Face Dimensions of General-Purpose Cutting Planes for Mixed-Integer Linear Programs

Matthias Walter[*]

November 11, 2020

Abstract

Cutting planes are a key ingredient to successfully solve mixed-integer linear programs. For specific problems, their strength is often theoretically assessed by showing that they are facet-defining for the corresponding mixed-integer hull. In this paper we experimentally investigate the dimensions of faces induced by general-purpose cutting planes generated by a state-of-the-art solver. Therefore, we relate the dimension of each cutting plane to its impact in a branch-and-bound algorithm.

1 Introduction

We consider the mixed-integer program

$$\begin{aligned} &\max\ c^{\intercal} x && \text{(1a)}\\ &\text{s.t.}\quad Ax \leq b && \text{(1b)}\\ &\qquad\quad x_i \in \mathbb{Z} \quad \forall i \in I && \text{(1c)} \end{aligned}$$

for a matrix $A \in \mathbb{R}^{m \times n}$, vectors $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$ and a subset $I \subseteq \{1, 2, \dots, n\}$ of integer variables. Let $P := \operatorname{conv}\{x \in \mathbb{R}^n : x \text{ satisfies (1b) and (1c)}\}$ denote the corresponding *mixed-integer hull.* A *cutting plane* for (1) is an inequality $a^{\intercal} x \leq \beta$ that is valid for P (and possibly invalid for some point computed during branch-and-cut). Such a valid inequality *induces the face* $F := \{x \in P : a^{\intercal} x = \beta\}$ of P. For more background on polyhedra we refer to [10].

To tackle specific problems, one often tries to find *facet-defining* inequalities, which are inequalities whose induced face F satisfies $\dim(F) = \dim(P) - 1$. This is justified by the fact that any system $Cx \leq d$ with $P = \{x : Cx \leq d\}$ has to contain an inequality that induces F. Since the dimension of a face can vary between -1 (no $x \in P$ satisfies $a^{\intercal} x = \beta$) and $\dim(P)$ (all $x \in P$ satisfy $a^{\intercal} x = \beta$), the following hypothesis is a reasonable generalization of facetness as a strength indicator.

Hypothesis 1. *The practical strength of an inequality correlates with the dimension of its induced face.*

There is no unified notion of the practical strength of an inequality, but we will later define one that is related to its impact in a branch-and-bound algorithm. The main goal of this paper is to computationally test this hypothesis for general-purpose cutting planes used in MIP solvers.

Outline. In Section 2 we present the algorithm we used to compute dimensions. Section 3 is dedicated to the score we used to assess a cutting plane's impact, and in Section 4 we present our findings, in particular regarding Hypothesis 1.

[*]Department of Applied Mathematics, University of Twente, The Netherlands, `m.walter@utwente.nl`

2 Computing the dimension of a face

This section is concerned about how to effectively compute the dimension of P or of one of its faces F induced by some valid inequality $a^\intercal x \leq \beta$. In our experiments we will consider instances with several hundreds variables, and hence the enumeration of all vertices of P or F is typically impossible. Instead, we use an oracle-based approach. An *optimization oracle* for P is a black-box subroutine that can solve any linear program over the polyhedron P, i.e., for any given $w \in \mathbb{R}^n$ it can solve

$$\max\ w^\intercal x \text{ s.t. } x \in P. \tag{2}$$

In case (2) is feasible and bounded, the oracle shall return an optimal solution, and in case it is feasible and unbounded, it shall return an unbounded direction. Note that for such an oracle we neither require all vertices of P nor all valid (irredundant) inequalities. Indeed, we can use any MIP solver and apply it to problem (1) with $c \coloneqq w$.

We now discuss how one can compute $\dim(P)$ by only accessing an optimization oracle. To keep the presentation simple we assume that P is bounded, although our implementation can handle unbounded polyhedra as well. The algorithm maintains a set $X \subseteq P$ of affinely independent points and a system $Dx = e$ of valid equations, where D has full row-rank r. Hence, $\dim(X) \leq \dim(P) \leq n - r$ holds throughout and the algorithm works by either increasing $\dim(X)$ or r in every iteration. The details are provided in Algorithm 1.

Algorithm 1: Affine hull of a polytope via optimization oracle

Input: Optimization oracle for a polytope $\emptyset \neq P \subseteq \mathbb{R}^n$.

Output: Affine basis X of $\operatorname{aff}(P)$, non-redundant system $Dx = e$ with $\operatorname{aff}(P) = \{x \in \mathbb{R}^n : Dx = e\}$.

```
1  Initialize $X \coloneqq \emptyset$ and $Dx = e$ empty.
2  while $|X| + 1 < n - |\text{rows}(D)|$ do
3      Compute a direction vector $d \coloneqq \operatorname{aff}(X)^\perp \setminus \operatorname{span}(\text{rows}(D))$.
4      Query the oracle to maximize $d^\intercal x$ over $x \in P$ and let $x^+ \coloneqq \arg\max\{d^\intercal x : x \in P\}$.
5      Query the oracle to maximize $-d^\intercal x$ over $x \in P$ and let $x^- \coloneqq \arg\min\{d^\intercal x : x \in P\}$.
6      if $d^\intercal x^+ = d^\intercal x^-$ then
7          Add equation $d^\intercal x = d^\intercal x^+$ to system $Dx = e$.
8          if $X = \emptyset$ then set $X \coloneqq \{x^+\}$.
9      end
10     else if $X = \emptyset$ then set $X \coloneqq \{x^+, x^-\}$.
11     else
12         Let $\gamma \coloneqq d^\intercal x$ for some $x \in X$.
13         if $d^\intercal x^+ \neq \gamma$ then set $X \coloneqq X \cup \{x^+\}$.
14         else set $X \coloneqq X \cup \{x^-\}$.
15     end
16 end
17 return $(X, Dx = e)$.
```

Proposition 2. *For an optimization oracle for a non-empty polytope $P \subseteq \mathbb{R}^n$, Algorithm 1 requires $2n$ oracle queries to compute a set $X \subseteq P$ with $|X| = \dim(P)+1$ and a system $Dx = e$ of $n-\dim(P)$ equations satisfying*

$$\operatorname{aff}(P) = \operatorname{aff}(X) = \{x \in \mathbb{R}^n : Dx = e\}.$$

Proof. Since every point $x \in X$ was computed by the optimization oracle, we have $X \subseteq P$. Moreover, $Dx = e$ is valid for P since for each equation $d^\intercal x = \gamma$, $\min\{d^\intercal x : x \in P\} = \gamma = \max\{d^\intercal x : x \in P\}$ holds. We now prove by induction on the number of iterations that in every iteration of the algorithm, the points $x \in X$ are affinely independent and that the rows of D are linearly independent.

Initially, this is invariant holds because $X = \emptyset$ and D has no rows. Whenever an equation $d^\intercal x = d^\intercal x^+$ is added to $Dx = e$ in step 7, the vector d is linearly independent to the rows of D (see step 3). If in step 8, X is initialized with a single point, it is clearly affinely independent.

Similarly, the two points in step 10 also form an affinely independent set since $d^\intercal x^+ > d^\intercal x^-$ implies $x^+ \neq x^-$. Suppose X is augmented by $\bar{x} := x^+$ in step 13 or by $\bar{x} := x^-$ in step 14. Note that by the choice of d in step 3, $d^\intercal x = \gamma$ holds for all $x \in X$. Due to $d^\intercal \bar{x} \neq \gamma$, $X \cup \{\bar{x}\}$ remains affinely independent.

After the first iteration, we have $|X| + |\operatorname{rows}(C)| = 2$, and in every further iteration, this quantity increases by 1. Hence, the algorithm requires n iterations, each of which performs 2 oracle queries. The result follows. □ □

For a proof that the number $2n$ of oracle queries is optimal we refer to [11].

Implementation details. We now describe some details of the implementation within the software framework `IPO` [12]. In the description of Algorithm 1 we did not specify step 3 precisely. While one may enforce the requirement $d \notin \operatorname{span}(\operatorname{rows}(D))$ via $d \in \operatorname{rows}(D)^\perp$, it turned out that this is numerically less stable than first computing a basis of $\operatorname{aff}(X)^\perp$ and selecting a basis element d that is not in the span of D's rows. Moreover, we can take d's sparsity and other numerical properties into account. Sparsity can speed-up the overall computation since very sparse objective vectors are sometimes easier to optimize for a MIP solver. In theory, for all $x \in X$, the products $d^\intercal x$ have the same value. However, due to floating-point arithmetic the computed values may differ. It turned out that preferring directions d for which the range of these products is small helps to avoid numerical difficulties.

In our application we compute the dimension of P and of several of its faces. We can exploit this by caching all points $x \in P$ returned by an optimization oracle in a set $\bar{X}$. Then, for each face F induced by an inequality $a^\intercal x \leq \beta$ we can then compute the set $\bar{X}_F := \{x \in \bar{X} : a^\intercal x = \beta\}$. Before querying the oracle we can then check the set $\bar{X}_F$ for a point with sufficiently large objective value, which saves two calls to the MIP solvers.

Let $\bar{D}x = \bar{e}$ be the equation system returned by Algorithm 1 for P. Now, for a face F induced by inequality $a^\intercal x \leq \beta$ we can initialize $Dx = e$ in the algorithm by $\bar{D}x = \bar{e}$. Moreover, if $a^\intercal x = \beta$ is not implied by $Dx = e$ we add this equation to $Dx = e$ as well.

Since the algorithm is implemented in floating-point arithmetic, errors can occur which may lead to wrong dimension results. We checked the results using an exact arithmetic implementation of Algorithm 1 for easier instances (with less than 200 variables). For these, the computed dimension varied by at most 2 from the true dimension.

In a first implementation, our code frequently reported dimension -1, and it turned out that often the right-hand side was only slightly larger than needed to make the inequality supporting. Thus, for each inequality $a^\intercal x \leq \beta$, normalized to $||a||_2 = 1$, we computed $\beta^{\text{true}} := \max\{a^\intercal x : x \in P\}$ by a single oracle query. Whenever we observed $\beta < \beta^{\text{true}} - 10^{-4}$, we considered the cut $a^\intercal x \leq \beta$ as invalid (indicated by the symbol ϟ). If $\beta > \beta^{\text{true}} + 10^{-4}$, we declare the cut to be non-supporting. Otherwise, we replace β by β^{true} before running Algorithm 1.

3 Measuring the strength of a single inequality

In this section we introduce our *cut impact measure* for indicating, for a given cutting plane $a^\intercal x \leq \beta$, how useful its addition in the context of branch-and-cut is. The main goal of solving an LP at a branch-and-bound node is to determine a dual bound of the current subproblem. If this bound is at most the value of the best feasible solution known so far, then the subproblem can be discarded. Thus, we consider the value of such a bound (after adding a certain cut) in relation to the problem's optimum $z^\star := \max\{c^\intercal x : Ax \leq b,\ x_i \in \mathbb{Z}\ \forall i \in I\}$ and the value $z^{\text{LP}} := \max\{c^\intercal x : Ax \leq b\}$ of the LP relaxation.

Our first approach was to just evaluate the dual bound obtained from the LP relaxation $Ax \leq b$ augmented by $a^\intercal x \leq \beta$. However, adding a single inequality often does not cut off the optimal face of the LP relaxation, which means that the bound does not change. In a second attempt we tried to evaluate the dual bound of the LP relaxation augmented by a random selection of k cutting planes. However, the variance of the resulting cut impact measure was very large even after averaging over 10.000 such selections.

As a consequence, we discarded cut impact measures based on the combined impact of several cutting planes. Instead we carried out the following steps for given cutting planes $a_1^\intercal x \leq \beta_1$, $a_2^\intercal x \leq \beta_2$, ..., $a_k^\intercal x \leq \beta_k$:

1. Compute the optimum $z^\star$ and optimum solution $x^\star \in P$.
2. Compute z^{LP}.
3. For $i = 0, 1, 2, \dots, k$, solve

$$\max c^\intercal x \text{ s.t. } Ax \leq b,\ x_j \in \mathbb{Z}\ \forall j \in I \text{ and } a_i^\intercal x \leq \beta_i \text{ if } i \geq 1$$

   with $x^\star$ as an initial incumbent, with presolve, cutting planes and heuristics disabled and where a time limit of 60 seconds is enforced.

4. Let N denote the minimum number of branch-and-bound nodes used in any these $k+1$ runs.
5. For $i = 0, 1, \dots, k$, let z^i be the dual bound obtained in run i when stopping after N branch-and-bound nodes. For $i \geq 1$, the *closed gap of cut i* is defined as $(z^i - z^{\text{LP}})/(z^\star - z^{\text{LP}})$. For $i = 0$, this yields the *closed gap without cuts*.

Remarks. The closed gap is essentially the dual bound, normalized such that a value of 0 means no bound improvement over the root LP bound without cuts and a value of 1 means that the instance was solved to optimality. The effective limit of N branch-and-bound nodes was introduced such that all runs reach this limit. This circumvents the question of how to compare runs in which the problem was solved to optimality with those that could not solve it. For all runs, presolve and domain propagation were disabled due to our focus on the branch-and-bound algorithm itself. To avoid interaction with heuristics, the latter were disabled, but the optimal solution $x^\star$ was provided.

4 Computational study

In order to test Hypothesis 1 we considered the 65 instances from the MIPLIB 3 [9]. For each of them we computed the dimension of the mixed-integer hull P, imposing a time limit of 10 minutes[1]. Moreover, we ran the state-of-the-art solver `SCIP` [3] and collected all cutting planes generated in the root node, including those that were discarded by `SCIP`'s cut selection routine[2]. For each of the cuts we computed the dimension of its induced face (see Section 2) as well as the closed gap after processing N (as defined in Section 3) branch-and-bound nodes.

While for instances `air03`, `air05`, `nw04`, no cuts were generated by `SCIP`, our implementation of Algorithm 1 ran into numerical difficulties during the computation of $\dim(P)$ for `set1ch`. Moreover, $\dim(P)$ could not be computed within 10 minutes for instances `air04`, `arki001`, `cap6000`, `dano3mip`, `danoint`, `dsbmip`, `fast0507`, `gesa2_o`, `gesa3_o`, `l152lav`, `mas74`, `misc06`, `mitre`, `mkc`, `mod010`, `mod011`, `pk1`, `pp08aCUTS`, `pp08a`, `qnet1`, `rentacar`, `rout` and `swath`.

We evaluated the remaining 37 instances, whose characteristic data is shown in Table 1. We distinguish how many cuts `SCIP` found by which cut separation method, in particular to investigate whether certain separation routines tend to generate cuts with low- or high-dimensional faces.

We verified some of the invalid cutting planes manually, i.e., checked that these cuts are generated by `SCIP` and that there exists a feasible solution that is indeed cut off. The most likely reason for their occurrence is that `SCIP` performed dual reductions although we disabled them via corresponding parameters[3].

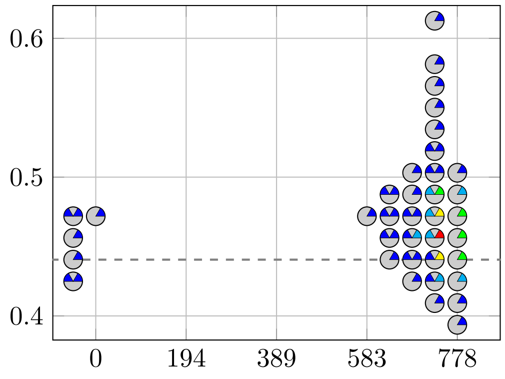
[1] All experiments were carried out on a single core of an Intel Core i3 CPU running at 2.10 GHz with 8 GB RAM.
[2] We disabled presolve, domain propagation, dual reductions, symmetry, and restarts.
[3] We set `misc/allowweakdualreds` and `misc/allowstrongdualreds` to `false`.

Table 1: Characteristics of the relevant 37 instances with number of successfully analyzed cuts, failures (numerical problems 0/0, timeouts ◷, invalid cuts ϟ), dimension of the mixed-integer hull, and number N of branch-and-bound nodes (see Section 3).
◔ – Lifted extended weight inequalities [14, 8, 13]
◕ – Complemented Mixed-Integer Rounding (c-MIR) inequalities [7, 14]
◑ – $\{0, 1/2\}$-Chvátal-Gomory inequalities [5, 2]
◓ – Strengthened Chvátal-Gomory inequalities [6]
◐ – Lifted flow-cover inequalities [4]
◒ – Multi-commodity flow inequalities [1]

Instance	**Cuts**	**Analyzed by class**						**Failed**			**Dim.**	**B&B**
	total	◔	◕	◑	◓	◐	◒	0/0	◷	ϟ	**of P**	**nodes**
bell3a	25		25								121	53075
bell5	53		36				1	11		5	97	7815
blend2	8		7							1	245	597
dcmulti	172		137					14		21	467	773
egout	125		97			1		24		3	41	1
enigma	82		59	1	17	5					3	1
fiber	533	127	139	7	4	9	207	33	6	1	946	23395
fixnet6	772		612				83	42		35	779	53900
flugpl	42		42								9	1753
gen	28	17	8					3			540	21
gesa2	470	16	419	2	6			27			1176	21114
gesa3	259		194	2	6	15		38	4		1104	729
gt2	143		92	3	39	8			1		188	1
harp2	1028	659	210	9	4	112		5	28	1	1300	13748
khb05250	122		78					5		39	1229	425
lseu	125	35	85	2	3						89	3495
markshare1	107		104			3					50	318260
markshare2	84		79			3			2		60	286347
mas76	225		204						1	20	151	211381
misc03	634	3	283	33	311				4		116	13
misc07	808		286	34	440			36	12		204	22839
mod008	441	119	272						3	47	319	1158
modglob	268		186					3	1	78	327	112516
noswot	164		151		11			2			120	160344
p0033	94	14	40		10			30			27	127
p0201	263	12	152	14	78			7			139	21
p0282	904	368	441	6	15	1		4		69	282	57
p0548	577	159	213	10	16	62		117			520	921
p2756	1000	82	96	22	48	55		26	671		2716	15149
qiu	63		51					2	10		709	10406
qnet1	162		95	10		47		6	4		1233	5
rgn	278		168			65		45			160	1691
seymour	6246		656	53	5528			9	0		1255	9
stein27	886		517	9	360						27	3673
stein45	1613		1221	10	382						45	45371
vpm1	281		129			32		117		3	288	27881
vpm2	353		251	1		30		63		8	286	172271

Since the results on face dimensions and cut strength turned out to be very instance-specific, we created one plot per instance. We omit the ones for `p2756` (too many failures, see Table 1), `blend2` (only 7 cuts analyzed), `enigma` ($\dim(P) = 3$ is very small), and for `markshare1`, `markshare2` and `noswot` (all cuts were ineffective). Moreover, we present several plots in the Appendix A since these are similar to those of other instances.

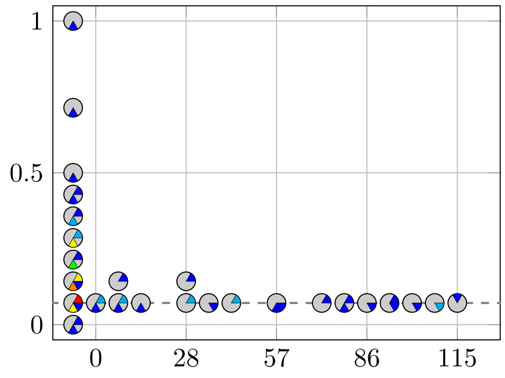
The plots show the dimension of the cuts (horizontal axis, rounded to 19 groups) together with their closed gap (vertical axis, 14 groups) according to Section 3. Each circle corresponds to a nonempty set of cuts, where the segments depict the respective cut classes (see Table 1) and their color depicts the number k of cuts, where the largest occurring number L is specified in the caption. The colors are red ($0.9L < k \leq L$), orange ($0.7L < k \leq 0.9L$), yellow ($0.5L < k \leq 0.7L$), green ($0.3L < k \leq 0.5L$), turquoise ($0.1L < k \leq 0.3L$) and blue ($1 \leq k < 0.1L$). For instance, the circle for `fixnet6` containing a red and a turquoise segment subsumes cutting planes with face dimensions between 730 and 777, and closed gap of approximately 0.45. As the legend next to the plot indicates, this circle represents $k \in [109, 121]$ c-MIR cuts and $k' \in [13, 36]$ multi-commodity flow cuts. The dashed horizontal line indicates the closed gap without cuts (see Section 3).



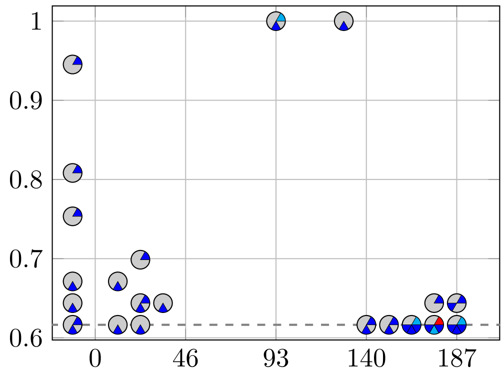
fixnet6: $L = 121$ cuts

Segment colors for `fixnet6`:

🟦 between 1 and 12 cuts
🟦 between 13 and 36 cuts
🟩 between 37 and 60 cuts
🟨 between 61 and 84 cuts
🟧 between 85 and 108 cuts
🟥 between 109 and 121 cuts

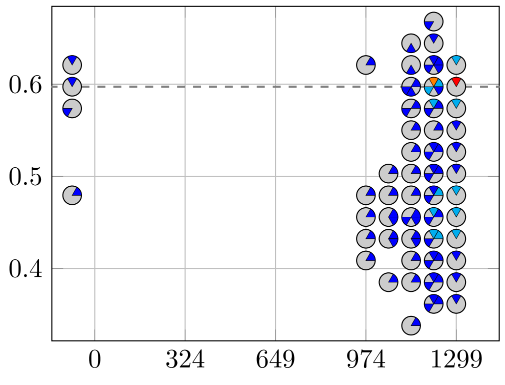


misc03: $L = 103$ cuts

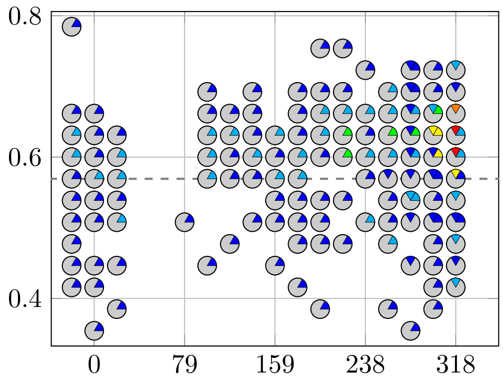


gt2: $L = 48$ cuts

The first three plots already highlight that the results are very heterogeneous: while the faces of the strongest cuts in `fixnet6` have a high dimension, the strongest ones for `misc03` are not even supporting. Even when considering non-supporting cuts as outliers, the dimension does not indicate practical strength, as the plot for `gt2` shows. A quick look at the other plots lets us conclude that Hypothesis 1 is false — at least for the strength measure from Section 3.

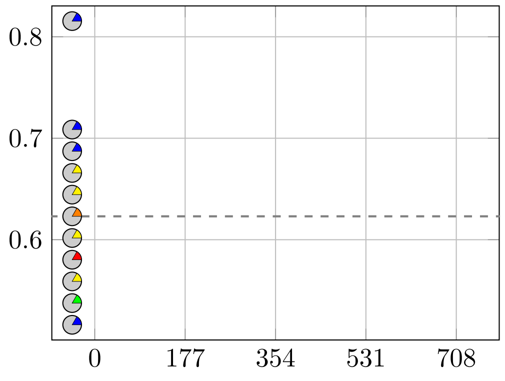


harp2: $L = 186$ cuts

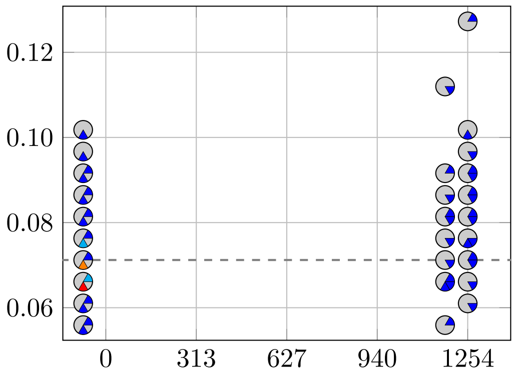


mod008: $L = 19$ cuts

The dashed line for `harp2` shows that adding a single cut does not necessarily help in branch-and-bound, which may be due to side-effects such as different branching decisions. For some instances, such as `mod008`, the cuts' face dimensions are well distributed. In contrast to this, some instances exhibit only very few distinct dimension values, e.g., only non-supporting cuts for `qiu`. Interestingly, the 6246 cuts for `seymour` induce only empty faces as well as faces with dimensions between 1218 and 1254. This can partially be explained via the cut classes. On the one hand, all generated strengthened Chvátal-Gomory cuts are non-supporting. On the other hand, some of the c-MIR cuts and $\{0, 1/2\}$-cuts are non-supporting while others induce faces of very high dimension.

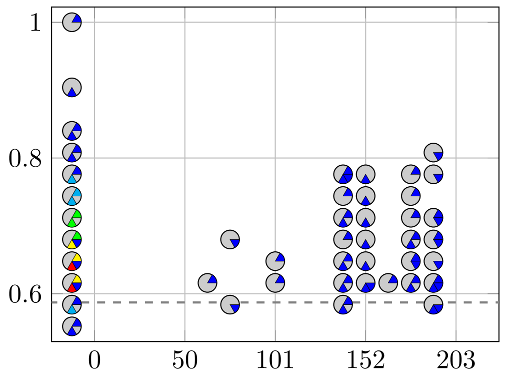


qiu: $L = 10$ cuts

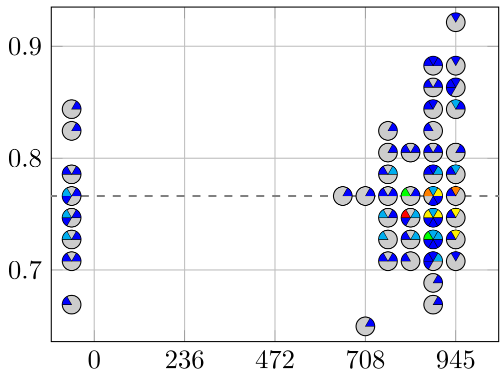


seymour: $L = 2703$ cuts

In general we don't see an indication that cuts from certain classes induce higher dimensional faces than others. At first glance, such a pattern is apparent for `misc07` (at dimensions 130–160), however one has to keep in mind that these blue segments constitute a minority of the cuts. In line with that, the majority of the cuts for `fiber` is concentrated around dimension 900 with a closed gap similar to that without cuts.



misc07: $L = 114$ cuts



fiber: $L = 39$ cuts

Despite the heterogeneity of the results, one observation is common to many instances: the distribution of the face dimensions is biased towards -1 and high-dimensions, i.e., not many cuts inducing low dimensional faces are generated.

A corresponding histogram is depicted in Fig. 1. We conjecture that the high dimensions occur because lifting and strengthening techniques for cutting planes are quite evolved.

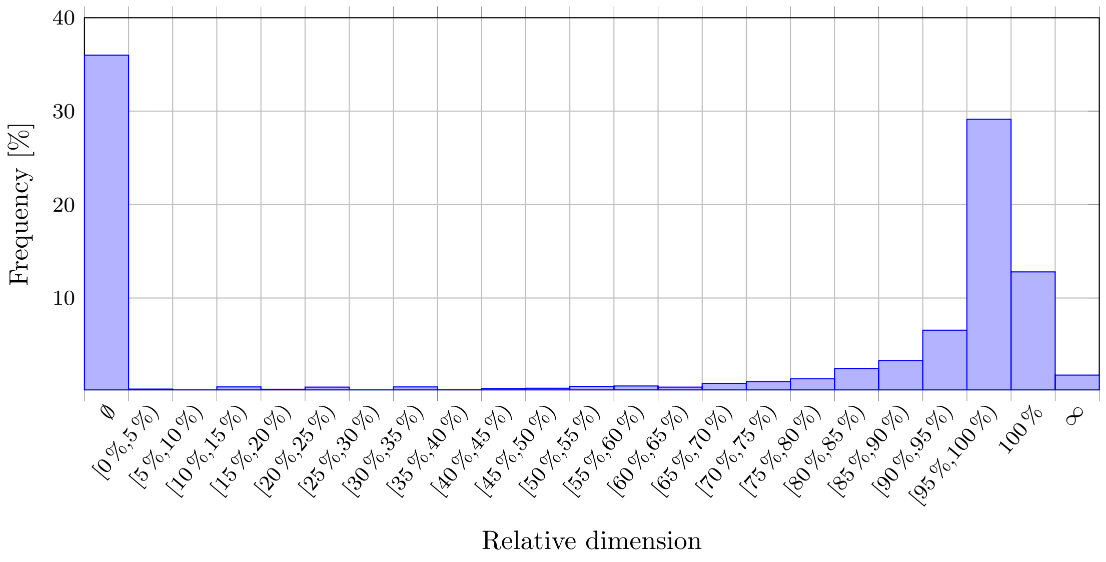


Figure 1: Distribution of relative dimensions over the 37 instances from Table 1 except for `p2756` (avoiding a bias due to many failures dimension computations). A cut of dimension k in an instance having ℓ cuts and with $\dim(P) = d$ contributes $1/(36\ell)$ to its bar. For $k = -1$ (resp. $k = d$), this is $\emptyset$ (resp. ∞), and it is $k/(d-1)$ otherwise.

The first bar in Fig. 1 indicates that for an instance chosen uniformly at random among the ones we considered and then a randomly chosen cut for this instance, this cut is non-supporting with probability greater than 35 %. This is remarkably high and thus we conclude this paper by proposing to investigate means to (heuristically) test for such a situation, with the goal of strengthening a non-supporting cutting plane by a reduction its right-hand side.

Acknowledgments. We thank R. Hoeksma and M. Uetz as well as the `SCIP` development team, in particular A. Gleixner, C. Hojny and M. Pfetsch for valuable suggestions on the computational experiments and their presentation.

References

[1] Tobias Achterberg and Christian Raack. The MCF-separator: detecting and exploiting multi-commodity flow structures in MIPs. *Mathematical Programming Computation*, 2(2):125–165, 2010.

[2] Alberto Caprara and Matteo Fischetti. {0, 1/2}-Chvátal-Gomory cuts. *Mathematical Programming*, 74(3):221–235, 1996.

[3] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. Technical report, Optimization Online, March 2020.

[4] Zonghao Gu, George L. Nemhauser, and Martin W. P. Savelsbergh. Lifted flow cover inequalities for mixed 0-1 integer programs. *Mathematical Programming*, 85(3):439–467, 1999.

[5] Arie M.C.A. Koster, Adrian Zymolka, and Manuel Kutschka. Algorithms to Separate $\{0, \frac{1}{2}\}$-Chvátal-Gomory Cuts. *Algorithmica*, 55(2):375–391, 2009.

[6] Adam N. Letchford and Andrea Lodi. Strengthening Chvátal–Gomory cuts and Gomory fractional cuts. *Operations Research Letters*, 30(2):74–82, 2002.

[7] Hugues Marchand and Laurence A. Wolsey. Aggregation and Mixed Integer Rounding to Solve MIPs. *Operations Research*, 49(3):363–371, 2001.

[8] Alexander Martin. Integer programs with block structure, 1999.

[9] Cassandra M. Mczeal, Martin W. P. Savelsbergh, and Robert E. Bixby. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[10] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986.

[11] Matthias Walter. *Investigating Polyhedra by Oracles and Analyzing Simple Extensions of Polytopes*. PhD thesis, Otto-von-Guericke-Universität Magdeburg, 2016.

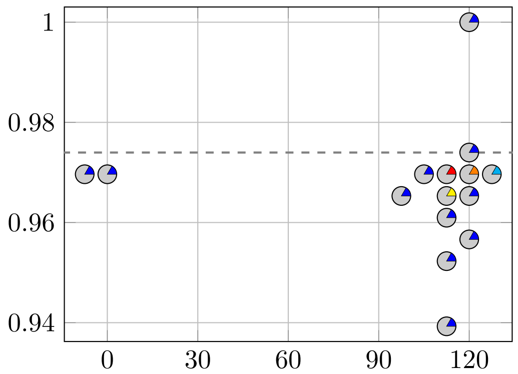
[12] Matthias Walter. IPO – Investigating Polyhedra by Oracles, 2016. Software available at: bitbucket.org/matthias-walter/ipo/.

[13] Robert Weismantel. On the 0/1 knapsack polytope. *Mathematical Programming*, 77(3):49–68, 1997.

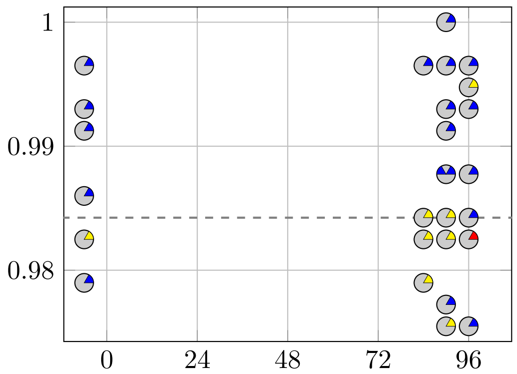
[14] Kati Wolter. Implementation of Cutting Plane Separators for Mixed Integer Programs. Master's thesis, Technische Universität Berlin, 2006.

A Additional plots

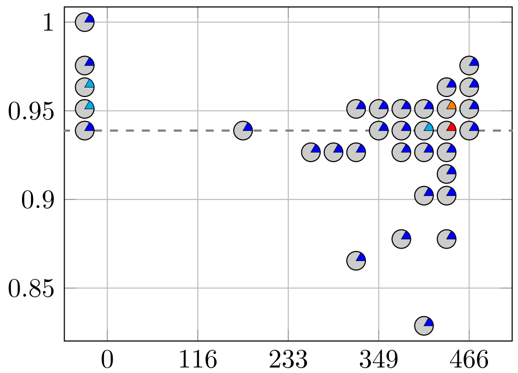
Here we provide additional instance-specific plots. This underlines the conclusions drawn in Section 4 and allows inspection of results for instances with certain characteristics (see Table 1).



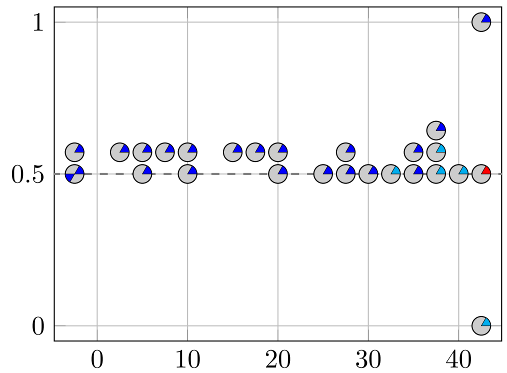
bell3a: $L = 5$ cuts



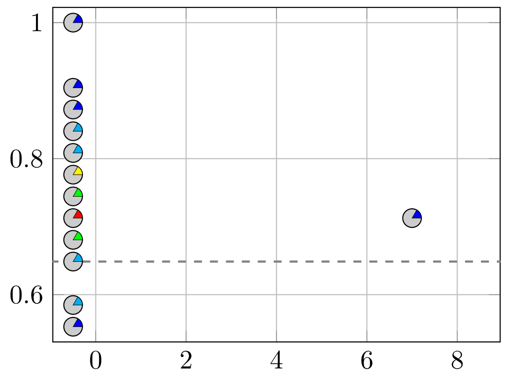
bell5: $L = 3$ cuts



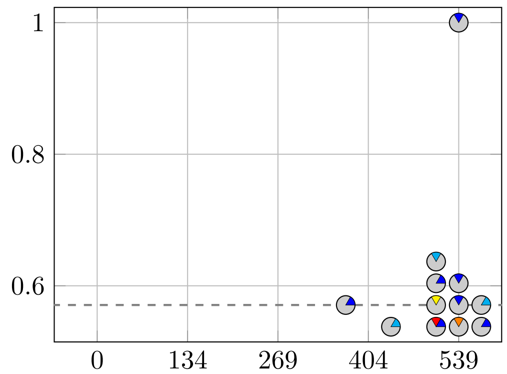
dcmulti: $L = 38$ cuts



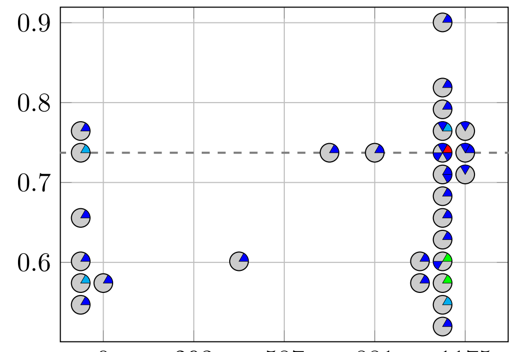
egout: $L = 31$ cuts



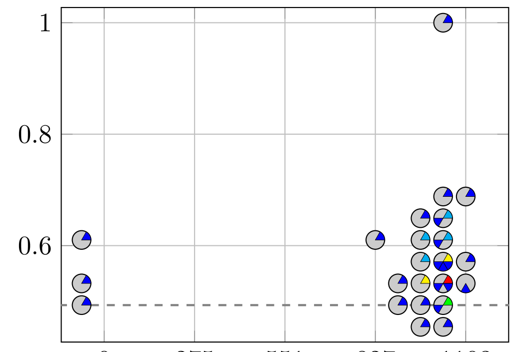
flugpl: $L = 11$ cuts



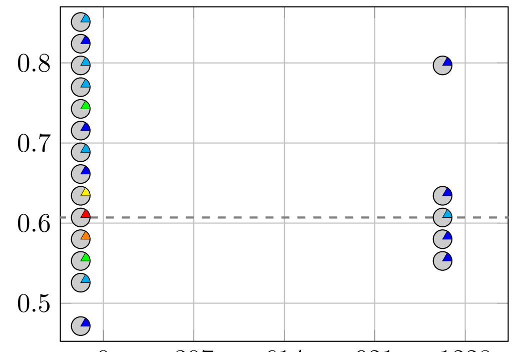
gen: $L = 5$ cuts



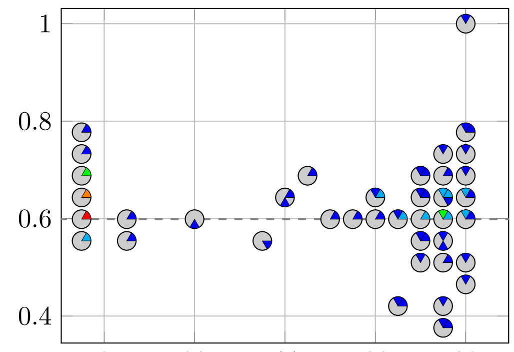
gesa2: $L = 139$ cuts



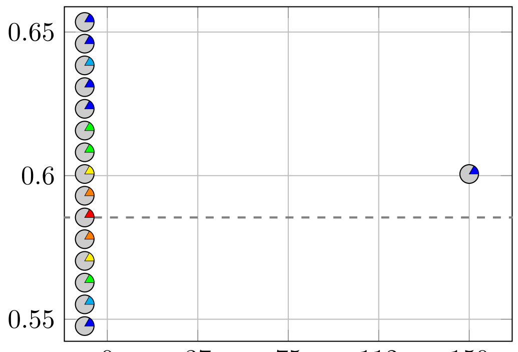
gesa3: $L = 51$ cuts



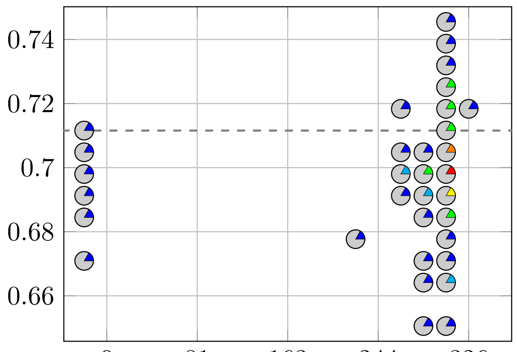
khb05250: $L = 15$ cuts



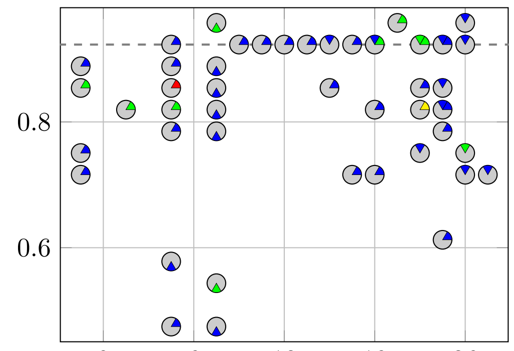
lseu: $L = 17$ cuts



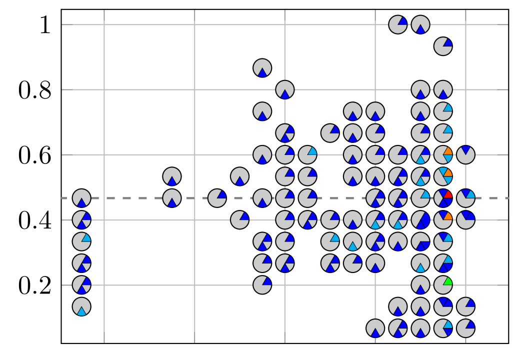
mas76: $L = 37$ cuts



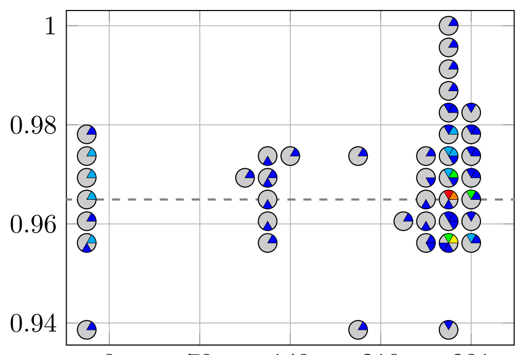
modglob: $L = 31$ cuts



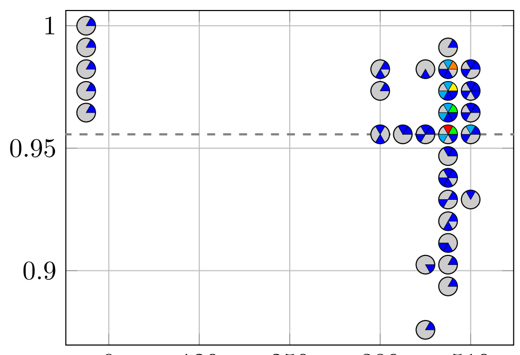
p0033: $L = 4$ cuts



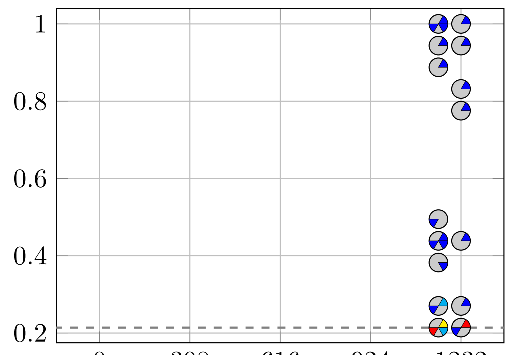
p0201: $L = 15$ cuts



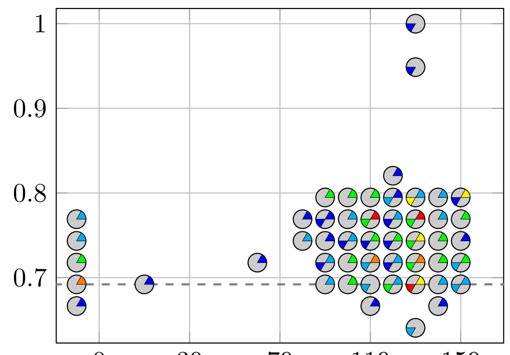
p0282: $L = 136$ cuts



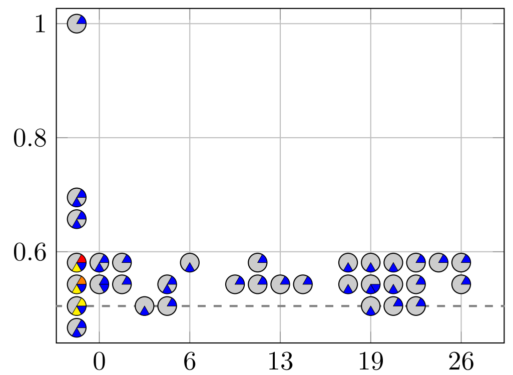
p0548: $L = 73$ cuts



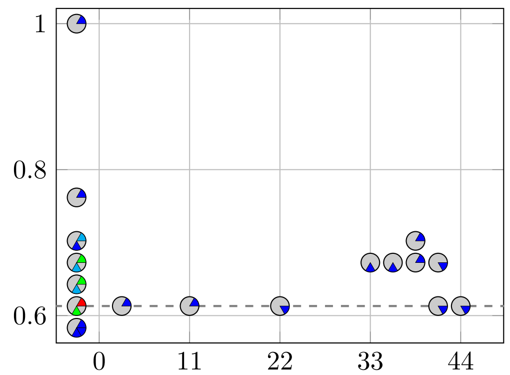
qnet1: $L = 40$ cuts



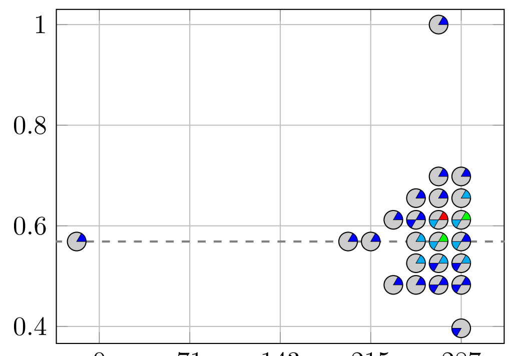
rgn: $L = 11$ cuts



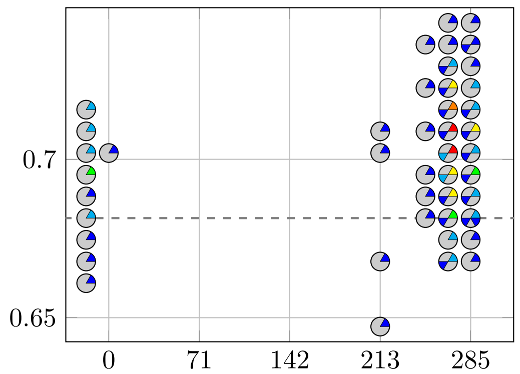
stein27: $L = 212$ cuts



stein45: $L = 625$ cuts



vpm1: $L = 39$ cuts



vpm2: $L = 25$ cuts