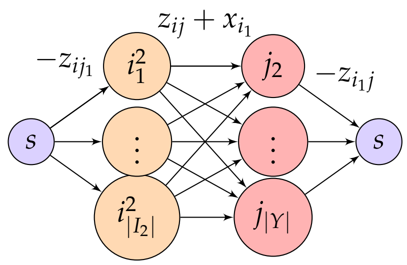
Algorithm 3 is written to separate over cycle inequalities of the form (9), but can easily be extended to separate over their switchings as well. Note that such a switching is simply a lower bound instead of an upper bound on the same left-hand-side expression. By swapping all inequality and optimization senses in steps 7–13, one can obtain the most violated switched cycle inequality simultaneously.

In Appendix D, we give a template for the separation of 0-lifted valid inequalities for BQP with bounded Y-support which includes Algorithm 3 as a special case.

4.2 Separation of arrow inequalities

In Algorithm 4, we present a routine to separate over all arrow-1 inequalities. It is again based on an enumeration part and an auxiliary subproblem in each iteration, which takes the form of a minimum-cost-circulation problem (MCCP) in this case.

For distinct $I_1, I_2 \in \mathcal{I}$ as well as some $i_1 \in I_1$ and $j_1 \in Y$, let $J := Y \setminus \{j_1\}$, let $W := I_2 \times J$, and let $H = (\tilde{V}, A)$ be a directed graph with node set $\tilde{V} := \{s\} \cup I_2 \cup J$. The arc set A contains the arcs (s, i) for $i \in I_2$, all $(i, j) \in W$ and (j, s) for $j \in J$. Each arc has a capacity of at most one. The graph H is shown in Figure 6.



$$c_a := \begin{cases} -z_{ij_1} & \text{if } a = (s, i) \text{ for } i \in I_2 \\ z_{ij} + x_{i_1} & \text{if } a = (i, j) \in W \\ -z_{i_1 j} & \text{if } a = (j, s) \text{ for } j \in J. \end{cases}$$

Figure 6: Graph H for $I_2 = \{i^2_1, \ldots, i^2_{|I_2|}\}$ and $J = \{j_2, \ldots, j_{|Y|}\}$ (left). The node s is duplicated in the figure for better readability. The unit costs for each arc (right).

Algorithm 4 Separation of arrow-1 inequalities

Input: $(x, y, z) \in \mathbb{R}^{X \cup Y \cup E}$
Output: Most violated arrow-1 inequality for distinct $I_1, I_2 \in \mathcal{I}$, $i_1 \in I_1$ and $j_1 \in Y$.

1: **function** ARROW-1-SEPARATOR(x, y, z)
2:   **for** each distinct $I_1, I_2 \in \mathcal{I}$, each $i_1 \in I_1$ and each $j_1 \in Y$ **do**
3:     Compute minimum-cost circulation on H.
4:     $o \leftarrow$ optimal value determined in step 3
5:     $\tilde{A} \leftarrow \{(i, j) \in W \mid (i, j)$ is part of the optimal circulation$\}$
6:     $\{j_2, \ldots j_m\} \leftarrow \{j \in J \mid (i, j) \in \tilde{A}\}$ ▷ naming the elements
7:     **for** $n \in \{2, \ldots, m\}$ **do**
8:       Let $i_n \in \{i \in I_2 \mid (i, j_n) \in \tilde{A}\}$ ▷ set contains only one element
9:     **end for**
10:     **if** $o + y_{j_1} - z_{i_1 j_1} < 0$ **then**
11:       **return** violated inequality based on $(I_1, I_2, i_1, j_1, j_2, \ldots j_m, i_2, \ldots, i_m)$
12:     **end if**
13:   **end for**
14: **end function**

An arrow-1 inequality is uniquely defined by choosing distinct $I_1, I_2 \in \mathcal{I}$, an $i_1 \in I_1$ and a $j_1 \in Y$ on the one hand as well as $i_2, \ldots, i_m \in I_2$ and $j_2, \ldots, j_m \in J$ on the other hand. The former possibilities are enumerated in step 2 of Algorithm 4, while the latter is determined via an MCCP in steps 5–12 of the algorithm.

The running time of the algorithm is $\mathcal{O}(|\mathcal{I}|^2 |Y| I^{\max} D(I^{\max}, |Y|))$, where $I^{\max} := \max\{|I| \mid I \in \mathcal{I}\}$ is the cardinality of the largest subset in $\mathcal{I}$ and $D(I^{\max}, |Y|)$ is the running time of the algorithm to solve the MCCP on H. Via the push-relabel algorithm, this is possible, for example, in $\mathcal{O}(|\tilde{V}|^2 \sqrt{|A|})$ (see [CM89]), where $|\tilde{V}| = I^{\max} + |Y| + 1$ and $|A| = |I^{\max}||Y| + |I^{\max}| + |Y|$.

Observe that we did not have to specify m in Algorithm 4, but that it is determined by the amount of the flow through node s.

The arrow-2 inequalities can be separated via a slight modification of the algorithm by changing the cost function for the MCCP to reflect the support of the arrow-2 inequalities. Finally, how to separate over the switchings and copyings of both types of arrow inequalities is described Appendix B and Appendix C respectively.

5 Computational results

In the following, we evaluate the computational benefit of our polyhedral findings and separation routines for $P(G,\mathcal{I})$. We first present an empirical study of the strength of the facets classes we have derived. Then we show that the resulting cutting planes lead to vastly improved performance in solving a novel type of pooling problem for instances based on real-world data.

For the computations, we used a server with Intel Xeon E5-2690v2 3.00 GHz processors, 128 GB RAM, 1 core and *Gurobi 9.0.2* ([Gur20]) as a quadratic solver.

5.1 Strength of relaxation of the facet classes

We consider the optimization problem $\max\{c^T(x,y,z) \mid (x,y,z) \in P(G,\mathcal{I})\}$ for some $c \in \mathbb{R}^{X \cup Y \cup E}$. The standard LP relaxation of this problem, which state-of-art solvers like *Gurobi* use, is given by

$$\max\left\{ c^T x \;\middle|\; \begin{array}{l} z_{ij} \geq 0 \quad \forall \{i,j\} \in E, \quad x_i + y_j - z_{ij} \leq 1 \quad \forall \{i,j\} \in E, \\ z_{ij} \leq x_i \quad \forall i \in X, \quad z_{ij} \leq y_j \quad \forall j \in Y, \quad x \in X^{\mathcal{I}} \end{array} \right\}. \tag{14}$$

Apart from the multiple-choice constraints, this relaxation includes the famous *McCormick*-inequalities, which are a weaker form of the RLT inequalities we derived. In this computational study, we investigate for several of the facet classes presented in Section 3 by how much they close the gap between integer and LP optimum. To this end, we generated 7 differently sized complete bipartite graphs G for which we solve the problem using 10 different random objective functions c each. The coefficients of c are sampled uniformly and independently from the interval $[-10, 10]$. As most of the derived facet classes are exponential in size, we separated them using the routines from Section 4 until no more violated inequalities were found. Only for the RLT inequalities, we directly added all of them from the start.

In Table 1, we state the resulting gaps averaged over all 10 instances with the same underlying graph. The first column displays the short names of the different classes: LP for the LP relaxation with no additional cuts, *RLT* for the RLT inequalities, *C* for the cycle inequalities, *A1* for the arrow-1 inequalities and *A2* for the arrow-2 inequalities. An additional *S* at the end stands for all inequalities that are obtained via the switching operation, and the same holds for *C* with the copy operation. For the row *All*, we first added all RLT inequalities and then ran the separation routines for CC, A1S, A1C, A2S and A2C from Section 4 in an iterative loop, inserting all violated inequalities of each class before resolving the relaxation until no more violated inequalities of any class were found. The top line in each column indicates the size of the instances. These are stated in the format $|\mathcal{I}|$-$|I|$-$|Y|$ with all $I \in \mathcal{I}$ being of the same cardinality. For example, 5-5-10 means 5 subsets in the partition with 5 nodes each and 10 nodes in Y. The instances of type 10-$*$-25 have a partition with 10 subsets of sizes $1, \ldots, 10$ and 25 nodes in Y.

We see that the CC inequalities have by far the highest impact in reducing the gap. On the first three instance classes, they are the only group that is able to completely close the gap. Furthermore, they are considerably stronger than the cycle inequalities without copying, which is notable because there is practically no additional effort in separating over the copyings as well. We remark that on the larger instances, we found about three times as many violated CC

Table 1: Average gaps in % between integer and LP optimum for instances of different size and for various facet classes added to the linear relaxation

	5-5-10	10-10-10	15-15-10	5-5-20	5-5-40	5-5-60	10-∗-25
LP	17.83	21.81	22.47	29.03	37.85	42.28	40.66
RLT	8.02	13.39	14.01	20.90	31.66	36.57	31.88
C	3.63	11.90	14.30	17.05	30.51	36.08	28.82
CC	0.00	0.00	0.00	0.00	1.99	4.43	4.05
A1	1.95	7.55	8.97	10.65	24.71	30.34	21.18
A1S	0.00	0.13	0.42	3.64	17.31	23.47	11.73
A1C	0.28	1.85	2.82	6.41	18.41	23.58	15.50
A2	9.52	16.85	18.14	22.82	34.12	38.87	33.51
A2S	3.00	11.66	14.30	17.05	30.51	36.08	27.23
A2C	0.31	2.74	4.13	7.40	19.78	24.88	17.20
All	0.00	0.00	0.00	0.00	0.45	1.48	1.64

inequalities as C inequalities. Note that C lies in the intersection of the classes A1S and A2S. While the A2S inequalities are empirically only marginally stronger than the C inequalities, the A1 inequalities perform much better than both other classes. This performance can be improved significantly by passing over to the larger set of the A1S inequalities. For the smaller instances, they are almost as strong as the CC inequalities; however, for large instances the latter are far superior. The RLT inequalities contribute notably to closing the gap for small instances and are still helpful for large instances, especially because they achieve this effect with relatively few cutting planes to be added. When we look at the strength of all inequalities taken together, we see that even for the largest instances there only remains a very small integrality gap. This means that the facet classes studied here already describe the polytope under consideration very well.

Summarizing, we conclude that the CC inequalities are the most important ones to separate, and, favourably, also the computationally easiest ones to separate. For larger instances, the other inequalities further help to reduce the gap. Indeed, in Appendix G we show that the CC and the A2C inequalities together are the best combination of any two types of cutting planes in our study. In Appendix F, we also show in detail how many cutting plans are required to obtain the results from Table 1.

5.2 Computational study on real-world pooling instances

We finally present a computational study on real-world instances of a special variant of the pooling problem, where we use some of the derived cuts to improve the solution process. First, we describe the underlying pooling model, then we give some details about the origin of the data. The results on our benchmark set will show that exploiting the multiple-choice structure in the problem yields a significant solution time benefit.

5.2.1 Pooling model

The pooling problem is a well-known continuous and non-convex optimization problem (see [GADC17] for a detailed introduction). From a practical point of view, it arises whenever different raw materials of varying quality need to be mixed over multiple production stages in order to obtain final products with given quality requirements. In the mathematical sense, it is a combination of the two famous problems minimum-cost flow and blending.

Typically, the pooling problem is represented over a directed graph $G = (V, A)$ with a node set $V = I \cup P \cup O$ which is split into a set of inputs I, pools P and outputs O as well as a set of quality specifications S. We assume that there exist no arcs between pools, which is the standard setting studied in the literature. For each $i \in I$ and $s \in S$, $\lambda_{is} \in \mathbb{R}_+$ denotes the value of specification s of input i. Further, for each output $o \in O$ and each specification $s \in S$, a minimum level $\mu_{os}^{\min} \in \mathbb{R}_+$ and a maximum level $\mu_{os}^{\max} \in \mathbb{R}_+$ are given. The maximum available amount of raw material at input $i \in I$ is given by $b_i \in \mathbb{R}_+$, and for $o \in O$, $d_o \in \mathbb{R}_+$ is the demand at output o. For each pool $l \in L$, $I_l \subseteq I$ shall denote the subset of inputs which have an arc pointing to l. The goal of pooling is then to send as much flow as possible from inputs to outputs through the graph while ensuring that the specification limits hold.

Up to this point, we have described the classical pooling problem from the literature. In our application, there are additional flow restrictions in the form of multiple-choice constraints, which correspond to the *recipes* used in tea production. They specify percentagewise how much of each raw material is needed to produce a given final product. This requirement leads to the following definitions: for each pool $l \in L$, let $\mathcal{I}_l := \{I_l^1, \dots, I_l^{r_l}\}$ be a partition of I_l with r_l-many subsets, where each subset contains only inputs of the same material. Moreover, we are given a factor $0 \leq \sigma_l^h \leq 1$ for each $l \in L$ and $h = 1, \dots, r_l$ stating the desired proportion of total flow from the associated inputs in $\mathcal{I}_l$ arriving at l. These factors satisfy $\sum_{h \in [r_l]} \sigma_l^h = 1$ for all $l \in L$. We now introduce the two model formulations which we will compare in our study.

q-Formulation for the pooling problem A well-known formulation for the pooling problem is the so-called *q-formulation* (see [GADC17]), which can easily be extended for the additional multiple-choice constraints. Let variable $y_{ij} \in \mathbb{R}_+$ be the flow on arc $a = (i,j) \in A$, and let variable $q_{il} \in \mathbb{R}_+$ be the proportion of flow from input $i \in I_l$ to pool $l \in L$. For notational simplicity, we will use the flow variables y_{ij} with the understanding that $y_{ij} \equiv 0$ if $(i,j) \notin A$. Furthermore, we define the auxiliary variables $v_{ilo} \in \mathbb{R}_+$ for $i \in I_l$, $l \in L$ and $o \in O$ for the product of y_{ij} and q_{il}. The pooling problem can then be modelled via the following constraints:

$$y_{il} \leq b_i \quad \forall i \in I, \forall l \in L, \quad y_{lo} \leq d_o \quad \forall o \in O, \forall l \in L, \tag{15}$$

$$\sum_{i \in I} y_{il} = \sum_{o \in O} y_{lo} \quad \forall l \in L, \tag{16}$$

$$\sum_{i \in I_l^h} y_{il} = \sigma_h \sum_{o \in O} y_{lo} \quad \forall l \in L, \forall h = 1, \dots, r_l, \tag{17}$$

$$\sum_{i \in I_l} q_{il} = 1 \quad \forall l \in L, \tag{18}$$

$$v_{ilo} = q_{il} y_{lo} \quad \forall l \in L, \forall i \in I_l, \forall o \in O, \tag{19}$$

$$y_{il} = \sum_{o \in O} v_{ilo} \quad \forall l \in L, \forall i \in I_l, \tag{20}$$

$$\mu_{os}^{\min} \sum_{i \in I \cup L} y_{io} \leq \sum_{i \in I} \lambda_{is} y_{io} + \sum_{l \in L} \sum_{i \in I_l} \lambda_{is} v_{ilo} \leq \mu_{os}^{\max} \sum_{i \in I \cup L} y_{io} \quad \forall o \in O, \forall s \in S. \tag{21}$$

Inequalities (15) specify the upper bounds on the flow variables, and equations (16) model the flow conservation. The novel recipe structure is modelled via (17). Inequalities (18) require that the proportion variables at each pool sum up to one. Constraint (19) defines the auxiliary variables to linearize the bilinear terms while constraint (20) ensures consistency between the q and y-variables. Finally, (21) demands that the specification bounds at the outputs be respected.

The aim is to maximize the total flow arriving at the outputs, which is realized via the objective function $\max \sum_{l \in L} \sum_{o \in O} y_{lo}$.

q+cuts formulation for the pooling problem The equations given by

$$\sum_{i \in I_l^h} q_{il} = \sigma_h \quad \forall l \in L, \forall h = 1, \ldots, r_l \tag{22}$$

are valid for the pooling problem (15)–(21), which can easily be checked by rearrangement of (17), (19) and (20). Observe that for each pool $l \in L$ as well as the corresponding subvectors q_l, y_l and v_l of q, y and v, the set

$$\{(q_l, y_l, v_l) \in \mathbb{R}_+^{I_l \times O \times (I_l \times O)} \mid (q_l, y_l, v_l) \text{ fulfils (15), (19) and (22)}\} \tag{23}$$

is a scaled and low-dimensional version of the boolean quadric polytope with multiple-choice constraints P on a complete bipartite graph. The low-dimensionality is due to (22) being an equation instead of an inequality.

In Appendix E, we derive the following valid equations for (23), which we call the RLT equations:

$$\sum_{i \in I_l^h} v_{ilo} = \sigma_h y_{lo} \quad \forall o \in O, l \in L, \forall h = 1, \ldots, r_l. \tag{24}$$

There we also show that valid constraints can easily be converted between the full-dimensional case and the low-dimensional case.

Note that for each pool $l \in L$, the set

$$\{(q_l, y_l, v_l) \in \mathbb{R}_+^{I_l \times O \times (I_l \times O)} \mid (q_l, y_l, v_l) \text{ fulfils (15), (18) and (19)}\} \tag{25}$$

also defines a low-dimensional instance of P. In this special case, where there is only a single multiple-choice constraint, the complete description for (25) is given by the corresponding RLT inequalities (see [Gup16, Theorem 1] as well as Theorem 3.6). Adding these RLT inequalities to (15)–(21) leads to the well-known *pq-formulation* for the pooling problem (see [GADC17]).

In our second formulation for the pooling problem with recipes, which we call *q+cuts*, we add to the q-formulation (15)–(21) the equations (22), the RLT equations (24) and the RLT inequalities for (23) for each pool. The q+cuts formulation can be seen as a *pq*-formulation for pooling with recipes, in the sense described above. We will demonstrate that it is vastly superior on our real-world instances when compared to the pure q-formulation.

5.2.2 Computational results

Based on real-world data from a tea producing company, we have created six instances of different sizes for the pooling problem with recipes introduced above. The amount of available raw material, the quality specifications at the inputs and typical levels for the upper and lower specification bounds at the outputs as well as the recipes were provided by our industry partner. Under this setting, we have simulated various possible customer request scenarios, for which we took the following assumptions. Each pool produces one final product, which is then sent to 5 different outputs (customers) with different specification limits and demands. From the provided reference quality levels, we derived 5 different sets of quality specifications for the different outputs by randomly decreasing the given lower bound and increasing the upper bound by up to 15% each. The demands were drawn uniformly and independently from the interval $[200, 4000]$, which yields typical demand values for this application. On average, the inputs had 6 measured quality specifications each while the outputs had an average of 190 specifications each.

In our experiments, we compared the performance of the q-formulation versus the q+cuts formulation within a time limit of 1 minute. The results are found in Table 2. We see that the q-formulation could solve only one out of the six instances within the time limit, while the

Table 2: Performance comparison for the q- and q+cuts formulation over 6 instances of different size: either solution time in seconds or optimality gap in percent after after a time limit of 1 minute (left). Number of rows, columns, bilinear terms (BLT) and non-zeros (NZ) of instance 6 after presolve in multiples of one thousand (right).

Instance	$\lvert I\rvert$	$\lvert P\rvert$	$\lvert O\rvert$	q	q+cuts
1	187	34	170	18.59%	0.42 s
2	229	43	215	3.91 s	0.71 s
3	306	82	410	38.69%	3.40 s
4	329	117	585	115.09%	0.14%
5	360	160	800	69.68%	0.40%
6	465	230	1 150	61.42%	0.36%

	q	q+cuts
#Rows	57	59
#Cols	19	17
#BLT	12	11
#NZ	157	164

other instances still have huge optimality gaps. In contrast, the q+cuts formulation solves the first three instances to optimality within seconds. The gaps of the other three instances are almost closed after 1 minute. To provide some more context, we have included the sizes of the two formulations for the largest instance 6 after Gurobi's presolve. Both formulations could be reduced to similar sizes, where, however, the q+cuts formulation has 10% less bilinear terms, which is an important factor influencing solution time. Furthermore, we observed that q+cuts found good primal solutions much faster; we assume that Gurobi's heuristics were able to benefit from the tighter relaxation it provides. Altogether, this shows that the q+cuts formulation is vastly superior to the q-formulation on our test set.

6 Conclusions

We have seen that the joint consideration of separable bilinear terms and multiple-choice constraints leads to a very rich combinatorial structure, whose exploitation is also beneficial from a computational point of view. Many symmetries of the bipartite boolean quadric polytope remain intact, which holds for the 0-lifted version of many facet-defining inequalities as well. Several subcases even allow for a characterization of the complete convex hull via reformulation-linearization inequalities. At the same time, there are very interesting new symmetries and facet classes which arise specifically due to the additional multiple-choice structure. Notably, the switching operation and the novel copy operation provide a lifting framework which is able to produce a vast amount of facets out of basic facet classes. Moreover, we gave separation routines for these facet classes, most of which run in polynomial time under the assumption of bounded support. All of these procedures allow for efficient implementation, and the corresponding cutting planes lead to an almost complete closure of the integrality gap on instances with up to 85 nodes in our experiments. Finally, we demonstrated that the bipartite boolean quadric polytope with multiple-choice constraints is an adequate model for pooling problems with fixed input proportions (i.e. recipes) at the pools. Putting our insights into practice allowed us to solve very hard real-world instances to (near-)optimality within a minute, to which a standard solver did not even come close in most cases.

Acknowledgements

We thank Francisco Javier Zaragoza Martínez for our fruitful discussions on the topic as well as Mark Zuckerberg and Thomas Kalinowski for clarifying details about Zuckerberg's technique for convex-hull proofs. Futhermore, we acknowledge financial support by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the Center for Analytics – Data – Applications (ADA-Center) within the framework of "BAYERN DIGITAL II".

References

[ABH+04] Charles Audet, Jack Brimberg, Pierre Hansen, Sébastien Le Digabel, and Nenad Mladenović. Pooling problem: Alternate formulations and solution methods. *Management science*, 50(6):761–776, 2004.

[AI07] David Avis and Tsuyoshi Ito. New classes of facets of the cut polytope and tightness of i_{mm22} Bell inequalities. *Discrete Applied Mathematics*, 155(13):1689–1699, 2007.

[AII08] David Avis, Hiroshi Imai, and Tsuyoshi Ito. Generating facets for the cut polytope of a graph by triangular elimination. *Mathematical programming*, 112(2):303–325, 2008.

[AIIS05] David Avis, Hiroshi Imai, Tsuyoshi Ito, and Yuuya Sasaki. Two-party Bell inequalities derived from combinatorics via triangular elimination. *Journal of Physics A: Mathematical and General*, 38(50):10971–10987, 2005.

[BDK+17] Natashia Boland, Santuna S. Dey, Thomas Kalinowski, Marco Molinaro, and Fabian Rigterink. Bounding the gap between the McCormick relaxation and the convex hull for bilinear functions. *Mathematical Programming, Series A*, 162:523–535, 2017.

[BGM20] Andreas Bärmann, Patrick Gemander, and Maximilian Merkert. The clique problem with multiple-choice constraints under a cycle-free dependency graph. *Discrete Applied Mathematics*, 283:59–77, 2020.

[BGMS18] Andreas Bärmann, Thorsten Gellermann, Maximilian Merkert, and Oskar Schneider. Staircase compatibility and its applications in scheduling and piecewise linearization. *Discrete Optimization*, 29:111–132, 2018.

[BH93] Endre Boros and Peter L Hammer. Cut-polytopes, boolean quadric polytopes and nonnegative quadratic pseudo-boolean functions. *Mathematics of Operations Research*, 18(1):245–253, 1993.

[BM86] Francisco Barahona and Ali Ridha Mahjoub. On the cut polytope. *Mathematical programming*, 36(2):157–173, 1986.

[BMS20] Andreas Bärmann, Alexander Martin, and Oskar Schneider. Efficient formulations and decomposition approaches for power peak reduction in railway traffic via timetabling. *Transportation Science*, 2020. To appear.

[BZ04] Daniel Bienstock and Mark Zuckerberg. Subset algebra lift operators for 0-1 integer programming. *SIAM Journal on Optimization*, 15(1):63–95, 2004.

[Cas15] Pedro M. Castro. Tightening piecewise McCormick relaxations for bilinear problems. *Computers & Chemical Engineering*, 72:300–311, 2015.

[CG04] Daniel Collins and Nicolas Gisin. A relevant two qubit Bell inequality inequivalent to the CHSH inequality. *Journal of Physics A: Mathematical and General*, 37(5):1775–1787, 2004.

[CM89] Joseph Cheriyan and SN Maheshwari. Analysis of preflow push algorithms for maximum network flow. *SIAM Journal on Computing*, 18(6):1057–1086, 1989.

[CSPB17] Ante Ćustić, Vladyslav Sokol, Abraham P. Punnen, and Binay Bhattacharya. The bilinear assignment problem: Complexity and polynomially solvable special cases. *Mathematical Programming, Series A*, 166:185–205, 2017.

[DLL11] Claudia D'Ambrosio, Jeff Linderoth, and James Luedtke. Valid inequalities for the pooling problem with binary variables. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 117–129. Springer, 2011.

[DLW97] MM Deza, Monique Laurent, and R Weismantel. Geometry of cuts and metrics. *Mathematical Methods of Operations Research-ZOR*, 46(3):282–283, 1997.

[DS90] Caterina De Simone. The cut polytope and the boolean quadric polytope. *Discrete Mathematics*, 79(1):71–75, 1990.

[FL18] Marcia Fampa and Jon Lee. Efficient treatment of bilinear forms in global optimization. `https://arxiv.org/abs/1803.07625`, 2018.

[FT05] Alain Faye and Quoc-an Trinh. A polyhedral approach for a constrained quadratic 0–1 problem. *Discrete Applied Mathematics*, 149(1-3):87–100, 2005.

[GACD13] Akshay Gupte, Shabbir Ahmed, Myun Seok Cheon, and Santanu Dey. Solving mixed integer bilinear problems using MILP formulations. *SIAM Journal on Optimization*, 23(2):721–744, 2013.

[GADC17] Akshay Gupte, Shabbir Ahmed, Santanu S Dey, and Myun Seok Cheon. Relaxations and discretizations for the pooling problem. *Journal of Global Optimization*, 67(3):631–669, 2017.

[GGL19] Laura Galli, Akshay Gupte, and Adam N Letchford. Separable bilinear programs, facial disjunctions and the reformulation-linearization technique. `https://pdfs.semanticscholar.org/8adc/5b08fdfbb4f2fdfa1c7584ff78810aeb3567.pdf`, 2019.

[GJ00] Ewgenij Gawrilow and Michael Joswig. `polymake`: a framework for analyzing convex polytopes. In *Polytopes—combinatorics and computation* (*Oberwolfach, 1997*), volume 29 of *DMV Sem.*, pages 43–73. Birkhäuser, Basel, 2000.

[GKRW20] Akshay Gupte, Thomas Kalinowski, Fabian Rigterink, and Hamish Waterer. Extended formulations for convex hulls of some bilinear functions. *Discrete Optimization*, 36:100569, 2020.

[GLL12] Oktay Günlük, Jon Lee, and Janny Leung. A polytope for a product of real linear functions in 0/1 variables. In *Mixed Integer Nonlinear Programming*, pages 513–529. Springer, 2012.

[Gup16] Akshay Gupte. A note on simplicial bilinear optimization. `http://agupte.people.clemson.edu/BilinSimpl.pdf`, 2016.

[Gur20] Gurobi Optimization, LLC. Gurobi optimizer reference manual. `http://www.gurobi.com`, 2020.

[HLL98] Jill Hardin, Jon Lee, and Janny Leung. On the boolean-quadric packing uncapacitated facility location polytope. *Annals of Operations Research*, 83:77–94, 1998.

[KCG13] Scott Kolodziej, Pedro M. Castro, and Ignacio E. Grossmann. Global optimization of bilinear programs with a multiparametric disaggregation technique. *Journal of Global Optimization*, 57:1039–1063, 2013.

[KPP04] Hans Kellerer, Urlrich Pferschy, and David Pisinger. *Knapsack Problems*, chapter The Multiple-Choice Knapsack Problem, pages 317–347. Springer, 2004.

[LG17] Ricardo M. Lima and Ignacio E. Grossmann. On the solution of nonconvex cardinality boolean quadratic programming problems: A computational study. *Computational Optimization and Applications*, 66(1):1–37, 2017.

[LL04] Jon Lee and Janny Leung. On the boolean quadric forest polytope. *INFOR: Information Systems and Operational Research*, 42(2):125–141, 2004.

[LM16] Frauke Liers and Maximilian Merkert. Structural investigation of piecewise linearized network flow problems. *SIAM Journal on Optimization*, 26(4):2863–2886, 2016.

[LS14] Adam N Letchford and Michael M Sørensen. A new separation algorithm for the boolean quadric and cut polytopes. *Discrete Optimization*, 14:61–71, 2014.

[McC76] Garth P McCormick. Computability of global solutions to factorable nonconvex programs: Part I – convex underestimating problems. *Mathematical programming*, 10(1):147–175, 1976.

[Meh97] Anuj Mehrotra. Cardinality constrained boolean quadratic polytope. *Discrete Applied Mathematics*, 79(1-3):137–154, 1997.

[MF09] Ruth Misener and Christodoulos A Floudas. Advances for the pooling problem: modeling, global optimization, and computational studies. *Appl. Comput. Math*, 8(1):3–22, 2009.

[Nau87] Robert M. Nauss. The 0-1 knapsack problem with multiple choice constraints. *European Journal of Operational Research*, 2(2):125–131, 1987.

[Pad89] Manfred Padberg. The boolean quadric polytope: some characteristics, facets and relatives. *Mathematical programming*, 45(1-3):139–172, 1989.

[Pen07] David W Pentico. Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176(2):774–793, 2007.

[Pit91] Itamar Pitowsky. Correlation polytopes: their geometry and complexity. *Mathematical Programming*, 50(1-3):395–414, 1991.

[PSK13] Abraham P. Punnen, Piyashat Sripratak, and Daniel Karapetyan. Domination analysis of algorithms for bipartite boolean quadratic programs. In *International Symposium on Fundamentals of Computation Theory (FCT)*, pages 271–282, 2013.

[PSK15] 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.

[PW16] Abraham P. Punnen and Yang Wang. The bipartite quadratic assignment problem and extensions. *Discrete Optimization*, 250:715–725, 2016.

[SA92] Hanif D Sherali and Amine Alameddine. A new reformulation-linearization technique for bilinear programming problems. *Journal of Global optimization*, 2(4):379–410, 1992.

[SLA95] Hanif D Sherali, Youngho Lee, and Warren P Adams. A simultaneous lifting strategy for identifying new classes of facets for the boolean quadric polytope. *Operations Research Letters*, 17(1):19–26, 1995.

[SPS19] Piyashat Sripratak, Abraham P. Punnen, and Tamon Stephen. The bipartite boolean quadric polytope. Technical report, Simon Fraser University, 2019.

[Sri14] Piyashat Sripratak. *The bipartite boolean quadratic programming problem*. PhD thesis, Science: Mathematics, 2014.

[WW01] Reinhard F. Werner and Michael M. Wolf. Bell inequalities and entanglement. `https://arxiv.org/abs/quant-ph/0107093`, 2001.

[Zuc16] Mark Zuckerberg. Geometric proofs for convex hull defining formulations. *Operations Research Letters*, 44(5):625–629, 2016.

Appendix

Here we give some of the details about the results omitted above. In Appendix A, we prove Theorem 3.11, which states that the arrow-2 inequalities define facets of $P(G, \mathcal{I})$. The separation algorithms for switchings and copyings of the arrow-1 and arrow-2 inequalities are given in Appendix B and Appendix C respectively. A separation template for general 0-lifted valid inequalities from BQP is presented in Appendix D. In Appendix E, we describe how to obtain valid inequalities for the lower-dimensional version of $P(G, \mathcal{I})$ where the multiple-choice constraints have to be fulfilled with equality. To complement the computational experiments presented in Section 5.1, we first investigate the number of cutting planes found per facet class in Appendix F. Finally, in Appendix G we examine which class of facet-defining inequalities is the second-strongest after the cycle+copying inequalities empirically.

A Proof of Theorem 3.11

Proof. We fix some distinct $I_1, I_2 \in \mathcal{I}$, a $i_1 \in I_1$, pairwise distinct $i_2, \ldots, i_m \in I_2$, pairwise distinct $j_1, \ldots, j_m \in Y$ and some $m \geq 3$. Let $a^T(x, y, z) \leq b$ be a facet-defining inequality which contains the face F induced by (13). In Algorithm 5, we indicate how to construct $|X| + |Y| + |E|$ affinely independent points on F to show that a and b are multiples of the coefficients of (13).

Copying lifts further variables into the inequality. For each $i \in I_1 \setminus \{i_1\}$, there are two possibilities: either all coefficients associated with i (for the x- and z-variables) are 0, then similar steps as 12–19 need to be done, or they are identical to those of i_1, then the construction can be adapted by replacing i with i_1 in Algorithm 5. For $i \in I_2 \setminus \{i_2, \ldots i_m\}$, there are m possibilities: again, either all coefficients are 0, or they coincide with those corresponding to some node from $k \in \{i_2, \ldots, i_m\}$, in which case we can replace i with k in Algorithm 5. Therefore, the arrow-2 inequalities and all their copyings define facets. □

B Separation of arrow-1+switching inequalities

In order to separate over all switchings of an arrow-1 inequality, Algorithm 4 for the separation of the original arrow-1 inequalities needs some minor changes. Firstly, the graph H from Figure 6 needs to be modified as shown in Figure 7.

The two possible cycles through j' and j'' represent the decision whether to switch on $\{j\}$ for $j \in J$ or not. The number of nodes and edges in the modified graph is $|\tilde{V}| = I^{\max} + 3|Y| + 1$ and $|A| = 4I^{\max}|Y| + I^{\max} + |Y|$ respectively, so the solution time of the resulting MCCP is moderately larger than before.

Algorithm 5 Construction of affinely independent points on an arrow-2 facet

```
From 0 ∈ F follows b = 0
From e_{j_1} ∈ F follows a_1 = 0
From e_{i_1} + e_{j_1} + e_{i_1 j_1} ∈ F follows a_{i_1 j_1} = −a_{i_1}
From e_{i_p} ∈ F follows a_{i_p} = 0 for all p = 2, ..., m
for p = 2, ..., m do
    From e_{i_p} + e_{j_p} + e_{i_p j_p} ∈ F follows a_{i_p j_p} = −a_{j_p}
    From e_{i_1} + e_{j_1} + e_{j_p} + e_{i_1 j_1} + e_{i_1 j_p} ∈ F follows a_{i_1 j_p} = −a_{j_p}
    From e_{i_1} + e_{i_p} + e_{j_1} + e_{j_p} + e_{i_1 j_1} + e_{i_1 j_p} + e_{i_p j_1} + e_{i_p j_p} ∈ F follows that a_{i_p j_1} = a_{j_p}
    From e_{i_1} + e_{i_p} + e_{j_p} + e_{i_1 j_p} + e_{i_p j_p} ∈ F follows that a_{i_1} = a_{j_p}
    From e_{i_p} + e_{i_q} + e_{j_p} + e_{i_p j_p} + e_{i_q j_p} ∈ F follows a_{i_q j_p} = 0 for all q ∈ {2, ..., m} \ p
end for
Ŷ ← Y \ {j_1, ..., j_m} and Î ← 𝓘 \ {I_1, I_2}
From e_j ∈ F follows a_j = 0 for all j ∈ Ŷ
From e_i ∈ F follows a_i = 0 for all i ∈ I with I ∈ Î
From e_{j_p} + e_{i_p} + e_i + e_{i j_p} + e_{i_p j_p} ∈ F follows a_{i j_p} = 0 for all i ∈ I with I ∈ Î and p = 2, ..., m
From e_j + e_i + e_{ij} ∈ F follows a_{i j_q} = 0 for all i ∈ I with I ∈ Î and j ∈ Ŷ
From e_{j_1} + e_i + e_{i j_1} ∈ F follows a_{i j_1} = 0 for all i ∈ I with I ∈ Î
From e_{i_1} + e_{j_1} + e_{i_1 j_1} + e_j + e_{i_h j} ∈ F follows a_{i_1 j} = 0 for all j ∈ Ŷ
From e_{i_h} + e_j + e_{i_h j} ∈ F follows a_{i_h j} = 0 for all h = 2, ..., m and j ∈ Ŷ
```

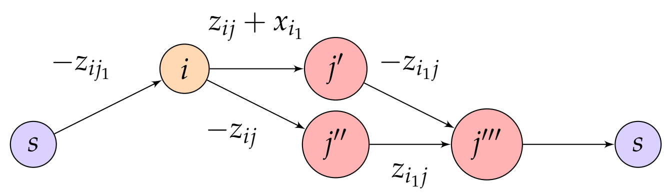


Figure 7: The necessary changes to graph H to separate over all switchings of the arrow-1 inequalities. Each node $j \in J$ is split into three nodes j', j'' and j'''. Then the arcs (i, j') and (i, j'') for $i \in I_2$ as well as (j', j'''), (j'', j''') and (j''', s) are introduced with costs as shown above.

Secondly, the enumeration part in Algorithm 4 needs to be extended, too. We loop twice over all $j_1 \in Y$, once with arc costs directly as in Figure 7 and once more an arc cost on (s, i) incorporating the switching on $\{j_1\}$. This basically doubles the running time of the overall algorithm.

Furthermore the arrow-2+switching inequalities can be separated via a slight modification of the algorithm by changing the cost function for the MCCP to reflect the support of the arrow-2+switching inequalities.

C Separation of arrow-1+copying inequalities

Now we consider separating over all copyings of an arrow-1 inequality, which are given by

$$\sum_{i \in S_1} \left((m-1)x_i - \sum_{p=1}^{m} z_{ij_p} \right) + y_{j_1} + \sum_{p=2}^{m} \sum_{i \in S_p} \left(-z_{ij_1} + z_{ij_m} \right) \geq 0, \tag{26}$$

for distinct $I_1, I_2 \in \mathcal{I}$, a non-empty subset $S_1 \subseteq I_1$, pairwise disjoint and non-empty subsets $S_2, \ldots, S_m \subseteq I_2$ and pairwise distinct $j_1, \ldots, j_m \in Y$ with $m \geq 3$. To this end, we replace Algorithm 4 by Algorithm 6. The inner optimization subproblem is now an integer MCCP on the same graph, as described in the following.

For distinct $I_1, I_2 \in \mathcal{I}$ as well as $j_1 \in Y$, let $J := Y \setminus \{j_1\}$, and let $H = (\tilde{V}, A)$ be a directed graph with node set $\tilde{V} := \{s\} \cup I_2 \cup J$. The arc set A contains the arcs (s,i) for $i \in I_2$, all $(i,j) \in I_2 \times J$ and (j,s) for $j \in J$.

Let $u_{ij} \in \mathbb{N}$ for $(i,j) \in A$ be the flow variables. Variables $b_j \in \{0,1\}$ for $j \in J$ stand for the decision to use arc $(j,s) \in A$. Further, variables $r_i \in \{0,1\}$ decide if node $i \in I_1$ is chosen to be in set S_1. Finally, variables $p_{ij} \in \{0,1\}$ for $i \in I_1$ and $j \in J$ model the product between b_j and r_i. The constraints are then given by

$$u_{ij} \leq 1, \quad (i,j) \in \tilde{E} \setminus \{(j,s) : j \in J\} \tag{27a}$$
$$\sum_{j \in J} b_j \geq 1, \tag{27b}$$
$$\sum_{i \in I_1} r_i \geq 1, \tag{27c}$$
$$u_{si} = \sum_{j \in J} y_{ij}, \quad \forall i \in I_2, \tag{27d}$$
$$u_{js} = \sum_{i \in I_2} y_{ij}, \quad \forall j \in J, \tag{27e}$$
$$u_{js} \geq b_j, \quad \forall j \in J, \tag{27f}$$
$$u_{js} \leq |I_2| b_j, \quad \forall j \in J, \tag{27g}$$
$$p_{ij} \leq r_i, \quad \forall i \in I_1, j \in J, \tag{27h}$$
$$p_{ij} \leq b_j, \quad \forall i \in I_1, j \in J, \tag{27i}$$
$$r_i + b_j - p_{ij} \leq 1, \quad \forall i \in I_1, j \in J. \tag{27j}$$

Constraints (27a) enforce the arc capacities, (27b) ensure that at least one $j \in J$ is chosen, (27c) require to choose at least one $i \in I_1$, (27d) and (27e) are the flow conservation constraints, (27f) demand that there be flow on (j,s) if $j \in J$ is chosen, (27g) ensure that there are at most $|I_2|$ units of flow if $j \in J$ is chosen and that there is no flow otherwise, (27h) to (27j) model $p_{ij} = r_i b_j$.

The objective can be stated as

$$\min \sum_{i \in I_2} z_{ij_1} u_{si} + \sum_{i \in I_2, j \in J} z_{ij} u_{ij} + \sum_{i \in I_1, j \in J} (-z_{ij} + x_i) p_{ij} + \sum_{i \in I_1} (-z_{ij_1} r_i). \tag{28}$$

An arrow-1+copying inequality is uniquely defined if distinct $I_1, I_2 \in \mathcal{I}$, a non-empty subset $S_1 \subseteq I_1$, a $j_1 \in Y$, pairwise distinct $j_2, \dots j_m \in J$ and non-empty, pairwise disjoint $S_2, \dots, S_m \subseteq I_2$ are chosen. The possible choices for I_1, I_2 and j_1 are again enumerated. In step 3 we solve the integer MCCP. It determines adaptively the optimal value for m. The optimal S_1 is then extracted from the r variables in step 7, and the optimal $j_2, \dots, j_m$ are derived from the values of the b-variables in step 8. Finally, the optimal $S_2, \dots S_m$ are calculated in steps 9–11, based on the optimal flow in the graph.

The arrow-2+copying inequalities can be separated via a slight modification of the integer program by changing the cost function for the minimum-cost-circulation problem to reflect the support of the arrow-2+copying inequalities.

D Separation of general lifted inequalities from BQP

For a subset uniform graph G, any valid inequality for the BQP on the corresponding dependency graph $\mathcal{G}$ can be 0-lifted to $P(G, \mathcal{I})$. If $|\text{supp}_Y| = m$ for some $m \geq 1$, such an inequality is of the form

$$\sum_{k=1}^{h} a_{i_k} x_{i_k} + \sum_{p=1}^{m} a_{j_p} y_{j_p} + \sum_{k=1}^{h} + \sum_{p=1}^{m} (a_{i_k j_p} z_{i_k j_p}) \leq \delta,$$

Algorithm 6 Separation algorithm for arrow-1+copying inequalities

Input: $(x, y, z) \in R^{X \cup Y \cup E}$
Output: Most violated arrow-1+copy inequality for each $I_1, I_2 \in \mathcal{I}, I_1 \neq I_2, j_1 \in Y$.
1: **function** ARROW-1+COPY-SEPARATOR(x, y, z)
2:     **for** $I_1, I_2 \in \mathcal{I}, I_1 \neq I_2, j_1 \in Y$ **do**
3:         Solve integer problem defined by (27) and (28)
4:         $o \leftarrow$ objective value of the integer problem in step 3
5:         $\bar{u}, \bar{b}, \bar{r}, \bar{p} \leftarrow$ optimal solution of the integer MCCP in step 3
6:         **if** $o + y_{j_1} < 0$ **then**
7:             $S_1 \leftarrow \{i \in I_1 \mid \bar{r}_i = 1\}$
8:             $\{j_2, \ldots, j_m\} \leftarrow \{j \in J \mid \bar{b}_j = 1\}$ ▷ naming the elements
9:             **for** $n \in \{2, \ldots, m\}$ **do**
10:                 $S_n \leftarrow \{i \in I_2 \mid (i, j_n) \in W, \bar{u}_{ij_n} = 1\}$
11:             **end for**
12:             **return** violated inequality based on $(I_1, I_2, S_1, S_2, \ldots S_m, j_1, j_2, \ldots j_m)$
13:         **end if**
14:     **end for**
15: **end function**

with $i_1 \in I_1, \ldots, i_h \in I_h$ for pairwise distinct $I_1, \ldots, I_h \in \mathcal{I}$ and pairwise distinct $j_1, \ldots, j_m \in Y$. We now assume that for a specific class of constraints to separate over, the coefficient vector a is already uniquely determined by the above choice of $i_1, \ldots, i_h$ and $j_1, \ldots, j_m$. The cycle or the I_{mm22} Bell inequalities would be examples here. For such BQP inequalities with bounded Y-support, we can give a separation template which generalizes Algorithm 3 for the cycle inequalities. It is shown in Algorithm 7.

Algorithm 7 Separation algorithm for a given class of 0-lifted facets from BQP

Input: $(x, y, z) \in \mathbb{R}^{X \cup Y \cup E}, m \in [|Y|], h \in [|\mathcal{I}|]$
Output: Most violated inequality for pairwise distinct $j_1, \ldots, j_m \in Y$
1: **function** LIFT-SEPARATOR$((x, y, z))$
2:     **for** pairwise distinct $j_1, \ldots, j_m \in Y$ **do**
3:         **for** $I \in \mathcal{I}, k \in \{1, \ldots, h\}$ **do**
4:             $c_k^I \leftarrow \max\{a_{i_k} x_r + \sum_{p=1}^{m} a_{i_k j_p} z_{r j_p} \mid r \in I\}$
5:         **end for**
6:         Solve the h-cardinality assignment problem with objective c
7:         $o \leftarrow$ objective value of the assignment problem in step 6
8:         $v \leftarrow \sum_{s=1}^{m} a_{j_s} y_{j_s}$
9:         **if** $o + v > \delta$ **then**
10:             $\{(I_1, 1), \ldots (I_h, h)\} \leftarrow$ optimal solution in step 6
11:             **for** $k \in \{1, \ldots, h\}$ **do**
12:                 $i_k \leftarrow \operatorname{argmax}\{a_{i_k} x_r + \sum_{p=1}^{m} a_{i_k j_p} z_{r j_p} \mid r \in I_k\}$
13:             **end for**
14:             **return** violated inequality based on $(j_1, \ldots, j_m, I_1, \ldots, I_h, i_1, \ldots, i_h)$
15:         **end if**
16:     **end for**
17: **end function**

A 0-lifted BQP inequality of the considered class is uniquely defined if we chose pairwise distinct $j_1, \ldots, j_m \in Y$, pairwise distinct $I_1, \ldots, I_h \in \mathcal{I}$ and one node from each $I_1, \ldots, I_h$. In general, there are exponentially many possible inequalities of this type. For each fixed choice of nodes $j_1, \dots, j_m \in Y$ (step 2), we define the values c_k^I for $I \in \mathcal{I}$ and $k = 1, \dots, h$ as the largest contribution that any element $i \in I$ would add to the left-hand side of the inequality to be separated (steps 3–5) if i was assigned to the coefficients associated with k. Then an h-cardinality maximum assignment problem (see [Pen07]) with objective c needs to be solved (step 6). It determines which $I \in \mathcal{I}$ is assigned to which $k \in \{1, \dots, h\}$. From the optimal assignment, the best possible choice of the subsets $I_1, \dots I_h$ can be extracted (step 10), from which we can derive the optimal $i_1, \dots, i_h$ (steps 11–13).

If the considered inequality class from BQP contains inequalities with y-supports of different sizes, the algorithm needs to be run for every possible m. In order to separate also over all copies of these inequalities, the maximum in step 4 needs to be taken only over the sum of all elements that would make a positive contribution to the left-hand side, similar to steps 7–9 in Algorithm 3.

In fact, Algorithm 3 (extended to include the copyings) is a special case of Algorithm 3, where we have $m = 2$, and where solving the assignment subproblem can be done via a greedy algorithm. The running time of Algorithm 3 is $\mathcal{O}((|Y|)!/m!D(|\mathcal{I}|, h))$, where $D(|\mathcal{I}|, h)$ is the running time needed to solve the h-cardinality assignment subproblem. It can be solved as a minimum-cost flow problem on a graph, where the node set has size $|V| = |\mathcal{I}| + h + 2$ and the edge set has size $|E| = |V||\mathcal{I}| + |\mathcal{I}| + |V|$. The push-relabel algorithm (see [CM89]), for example solving this minimum-cost flow problem within $\mathcal{O}(|V|^2\sqrt{|E|})$. Note that Algorithm 7 is super-exponential. However, for fixed m it runs in polynomial time.

E Transformation between lower- and full-dimensional space

Recall that we have defined the multiple-choice set

$$X^{\mathcal{I}} := \left\{ x \in \{0,1\}^X \;\middle|\; \sum_{i \in I} x_i \le 1 \;\forall I \in \mathcal{I} \right\}$$

and the boolean quadric polytope with multiple-choice constraints

$$P(G, \mathcal{I}) := \operatorname{conv}\left\{ (x,y,z) \in \{0,1\}^{X \cup Y \cup E} \;\middle|\; x_i y_j = z_{ij} \;\forall \{i,j\} \in E, x \in X^{\mathcal{I}} \right\}$$

as full-dimensional polyhedra.

In the following, we give a one-to-one transformation between valid inequalities for $P(G, \mathcal{I})$ and valid inequalities for a lower-dimensional variant of $P(G, \mathcal{I})$, where all multiple-constraints have to be fulfilled with equality. Indeed, this lower-dimensional variant corresponds to a face of $P(G, \mathcal{I})$.

We assume that G is a complete bipartite graph. Then we define a new complete bipartite graph $\tilde{G} = (\tilde{X} \cup Y, \tilde{E})$ with one extra node in each subset in the partition $\mathcal{I}$ of X. Let $\tilde{\mathcal{I}}$ be the corresponding partition of $\tilde{X}$. Further, we define the multiple-choice set

$$\tilde{X}^{\mathcal{I}} := \left\{ x \in [0,1]^{\tilde{X}} \;\middle|\; \sum_{i \in I} x_i = 1 \;\forall I \in \tilde{\mathcal{I}} \right\} \tag{29}$$

and the polytope

$$\tilde{P}(\tilde{G}, \tilde{\mathcal{I}}) := \operatorname{conv}\left\{ (x,y,z) \in \{0,1\}^{\tilde{X} \cup Y \cup \tilde{E}} \;\middle|\; x_i y_j = z_{ij} \;\forall \{i,j\} \in \tilde{E},\, x \in \tilde{X}^{\mathcal{I}} \right\}. \tag{30}$$

We will now state an affine transformation which rotates $\tilde{X}^{\mathcal{I}}$ in such a way that one variable per subset of the partition $\tilde{\mathcal{I}}$ becomes constantly zero. For each $I \in \tilde{\mathcal{I}}$ we choose an arbitrary nodes $i_0^I \in I$ and define the matrices $B_I \in \mathbb{R}^{I \times I}$, $B \in \mathbb{R}^{\tilde{X} \times \tilde{X}}$ and vectors $b_I \in \mathbb{R}^I$, $b \in \mathbb{R}^{\tilde{X}}$ as follows:

$$B_I(i_1, i_2) = \begin{cases} 1 & \text{for } i_1 = i_2 \\ 1 & \text{for } i_1 = i_0^I \\ 0 & \text{else} \end{cases}, \quad B = \begin{bmatrix} B_{I_1} & & 0 \\ & \ddots & \\ 0 & & B_{I_m} \end{bmatrix},$$

$$b_I(i) = \begin{cases} -1 & \text{for } i = i_0^I \\ 0 & \text{else} \end{cases}, \quad b = \begin{bmatrix} b_{I_1} \\ \vdots \\ b_{I_m} \end{bmatrix}.$$

By applying the invertible affine transformation $f\colon \mathbb{R}^{\tilde{X}} \to \mathbb{R}^{\tilde{X}}, x \mapsto Bx - b$ to $\tilde{X}^{\mathcal{I}}$, we arrive at

$$\bar{X}^{\mathcal{I}} := f(\tilde{X}^I) = \left\{ x \in \{0,1\}^{\tilde{X}} \;\middle|\; \sum_{i \in I \setminus \{i_0^I\}} x_i \leq 1,\, x_{i_0^I} = 0 \;\forall I \in \tilde{\mathcal{I}} \right\}.$$

This allows us to define the polytope

$$\bar{P}(\tilde{G}, \tilde{\mathcal{I}}) := \operatorname{conv} \left\{ (x, y, z) \in \{0,1\}^{\tilde{X} \cup Y \cup \tilde{E}} \;\middle|\; x_i y_j = z_{ij} \;\forall \{i,j\} \in \tilde{E},\, x \in \bar{X}^{\mathcal{I}} \right\},$$

which is the canonical embedding of $P(G, \mathcal{I})$ into $\mathbb{R}^{\tilde{X}^I} = \mathbb{R}^{\bar{X}^I}$ (in fact an extended formulation). Using Lemma 2 in [GGL19], we can infer the two relations

$$\bar{P}(\tilde{G}, \tilde{\mathcal{I}}) = \left\{ (Bx - b, y, Bz - by^T) \in [0,1]^{\tilde{X} \cup Y \cup \tilde{E}} \;\middle|\; (x, y, z) \in \tilde{P}(\tilde{G}, \tilde{\mathcal{I}}) \right\}$$

and

$$\tilde{P}(\tilde{G}, \tilde{\mathcal{I}}) = \left\{ (B^{-1}(x + b), y', B^{-1}(z + by^T)) \in [0,1]^{\tilde{X} \cup Y \cup \tilde{E}} \;\middle|\; (x, y, z) \in \bar{P}(\tilde{G}, \tilde{\mathcal{I}}) \right\}.$$

Based on this observation, we can formulate the following lemma, which transform inequalities back and forth between the two polytopes.

Lemma E.1. *Let $a^T(x, y, z) \leq b$ be a valid inequality for $\bar{P}(\tilde{G}, \tilde{\mathcal{I}})$, then the inequality $a^T(B^{-1}(x + b), y, B^{-1}(z + by^T)) \leq \bar{b}$ is valid for $\tilde{P}(\tilde{G}, \tilde{\mathcal{I}})$ and vice versa.*

One can easily see that $\bar{P}(\tilde{G}, \tilde{\mathcal{I}})$ and $P(G, \mathcal{I})$ are the "same" polytope in different dimension. As all variables which do not appear in $P(G, \mathcal{I})$ are fixed to 0 in $\bar{P}(\tilde{G}, \tilde{\mathcal{I}})$, we can conclude from Theorem 2.4 that the extension of a facet of $P(G, \mathcal{I})$ is also a facet of $\bar{P}(\tilde{G}, \tilde{\mathcal{I}})$. In addition, the following *basic equations* hold for $\bar{P}(\tilde{G}, \tilde{\mathcal{I}})$ (cf. the basic inequalities in Section 3.1):

$$x_{i_0^I} = 0, \quad \forall I \in \tilde{\mathcal{I}}, \tag{31}$$

$$z_{i_0^I j} = 0, \quad \forall I \in \tilde{\mathcal{I}}, \forall j \in Y. \tag{32}$$

Via the inverse transformation to f applied to (32), we obtain the following further valid equations:

$$\sum_{i \in I} z_{ij} = y_j, \quad \forall j \in Y, I \in \hat{\mathcal{I}}. \tag{33}$$

As these equations can also be derived from (32) via the RLT procedure, we call them the *RLT equations*.

Altogether, in order to obtain tighter linear relaxations for $\tilde{P}(\tilde{G}, \tilde{\mathcal{I}})$, we can add the RLT equations to the initial formulation and can derive further valid inequalities as follows. We remove one arbitrary node from each subset $\tilde{\mathcal{I}}$, which yields a graph $\hat{G} = (\hat{X}, \hat{E})$ and the corresponding partition $\hat{\mathcal{I}}$ on $\hat{X}$. Any valid inequality for $P(\hat{G}, \hat{\mathcal{I}})$ can now directly be added to the relaxation as well (as a 0-lifted inequality).

In the pooling problem with recipes, which we have presented in Section 5.2, we need to consider multiple instances of $\tilde{P}(\tilde{G},\tilde{\mathcal{I}})$ occurring as a substructure of the overall problem. In order to make use of the above technique to improve the relaxation, we first rescale all multiple-choice constraints (22) such that they have a right-hand side of 1 instead of σ_h, and then rescale all variables to have an upper bound of 1. This way, the polytopes (23) are indeed of the form $\tilde{P}(\tilde{G},\tilde{\mathcal{I}})$, which allows us to separate the cutting planes derived in Section 3 using the techniques presented in Section 4.

F Number of cutting planes produced in Section 5.1

In Section 5.1, we conducted experiments on random instances to find out which classes of facet-defining inequalities we derived are able to close the integrality gap how far. The results were given in Table 1. Here we report how many cutting planes of each type were needed for each given instance type to produce these results. In Table 3, we see the number of cutting planes found for the corresponding cells of Table 1, again averaged over all 10 instances of each type.

Table 3: Number of cutting planes separated for instances of different size and for various facet classes until no more violated inequalities were found

	5-5-10	10-10-10	15-15-10	5-5-20	5-5-40	5-5-60	10-*-25
RLT	100	200	300	200	400	600	500
C	460	1 014	1 180	1 937	4 963	13 473	3 796
CC	361	1 026	1 709	2 496	12 242	26 285	9 749
A1	979	8 390	30 839	3 048	5 448	7 662	16 460
A1S	1 538	28 946	141 927	8 682	23 971	36 671	57 074
A1C	418	3 370	8 927	1 716	4 036	6 546	8 448
A2	355	1 076	1 947	1 134	3 979	8 428	5 451
A2S	1 454	5 886	12 591	4 704	14 714	31 140	22 551
A2C	646	4 670	12 008	2 490	6 682	12 071	10 674
All	1 954	20 464	69 273	8 635	31 591	62 479	57 206

The RLT inequalities, which had all been added from the start, are a comparably small class of facet-defining inequalities. Nevertheless, we saw from our experiments on both random and real-world instances, that they are very helpful in moving the dual bound.

The number of CC inequalities we found is about 2.7 times the number of C inequalities, which is an indication that the copying operation is able to increase the class of ordinary cycle facets tremendously. It is also interesting to note that much fewer A1C inequalities than original A1 inequalities are found, but nevertheless the former provide a significantly better dual bound. This and the results for the C/CC inequalities point to a high potential of the coyping operation as a lifting method for a given basic class of valid inequalities. The All row finally shows that a relatively high number of violated inequalities is separated when considering all classes jointly and iterating until no further violated inequalities are found – up to 70 000 for the largest instances.

G Second strongest facet class after CC

From the results in Tables 1 and 3, we concluded that the CC inequalities are by far the most efficient ones to separate: they provide a very strong improvement in the dual bound per putting plane added to the relaxation, and at the same time, their separation is possible at very low computational cost. This motivated us to examine which of the facet classes we found is the second-strongest after the CC facets. In Table 4, we show the results obtained for the largest instances in our test set when first adding all violated CC inequalities and then iteratively separating the CC inequalities and the respective second class of inequalities jointly until no further violated inequalities are found.

Table 4: Average optimality gaps in % when separating the CC inequalities together with a second inequality class for the three classes of instances from Table 1 with a non-zero gap after separating the CC inequalities alone (left). The corresponding number of cutting planes found (right).

	5-5-40	5-5-60	10-*-25	5-5-40	5-5-60	10-*-25
CC	1.99	4.43	4.05	12 242	26 285	9 749
RLT	1.41	3.49	3.19	400(+335)	600(+827)	500(+645)
A1S	0.88	2.19	2.54	2 534(+451)	8 920(+1 307)	5 367(+894)
A1C	1.28	2.74	3.48	957(+344)	3 426(+1 069)	1 150(+479)
A2S	1.97	4.38	3.61	48(+34)	125(+82)	703(+383)
A2C	0.70	1.83	2.17	1 799(+660)	5 657(+1 928)	4 061(+1 314)
All	0.45	1.48	1.64			

The left-hand side of the table shows the remaining optimality gap while the right-hand side shows the number of cutting planes found. In the CC row and the All row, we repeat the results from Table 1 for comparison. The five rows in the middle indicate the results for choosing RLT, A1S, A1C, A2S and A2C, respectively, as the inequality class to separate jointly with CC. Furthermore, we see how many additional cutting planes of the respective second class were needed, with the number of additional CC inequalities in parentheses. We infer from the results that all classes make a certain further contribution to closing the gap compared to separating the CC inequalities alone. The A2C inequalities seem to be the most promising ones; however, they require the repeated solution of an integer program to separate. An almost equally good result is obtained for the A1S inequalities, where the subproblem is only a continuous flow problem. Finally, we see that the remaining gaps in the All row are significantly lower, which means that using more than two facet classes still yields further progress.