Fast Cube Tests for LIA Constraint Solving

Martin Bromberger, Christoph Weidenbach

Fast Cube Tests for LIA Constraint Solving

Martin Bromberger[1,2] and Christoph Weidenbach[1]

[1] Max Planck Institute for Informatics, Saarbrücken, Germany
{mbromber,weidenb}@mpi-inf.mpg.de
[2] Graduate School of Computer Science, Saarbrücken, Germany

Abstract. We present two tests that solve linear integer arithmetic constraints. These tests are sound and efficiently find solutions for a large number of problems. While many complete methods search along the problem surface for a solution, these tests use cubes to explore the interior of the problems. The tests are especially efficient for constraints with a large number of integer solutions, e.g., those with infinite lattice width. Inside the SMT-LIB benchmarks, we have found almost one thousand problem instances with infinite lattice width, and we have shown the advantage of our cube tests on these instances by comparing our implementation of the cube test with several state-of-the-art SMT solvers. Our implementation is not only several orders of magnitudes faster, but it also solves all instances, which most SMT solvers do not. Finally, we discovered an additional application for our cube tests: the extraction of equalities implied by a system of linear arithmetic inequalities. This extraction is useful both as a preprocessing step for linear integer constraint solving as well as for the combination of theories by the Nelson-Oppen method.

Keywords: Linear Arithmetic, SMT, Integer Arithmetic, Constraint Solving

1 Introduction

Finding an integer solution for a polyhedron that is defined by a system of linear inequalities $Ax \leq b$ is a well-known NP-complete problem [18]. Systems of linear inequalities have many real-world applications so that this problem has been investigated in different research areas, e.g., in optimization via *(mixed) integer linear programming* (MILP) [15] and in constraint solving via *satisfiability modulo theories* (SMT) [2,4,7,12].

It is standard for commercial MILP implementations to integrate preprocessing techniques, heuristics, and specialized tests [15]. Although these techniques are not complete, they are much more efficient on their designated target systems of linear inequalities than a complete algorithm alone. Since there exist specialized techniques for many classes of real-world problems representable as polyhedra, commercial MILP solvers are efficient on many real-world inputs—even though the problem, in general, is NP-complete.

The constraint solving community is still in the process of developing the same variety in specialized tests as the MILP community. The biggest challenge is to adopt the tests from the MILP community so that they still fit the input systems relevant for constraint solving. For example, SMT theory solvers have to solve a large number of incrementally connected, small systems of linear inequalities. Exploiting this incremental connection is key for making SMT theory solvers efficient [11]. In contrast, MILP solvers typically target one large system. The same holds for their specialized tests, which are not well suited to exploit incremental connections.

In this paper, we present two tests tailored toward SMT solvers: the *largest cube test* and the *unit cube test*. The idea is to find hypercubes that are contained inside the input polyhedron and guarantee the existence of an integer solution. Due to computational complexity, we will restrict ourselves to only those hypercubes that are parallel to the coordinate axes. The largest cube test finds a hypercube with maximum edge length contained in the input polyhedron, determines its real valued center, and rounds it to a potential integer solution. The unit cube test determines if a polyhedron contains a hypercube with edge length one, which is the minimal edge length that guarantees an integer solution.

Most SMT linear integer arithmetic theory solvers are based on a branch-and-bound algorithm on top of the simplex algorithm. They search for a solution at the surface of a polyhedron. However, our tests search in the interior of the polyhedron. This gives them an advantage on polyhedra with a large number of integer solutions, e.g., polyhedra with infinite lattice width [16]. Since the only difference between the input polyhedron $Ax \leq b$ and the associated unit cube polyhedron $Ax \leq b'$ are the row bounds, our unit cube test is especially easy to implement and integrate into SMT theory solvers.

SMT theory solvers are designed to efficiently exchange bounds [9]. This efficient exchange is the main reason why SMT theory solvers exploit the incremental connection between the different polyhedra so well. Our unit cube test also requires only an exchange of bounds. After applying the test, we can easily recover the original polyhedron by reverting to the original bounds. In doing so, the unit cube test conserves the incremental connection between the different original polyhedra. We make a similar observation about the largest cube test.

A variant of the linear program for the unit cube test first appeared in 1969 as a subroutine in a heuristic by Hillier for MILP optimization [13]. While Hillier was aware of the unit cube test, he applied it only to cones, a special class of polyhedra. His work never mentioned applications beyond cones, nor did he prove any structural properties connected to hypercubes. As mentioned before, the main advantage of the cube tests is that they compute interior point candidates. The same can be done using an interior point method [17] instead of the simplex algorithm. Therefore, Hillier's heuristic tailored for MILP optimization lost popularity as soon as interior point methods became efficient in practice. Nonetheless, our cube tests remain relevant for SMT theory solvers because there are no competitive incremental interior point methods.

Also, Bobot et al. discuss relations between hypercubes, called ∞-norm balls, and polyhedra [2]. In their paper, they detail the same relation between polyhedra with infinite lattice width and hypercubes that we discovered. Their work also includes a linear optimization program that detects polyhedra with infinite lattice width and positive linear combinations between inequalities. Our largest cube test can detect all of the above because it is, with some minor changes, the dual of the linear optimization program of Bobot et al. However, our tests are a lot closer to the original polyhedron and are, therefore, easier to construct, and the tests produce sample points as well. Via rounding, our tests use these sample points to compute an actual integer solution as proof. Moreover, our cube tests also find solutions for polyhedra with finite lattice width.

Our contributions are as follows: we define the linear cube transformation (Corollary 3) that allows us to efficiently compute whether a polyhedron $Ax \leq b$ contains a hypercube of edge length e by solely changing the bounds b in Section 3. Based on this transformation, we develop in Section 4 two tests: the largest cube test and the unit cube test. For polyhedra with infinite lattice width, both tests always succeed (Lemma 5). Inside the SMT-LIB benchmarks, there are almost one thousand problem instances with infinite lattice width, and we show the advantage of our cube tests on these instances by comparing our implementation of the cube test with several state-of-the-art SMT solvers in Section 5. Our implementation is not only several orders of magnitudes faster, but it also solves all instances, which most SMT solvers do not (Figure 7). It is more robust than the test suggested by Bobot et al. [2] (Figure 7). Eventually, we introduce in Section 6 an additional application for our cube tests: the extraction of equalities implied by a system of linear arithmetic inequalities. The paper ends with a discussion on possible directions for future research, Section 7.

2 Preliminaries

While the difference between matrices, vectors, and their components is always clear in context, we generally use upper case letters for matrices (e.g., A), lower case letters for vectors (e.g., x), and lower case letters with an index i or j (e.g., b_i, x_j) as components of the associated vector at position i or j, respectively. The only exceptions are the row vectors $a_i^T = (a_{i1}, \ldots, a_{in})$ of a matrix $A = (a_1, \ldots, a_m)^T$, which already contain an index i that indicates the row's position inside A. In order to save space, we write vectors only implicitly as columns via the transpose $(\ )^T$ operator, which turns all rows $(b_1, \ldots, b_m)$ into columns $(b_1, \ldots, b_m)^T$ and vice versa. We will also abbreviate $(\ldots, 0, \ldots)^T$ as $\mathbf{0}$.

In this paper, we treat *polyhedra* and their definitions through a *system of inequalities* $Ax \leq b$ as interchangeable. For such a system of inequalities, the row coefficients are given by $A = (a_1, \ldots, a_m)^T \in \mathbb{Q}^{m \times n}$, the inequality bounds are given by $b = (b_1, \ldots, b_m)^T \in \mathbb{Q}^m$, and the variables are given by $x = (x_1, \ldots, x_n)^T$.

We denote by $P_b^A = \{x \in \mathbb{R}^n : Ax \leq b\}$ the *set of real solutions* to the system of inequalities $Ax \leq b$ and, therefore, the points inside the polyhedron.

Similarly, we denote by $C_e^n(z) = \left\{x \in \mathbb{R}^n : \forall j \in 1,\ldots,n.\ |x_j - z_j| \leq \frac{e}{2}\right\}$ the set of points contained in the *n-dimensional hypercube* $C_e^n(z)$ that is parallel to the coordinate axes, has *edge length* $e \in \mathbb{R}_{\geq 0}$, and has *center* $z \in \mathbb{R}^n$. For the remainder of this paper, we will consider only hypercubes that are parallel to the coordinate axes. For simplicity, we call these restricted hypercubes *cubes*. Similar to polyhedra, we will use the set of points $C_e^n(z)$ interchangeably with the cube defined by the set.

Besides cubes and polyhedra, we use multiple p-norms $\|.\|_p$ in this paper [10]. These p-norms are defined as functions $(\|.\|_p : \mathbb{R}^n \to \mathbb{R})$ for $p \geq 1$ such that $\|x\|_p = (|x_1|^p + \ldots + |x_n|^p)^{1/p}$. A special p-norm is the *maximum norm*. It is defined by the limit of $\|.\|_p$ for $p \to \infty$: $\|x\|_\infty = \max\{|x_1|,\ldots,|x_n|\}$. If we compare the maximum norm and the definition of $C_e^n(z)$, we see that cubes and p-norms are related: $\left(\|x - z\|_\infty \leq \frac{e}{2}\right) \iff \left(\forall j \in 1,\ldots,n.\ |x_j - z_j| \leq \frac{e}{2}\right)$.

Using p-norms, we define a *closest integer* for a point x as a point $x' \in \mathbb{Z}^n$ with minimal distance $\|x - x'\|_p$ for all p-norms. We also define the operators $\lceil x_j \rfloor$ and $\lceil x \rfloor$ such that they describe a *closest integer* for x_j and x, respectively. Formally, this means that $\lceil x \rfloor = (\lceil x_1 \rfloor,\ldots,\lceil x_n \rfloor)^T$ and

$$\lceil x_j \rfloor = \begin{cases} \lfloor x_j \rfloor & \text{if } x_j - \lfloor x_j \rfloor < 0.5\,, \\ \lceil x_j \rceil & \text{if } x_j - \lfloor x_j \rfloor \geq 0.5\,. \end{cases}$$

This definition of $\lceil x \rfloor$ is also known as *simple rounding*.

Lemma 1. *For $x \in \mathbb{R}^n$, $\lceil x \rfloor$ is a closest integer to x:*

$$\forall p \geq 1.\ \forall x' \in \mathbb{Z}^n.\ \|x - \lceil x \rfloor\|_p \leq \|x - x'\|_p\ .$$

Proof. We first look at the one-dimensional case, where $\|x_j\|_p$ simplifies to $|x_j|$:

$$\forall p \geq 1.\ \forall x'_j \in \mathbb{Z}.\ |x_j - \lceil x_j \rfloor| \leq |x_j - x'_j|\,.$$

For $\lceil x_j \rfloor, x'_j \in \mathbb{Z}$, there exists $z_j \in \mathbb{Z}$ such that $x'_j = \lceil x_j \rfloor - z_j$. For $x_j \in \mathbb{R}$, there exists a $d_j \in [-0.5, 0.5]$ such that $d_j := x_j - \lceil x_j \rfloor$. The inequality trivially holds for $z_j = 0$:

$$|x_j - x'_j| = |x_j - \lceil x_j \rfloor + z_j| = |x_j - \lceil x_j \rfloor|\,.$$

Via the triangle inequality, for the remaining $z_j \neq 0$ we get :

$$|x_j - x'_j| = |x_j - \lceil x_j \rfloor + z_j| = |d_j + z_j| \geq |z_j| - |d_j|\,.$$

Since $z_j \neq 0$, and $d_j \in [-0.5, 0.5]$ imply $|z_j| \geq 1$, and $|d_j| \leq 0.5$, respectively, we get:

$$|x_j - x'_j| \geq |z_j| - |d_j| \geq 1 - |d_j| \geq 0.5 \geq |d_j| = |x_j - \lceil x_j \rfloor|\,.$$

The multidimensional case follows from the p-norms' monotonicity [10], i.e., if $|x_j - \lceil x_j \rfloor| \leq |x_j - x'_j|$ for all $j \in \{1,\ldots,n\}$, then $\|x - \lceil x \rfloor\|_p \leq \|x - x'\|_p$. □

3 Fitting Cubes into Polyhedra

We say that a cube $C_e^n(z)$ *fits* into a polyhedron defined by $Ax \leq b$ if all points inside the cube $C_e^n(z)$ are solutions of $Ax \leq b$, or formally: $C_e^n(z) \subseteq P_b^A$. In order to compute this, we transform the polyhedron $Ax \leq b$ into another polyhedron $Ax \leq b'$. For this new polyhedron, we merely have to test whether the cube's center point z is a solution ($z \in P_{b'}^A$) in order to also determine whether the

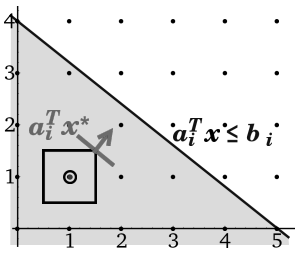

Fig. 1. A square (two-dimensional cube) fitting into an inequality $a_i^T x \leq b_i$ and the cube's maximum $a_i^T x^*$ for the objective $a_i^T x$

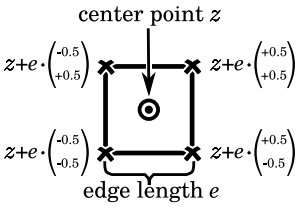

Fig. 2. The vertices of an arbitrary square parallel to the coordinate axes (two-dimensional cube with edge length e and center z)

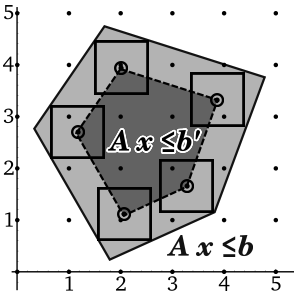

Fig. 3. The transformed polyhedron $Ax \leq b'$ for edge length 1 together with the original polyhedron $Ax \leq b$

cube $C_e^n(z)$ fits into the original polyhedron ($C_e^n(z) \subseteq P_b^A$). This is a simple test that requires only evaluation. We call this entire transformation the *linear cube transformation*.

We start explaining the linear cube transformation by looking at the case where the polyhedron is defined by a single inequality $a_i^T x \leq b_i$. A cube $C_e^n(z)$ fits into the inequality $a_i^T x \leq b_i$ if all points inside the cube $C_e^n(z)$ are solutions of $a_i^T x \leq b_i$, or formally: $\forall x \in C_e^n(z).\ a_i^T x \leq b_i$.

We can think of $a_i^T x$ as an objective function that we want to maximize and see b_i as a guard for the maximum objective of any solution in the cube. Thus, we can express the universal quantifier in the above equation as an optimization problem (see Figure 1): $\max\{a_i^T x : x \in C_e^n(z)\} \leq b_i$. This also means that all points in $x \in C_e^n(z)$ satisfy the inequality $a_i^T x \leq b_i$ if a point $x^* \in C_e^n(z)$ with maximum value $a_i^T x^* = \max\{a_i^T x : x \in C_e^n(z)\}$ for the objective function $a_i^T x$ satisfies the inequality $a_i^T x^* \leq b_i$. We can formalize the above optimization problem as a linear program:

$$\begin{array}{ll} \text{maximize} & a_i^T x \\ \text{subject to} & z_j - \frac{e}{2} \leq x_j \leq z_j + \frac{e}{2} \quad \text{for } j = 1, \ldots, n\,. \end{array}$$

However, for the case of cubes, there is an even easier way to determine the maximum objective value. Since every cube is a bounded polyhedron, one of the points with maximum objective value is a vertex $x^v \in C_e^n(z)$. A vertex x^v of the cube $C_e^n(z)$ is one of the points with maximum distance to the center z (see Figure 2), or formally: $x^v = \left(z_1 \pm \frac{e}{2}, \ldots, z_n \pm \frac{e}{2}\right)^T$. If we insert the above equation into the objective function $a_i^T x$, we get:

$$a_i^T \left(z_1 \pm \tfrac{e}{2}, \ldots, z_n \pm \tfrac{e}{2}\right)^T = a_i^T z + \tfrac{e}{2} \textstyle\sum_{j=1}^n \pm a_{ij}\,,$$

which in turn is maximal if we choose x^v such that $\pm a_{ij}$ is always positive:

$$a_i^T x^v = a_i^T z + \tfrac{e}{2} \textstyle\sum_{j=1}^n |a_{ij}| = a_i^T z + \tfrac{e}{2} \left\|a_i\right\|_1\,.$$

Hence, we transform the inequality $a_i^T x \leq b_i$ into $a_i^T x \leq b_i - \frac{e}{2} \left\|a_i\right\|_1$, and $C_e^n(z)$ fits into $a_i^T x \leq b_i$ if $a_i^T z \leq b_i - \frac{e}{2} \left\|a_i\right\|_1$.

Corollary 2. *Let $C_e^n(z)$ be a cube and $a_i^T x \leq b_i$ be an inequality. All $x \in C_e^n(z)$ fulfill $a_i^T x \leq b_i$ if and only if $a_i^T z \leq b_i - \frac{e}{2} \|a_i\|_1$.*

Next, we look at the case where multiple inequalities $a_i^T x \leq b_i$ (for $i = 1, \ldots, m$) define the polyhedron $Ax \leq b$. Since P_b^A is the intersection of all $P_{b_i}^{a_i}$, the cube fits into $Ax \leq b$ if and only if it fits into all inequalities $a_i^T x \leq b_i$, respectively:

$$\forall i \in \{1, \ldots, m\}.\ \forall x \in C_e^n(z).\ a_i^T x \leq b_i\,.$$

We can express this by m optimization problems:

$$\forall i \in \{1, \ldots, m\}.\ \max\{a_i^T x : x \in C_e^n(z)\} \leq b_i$$

and, after applying Corollary 2, by the following m inequalities:

$$\forall i \in \{1, \ldots, m\}.\ a_i^T z \leq b_i - \tfrac{e}{2} \|a_i\|_1\ .$$

Hence, the linear cube transformation transforms the polyhedron $Ax \leq b$ into the polyhedron $Ax \leq b'$, where $b'_i = b_i - \frac{e}{2} \|a_i\|_1$, and $C_e^n(z)$ fits into $Ax \leq b$ if $Az \leq b'$.

Corollary 3. *Let $C_e^n(z)$ be a cube and $Ax \leq b$ be a polyhedron. $C_e^n(z) \subseteq P_b^A$ if and only if $Az \leq b'$, where $b'_i = b_i - \frac{e}{2} \|a_i\|_1$.*

Until now, we have discussed how to use the linear cube transformation to determine if one cube $C_e^n(z)$ with fixed center point z fits into a polyhedron $Ax \leq b$. A generalization of this problem determines whether a polyhedron $Ax \leq b$ contains a cube of edge length e at all. Actually, a closer look at the transformed polyhedron $Ax \leq b'$ reveals that the linear cube transformation ($b'_i = b_i - \frac{e}{2} \|a_i\|_1$) is dependent only on the edge length e of the cube. Therefore, the solutions $P_{b'}^A$ of the transformed polyhedron $Ax \leq b'$ are exactly all center points of cubes with edge length e that fit into the original polyhedron $Ax \leq b$ (see Figure 3). By determining the satisfiability of the transformed polyhedron $Ax \leq b'$, we can now also determine whether a polyhedron $Ax \leq b$ contains a cube of edge length e at all. If we choose a suitable algorithm, e.g., the simplex algorithm, then we even get the center point z of a cube $C_e^n(z)$ that fits into $Ax \leq b$. This observation is the foundation for the cube tests that we will present in Section 4.

4 Fast Cube Tests

In contrast to arbitrary polyhedra, determining whether a cube $C_e^n(z)$ contains an integer point is easy. Because of the cubes symmetry, it is enough to test whether it contains a closest integer point $\lceil z \rfloor$ to the center z.

Lemma 4. *A cube $C_e^n(z)$ contains an integer point if and only if it contains a closest integer point $\lceil z \rfloor$ to the center z.*

Proof. The implication from left to right follows directly from Lemma 1 and from the relation between the maximum norm and cubes. The implication from right to left is obvious. □

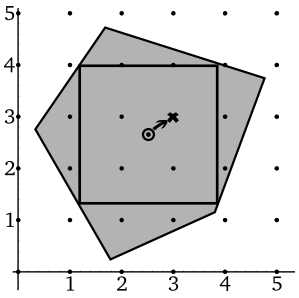

Fig. 4. The largest cube inside a polyhedron, its center point, and a closest integer point to the center

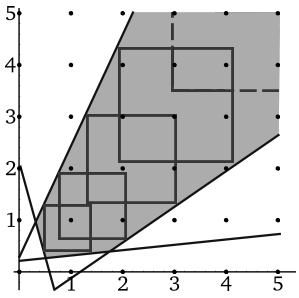

Fig. 5. An infinite lattice width polyhedron, containing cubes for every edge length $e > 0$.

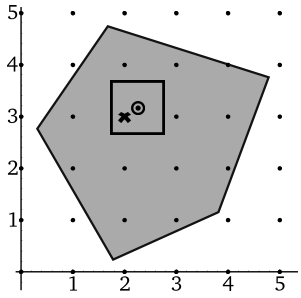

Fig. 6. A unit cube inside a polyhedron, its center point, and a closest integer point to the center

Note that every point $z \in \mathbb{R}^n$ is also a cube $C_0^n(z)$ of edge length 0. In order to be efficient, our tests will look only at cubes with special properties. In the case of the largest cube test, we check for an integer solution in one of the largest cubes fitting into the polyhedron $Ax \leq b$. In the case of the unit cube test, we look for a cube of edge length one, which always guarantees an integer solution. Due to these restrictions, both tests are not complete but very fast to compute.

4.1 Largest Cube Test

A well-known test, implemented in most ILP solvers, is *simple rounding*. For simple rounding, the ILP solver computes a real solution x for a set of inequalities, *rounds* it to a closest integer $\lceil x \rfloor$, and determines whether this point is an integer solution. Not all types of real solutions are good candidates for this test to be successful. Especially *surface points*, such as *vertices*, the usual output of the simplex algorithm, are not good candidates for rounding. For many polyhedra, *center and interior points* z are a better choice because all integer points adjacent to z are solutions, including a closest integer point $\lceil z \rfloor$.

To calculate a real center point with the simplex algorithm, we use the linear cube transformation (Section 3). The center point will be the center point of a largest cube that fits into the polyhedron $Ax \leq b$ (see Figure 4). We determine the center z of this largest cube and the associated edge length e with the following LP:

$$\begin{array}{ll} \text{maximize} & x_e \\ \text{subject to} & Ax + a' \frac{x_e}{2} \leq b, \text{ where } a'_i = \|a_i\|_1 \\ & x_e \geq 0\,. \end{array}$$

This linear program employs the linear cube transformation from Section 3. The only generalization is a variable x_e for the edge length instead of a constant value e. Additionally, this linear program maximizes the edge length as an optimization goal.

If the resulting maximum edge length is unbounded, the original polyhedron contains cubes of arbitrary edge length (see Figure 5) and, thus, infinitely many integer solutions. Since the linear program contains all solutions of the original polyhedron (see $x_e = 0$), the original polyhedron is empty if and only if the above linear program is infeasible. If the maximum edge length is a finite value e, we use the resulting assignment z for the variables x as a center point and $C_e^n(z)$ is a largest cube that fits into the polyhedron. From the center point, we round to a closest integer point $\lceil z \rfloor$ and determine if it fits into the original polyhedron. If this is the case, we are done because we have found an integer solution for $Ax \leq b$. Otherwise, the largest cube test does not know whether or not $Ax \leq b$ has an integer solution. An example for the latter case, are the following inequalities: $3x_1 - x_2 \leq 0$, $-2x_1 - x_2 \leq -2$, and $-2x_1 + x_2 \leq 1$. These inequalities have exactly one integer solution $(1, 3)^T$, but the largest cube contained by the inequalities has edge length $e = \frac{3}{17}$ and center point $(\frac{3}{17}, \frac{3}{2})^T$, which rounds to $(0, 2)^T$.

Instead of a cube, it is also possible to use a ball to compute a center point. The result is the *Chebyshev center* [3], i.e., the center of a largest ball that fits into the polyhedron:

$$\begin{array}{ll} \text{maximize} & x_r \\ \text{subject to} & Ax + a'x_r \leq b, \text{ where } a'_i = \|a_i\|_2 \\ & x_r \geq 0\,. \end{array}$$

However, the coefficients a'_i are then defined via the 2-norm $\|a_i\|_2 = \sqrt{\sum_{j=1}^{n} a_{ij}^2}$ and are, therefore, potentially irrational. As theory solvers in the SMT context use exact rational arithmetic, the Chebyshev center is not straightforward to integrate.

The largest cube test also upholds the incremental advantages of the dual simplex algorithm proposed by Dutertre and de Moura [9]. The only difference is the extra column $a' \frac{x_e}{2}$, which the theory solver can internally create while it is notified of all potential arithmetic literals. Adding this column from the start does not influence the correctness of the solution because $x_e \geq 0$ guarantees that the largest cube test is satisfiable exactly when the original inequalities $Ax \leq b$ are satisfiable. Even for explanations of unsatisfiability, it suffices to remove the bound $x_e \geq 0$ to obtain an explanation for the original inequalities $Ax \leq b$. The only disadvantage is the additional variable x_e. However, increasing x_e only shrinks the search space. Therefore, increasing x_e can never resolve any conflicts during the satisfiability search. The simplex solver recognizes this with at least one additional pivot that sets x_e to 0. Hence, adding the extra column $a' \frac{x_e}{2}$ from the beginning has only constant influence on the theory solver's run-time, and is therefore negligible.

4.2 Unit Cube Test

Most SMT theory solvers implement a simplex algorithm that is specialized towards feasibility and not towards optimization [1,6,9,12]. Therefore, a test based on optimization, such as the largest cube test, does not fit well with

existing implementations. As an alternative, we have developed a second test based on cubes that does not need optimization.

We avoid optimization by fixing the edge length e to the value 1 for all the cubes $C_e^n(z)$ we consider (see Figure 6). We do so because cubes $C_1^n(z)$ of edge length 1 are the smallest cubes to always guarantee an integer solution, completely independent of the center point z. A cube with edge length 1 is also called a *unit cube*. To prove this guarantee, we first fix $e = 1$ in the definition of cubes, $C_1^n(z) = \left\{x \in \mathbb{R}^n : \forall j \in 1, \ldots, n.\ |x_j - z_j| \leq \frac{1}{2}\right\}$, and look at the following property for the rounding operator $\lceil . \rfloor$: $\forall z_j \in \mathbb{R}. |\lceil z_j \rfloor - z_j| \leq \frac{1}{2}$. We see that any unit cube contains a closest integer $\lceil z \rfloor$ to its center point z. Furthermore, 1 is the smallest edge length that guarantees an integer solution for a cube with center point $z = (\ldots, \frac{1}{2}, \ldots)^T$. Thus, 1 is the smallest value that we can fix as an edge length to guarantee an integer solution for all cubes $C_1^n(z)$.

Our second test tries to find a unit cube that fits into the polyhedron $Ax \leq b$ and, thereby, a guarantee for an integer solution for $Ax \leq b$. Again, we employ the linear cube transformation from Section 3 and obtain the linear program:

$$Az \leq b', \text{ where } b'_i = b_i - \tfrac{1}{2} \left\|a_i\right\|_1 .$$

In addition to being a linear program without an optimization objective, we only have to change the row bounds b'_i of the original inequalities. In the dual simplex algorithm proposed by Dutertre and de Moura [9] and implemented in many SMT theory solvers [1,6,9,12], such a change of bounds is already part of the framework so that integrating the unit cube test into theory solvers is possible with only minor adjustments to the existing implementation. Since our unit cube test requires only an exchange of bounds, we can easily return to the original polyhedron by reverting the bounds. In doing so, the unit cube test upholds the incremental connection between the different original polyhedra.

5 Experiments

While our tests are useful for many types of polyhedra, the motivation for our tests stems from a special type of polyhedra, so-called *infinite lattice width* polyhedra [16]. A polyhedron $Ax \leq b$ has *infinite lattice width* if for every objective $c \in \mathbb{R}^n \setminus \{\mathbf{0}\}$, either its maximum or minimum objective value is unbounded, or formally:

$$\forall c \in \mathbb{R}^n \setminus \{\mathbf{0}\}.\ \sup\left\{c^T x \mid x \in P_b^A\right\} = \infty \text{ or } \inf\left\{c^T x \mid x \in P_b^A\right\} = -\infty .$$

Polyhedra with infinite lattice width seem trivial at first glance because their interior expands arbitrarily far in all directions (see Figure 5). Therefore, a polyhedron with infinite lattice width contains an infinite number of integer solutions [16]. Nonetheless, many SMT theory solvers have proven to be inefficient on those polyhedra because they use a branch-and-bound approach with an underlying simplex solver [9]. Although such an approach will terminate inside finite a priori bounds [18], it does not explore the infinite interior, but rather directs the search along the solutions suggested by the simplex solver: the vertices of the polyhedron. Thus, the SMT theory solvers concentrate their search on a bounded part of the polyhedron. This bounded part contains only a finite

number of integer solutions, whereas the complete interior contains infinitely many integer solutions. The advantage of our cube tests is that they actually exploit the infinite interior because polyhedra with infinite lattice width contain cubes for every edge length (see Figure 5). Our tests are always successful on polyhedra with infinite lattice width and usually need only a small number of pivoting steps before finding a solution.

Lemma 5. *Let $Ax \leq b$ be a polyhedron. Let $a' \in \mathbb{Z}^m$ be a vector such that its components are $a'_i = \|a_i\|_1$. Then, the following two statements are equivalent:*
(1) $Ax \leq b$ contains a cube $C_e^n(z)$ for every $e \in \mathbb{R}_{\geq 0}$, and
(2) $Ax \leq b$ has infinite lattice width.
Or formally:
(1) $\forall e \in \mathbb{R}_{\geq 0}.\ \exists x \in \mathbb{R}^n.\ Ax \leq b - \frac{e}{2} \cdot a'$,
(2) $\forall c \in \mathbb{R}^n \setminus \{\mathbf{0}\}.\ \sup\left\{c^T x \mid x \in P_b^A\right\} = \infty$ or $\inf\left\{c^T x \mid x \in P_b^A\right\} = -\infty$.

Proof. (1) $\Rightarrow$ (2): We first assume that $Ax \leq b$ contains a cube $C_e^n(z)$ for every $e \in \mathbb{R}_{\geq 0}$. Note that the center point z depends on the edge length e. Furthermore, we define the function:

$$\text{width}(c, S) = \left(\sup\left\{c^T x \mid x \in S)\right\} + \sup\left\{-c^T x \mid x \in S)\right\}\right) \tag{1}$$

for every vector $c \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ and for every set of points $S \subseteq \mathbb{R}^n$. Then, we prove that:

$$\lim_{e\to\infty} \text{width}(c, C_e^n(.)) \to \infty\,.$$

In Section 3, we have shown that:

$$\sup\left\{c^T x \mid x \in C_e^n(z)\right\} = c^T z + \tfrac{e}{2} \cdot \|c\|_1\ , \text{ and} \tag{2}$$

$$\sup\left\{-c^T x \mid x \in C_e^n(z)\right\} = -c^T z + \tfrac{e}{2} \cdot \|c\|_1\ . \tag{3}$$

Therefore, $\text{width}(c, C_e^n(z)) = e \cdot \|c\|_1$, which is independent of z. After inserting (2) and (3) into (1), we get:

$$\lim_{e\to\infty} \text{width}(c, C_e^n(.)) = \lim_{e\to\infty} e \cdot \|c\|_1 \to \infty\,.$$

Since $Ax \leq b$ contains cubes $C_e^n(z)$ for all $e \in \mathbb{R}$, it holds for all $e \in \mathbb{R}$ that

$$\text{width}(c, P_b^A) \geq \text{width}(c, C_e^n(.))\,,$$

and, thus, $\text{width}(c, P_b^A) = \infty$. Since P_b^A is also convex, it must hold that:

$$\sup\left\{c^T x \mid x \in P_b^A\right\} = \infty \text{ or } \inf\left\{c^T x \mid x \in P_b^A\right\} = -\infty\,.$$

(2) $\Rightarrow$ (1): By contradiction. Assume that $Ax \leq b$ has infinite lattice width but that there exists an $e \in \mathbb{R}_{\geq 0}$ such that $Ax \leq b$ contains no cube $C_e^n(z)$ of edge length e. By Corollary 3, $Ax \leq b$ contains no cube $C_e^n(z)$ of edge length e implies that $Ax \leq b - \frac{e}{2} \cdot a'$ is unsatisfiable. By Farkas Lemma [3], $Ax \leq b - \frac{e}{2} \cdot a'$ is unsatisfiable implies that there exists a $y \in \mathbb{R}^m$ such that: (a) $y_i \geq 0$ for all $i \in \{1, \ldots, m\}$, (b) $y_k > 0$ for at least one $k \in \{1, \ldots, m\}$, (c) $y^T A = 0$, and (d) $0 > y^T b - \frac{e}{2} \cdot y^T a'$. Because of (b), we can transform the equality (c) into the following form:

$$a_k = -\textstyle\sum_{i=1, i\neq k}^{m} \left(\frac{y_i}{y_k} a_i\right)\ . \tag{4}$$

By multiplying (4) with an $x \in P_b^A$, we get: $a_k^T x = -\sum_{i=1, i\neq k}^{m} \left(\frac{y_i}{y_k} a_i^T x\right)$. Since $a_i^T x \leq b_i$ and $y_i \geq 0$, we get a finite lower bound for $a_k^T x$:

$$a_k^T x = -\textstyle\sum_{i=1, i\neq k}^{m} \left(\frac{y_i}{y_k} a_i^T x\right) \geq -\sum_{i=1, i\neq k}^{m} \left(\frac{y_i}{y_k} b_i\right)\ .$$

Benchmark Name	CAV-2009		DILLIG		PRIME-CONE		SLACKS		ROTATE	
#Instances	503		229		19		229		229	
Solvers:	**solved**	**time**	**solved**	**time**	**solved**	**time**	**solved**	**time**	**solved**	**time**
SPASS-IQ-0.1+uc	**503**	22	**229**	9	**19**	0.4	**229**	**26**	**229**	**9**
SPASS-IQ-0.1	**503**	713	**229**	218	**19**	0.4	197	95	**229**	214
ctrl-ergo	**503**	**12**	**229**	**5**	**19**	0.4	**229**	46	24	6760
cvc4-1.4	467	12903	206	4146	18	3	152	4061	208	6964
mathsat5-3.9	**503**	6409	225	2314	**19**	3.5	181	4577	**229**	1513
yices-2.4.2	472	11461	213	2563	**19**	**0.1**	147	5767	180	10171
z3-4.4.0	466	764	213	525	**19**	0.2	158	383	213	528

Fig. 7. Experimental Results

Thus, the upper bound $\sup\left\{a_k^T x \mid x \in P_b^A\right\} \leq b_k < \infty$ and the lower bound $\inf\left\{a_k^T x \mid x \in P_b^A\right\} \geq -\sum_{i=1, i\neq k}^{m}\left(\frac{y_i}{y_k} b_i\right) > -\infty$ are finite, which contradicts the assumption that $Ax \leq b$ has infinite lattice width. □

We have found instances of polyhedra with the infinite lattice width property in some classes of the SMT-LIB benchmarks. These instances are 229 of the 233 *dillig* benchmarks designed by Dillig et al. [7], 503 of the 591 *CAV-2009* benchmarks also by Dillig et al. [7], 229 of the 233 *slacks* benchmarks which are the dillig benchmarks extended with slack variables [14], and 19 of the 37 *prime-cone* benchmarks, that is, "a group of crafted benchmarks encoding a tight n-dimensional cone around the point whose coordinates are the first n prime numbers" [14]. The remaining problems (4 from dillig, 88 from CAV-2009, 4 from slacks, and 18 from prime-cone) do not fulfill the infinite lattice width property because they are either tightly bounded or unsatisfiable. For our experiments, we look only at the instances of those benchmark classes that actually fulfill the infinite lattice width property.

Using these benchmark instances, we have confirmed our theoretical assumptions (Lemma 5) in practice. We integrated the unit cube test into our own branch-and-bound solver *SPASS-IQ*[1] and ran it on the infinite lattice width instances; once with the unit cube test turned on (*SPASS-IQ-0.1+uc*) and once with the test turned off (*SPASS-IQ-0.1*). For every problem, SPASS-IQ-0.1+uc applies the unit cube test exactly once. This application happens before we start the branch-and-bound approach. We also compared our solver with some of the state-of-the-art SMT solvers currently available for linear integer arithmetic: *cvc4-1.4* [1], *mathsat5-3.9* [5], *yices2.4.2* [8], and *z3-4.4.0* [6]. As mentioned before, all these solvers employ a branch-and-bound approach with an underlying dual simplex solver [9].

The solvers had to solve each problem in under 10 minutes. For the experiments, we used a Debian Linux server with 32 Intel Xeon E5-4640 (2.4 GHz) processors and 512 GB RAM. Figure 7 lists the results of the different solvers (column one) on the different benchmark classes (row one). Row two lists the number of benchmark instances we considered for our experiments. For each

[1] http://www.spass-prover.org/spass-iq

combination of benchmark class and solver, we have listed the number of instances the solver could solve in the given time as well as the total time (in seconds) of the instances solved (columns labelled with "solved" and "time", respectively).

Our solver that employs the unit cube test solves all instances with the application of the unit cube test and is 25 times faster than our solver without the test. The SMT theory solvers in their standard setting were not able to solve all instances within the allotted time. Moreover, our unit cube test was over 100 times faster than any state-of-the-art SMT solver.

We also compared our test with the *ctrl-ergo* solver, which includes a subroutine that is essentially the dual to our largest cube test [2]. As expected, both approaches are comparable for infinite lattice width polyhedra. In order to also compare the two approaches on benchmarks without infinite lattice width, we created the *rotate* benchmarks by adding the same four inequalities to all infinite width instances of the dillig benchmarks. These four inequalities essentially describe a square bounding the variables x_0 and x_1 in an interval $[-u, u]$. For a large enough choice of u (e.g., $u = 2^{10}$), the square is so large that the benchmarks are still satisfiable and not absolutely trivial for branch-and-bound solvers. To add a challenge, we rotated the square by a small factor $1/r$, which resulted in the following four inequalities:

$$-b \cdot r \cdot r + r \leq b \cdot r \cdot x_0 - x_1 \leq b \cdot r \cdot r - r\,, \text{ and}$$
$$-b \cdot r \cdot r + r \leq x_0 + b \cdot r \cdot x_1 \leq b \cdot r \cdot r - r\,.$$

These changes have nearly no influence on SPASS-IQ, and two SMT solvers even benefit from the proposed changes. However, the *rotate* benchmarks are very hard for ctrl-ergo because its subroutine detects only infinite lattice width. Without infinite lattice width, ctrl-ergo starts its search from the boundaries of the polyhedron instead of looking at the polyhedron's interior. We can even control the number of iterations (r^2) ctrl-ergo spends on the parts of the boundary without any integer solutions if we choose r accordingly (e.g., $r = 2^{10}$). In contrast, we use our cube tests to also extract interior points for rounding. This difference makes our tests much more stable under consideration of small changes to the polyhedron.

There exist alternative methods for solving linear integer constraints that do not rely on a branch-and-bound approach [4,14]. These have not yet matured enough to be competitive with our tests or state-of-the-art SMT theory solvers.

Most problems in the linear integer arithmetic SMT-LIB benchmarks with finite lattice width can be solved without using any actual integer arithmetic technique+. A standard simplex solver for the reals typically finds a real solution for such a problem that is also an integer solution. Applying the unit cube test on these trivial problem classes is a waste of time, worst case it doubles the eventual solution time. For these examples it is beneficial to first compute a general real solution and to check it for integer satisfiability before applying the unit cube test. This has the additional benefit that real unsatisfiable problems are also filtered out before applying the unit cube test. Also, the unit cube test is almost guaranteed to fail on problems containing boolean variables, i.e., variables that are either 0 or 1, unless they are absolutely trivial and describe a

unit cube themselves. Whenever the problem contains a boolean variable, it is often beneficial to skip the unit cube test.

6 Further Cube Test Applications

Equalities are the greatest challenge for the applicability of our cube tests. A polyhedron contains an equality $a_E^T x = b_E$ if $a_E^T x = b_E$ holds for all $x \in P_b^A$. An equality contained in $Ax \leq b$ is *explicit* if $Ax \leq b$ includes the inequalities $a_E^T x \leq b_E$ and $-a_E^T x \leq -b_E$. Otherwise, the equality is *implicit*. Polyhedra containing equalities have only surface points and, therefore, neither an interior nor a center. Thus, a largest cube has edge length zero and is just a point in the original polyhedron. Similar problems occur if we allow not only inequalities but also other types of constraints, such as negated equalities ($a_i^T x \neq b_i$), divisibility constraints ($d \mid a_i^T x + b_i$, i.e., $d \in \mathbb{Z}$ divides $a_i^T x + b_i$), and negated divisibility constraints ($d \nmid a_i^T x + b_i$). In this section, we propose additional transformations and strategies that are useful for resolving the aforementioned challenges and are also applicable even beyond our tests.

First of all, we can transform any divisibility constraint and negated divisibility constraint into an equality by introducing additional variables. For divisibility constraints $d \mid a_i^T x + b_i$, this transformation is known as the diophantine representation: $\exists q \in \mathbb{Z}.\ dq - a_i^T x = b_i$. For negated divisibility constraints $d \nmid a_i^T x + b_i$, there exists a similar transformation: $\exists q \in \mathbb{Z}.\ \exists r \in \mathbb{Z}.\ dq + r - a_i^T x = b_i \ \wedge\ 1 \leq r \leq d-1$. Both of these transformations describe the formal definition of dividing $a_i^T x + b_i$ by d: $a_i^T x + b_i = dq + r$, where q is the quotient of the division and r the remainder. Since the divisibility constraint enforces that d divides $a_i^T x + b_i$, the remainder r must be zero. Likewise, the negated divisibility constraint enforces that d does not divide $a_i^T x + b_i$. Therefore, the remainder r lies between 1 and $d - 1$. These transformations are useful beyond our tests because they can be used to integrate (negated) divisibility constraints into the simplex algorithm. The only disadvantage is that we have to introduce additional variables q (and r) for every (negated) divisibility constraint.

Next, we eliminate all equalities from $Ax \leq b$. We do so by taking an equality $a_i^T x = b_i$ contained in $Ax \leq b$ and replacing a variable x_k in $Ax \leq b$ by substituting with $x_k := \frac{1}{a_{ik}}(b_i - \sum_{j=1, j\neq k}^{n} a_{ij} x_j)$, where $a_{ik} > 0$. Naturally, replacing x_k in $Ax \leq b$ creates a new system of inequalities $A'x' \leq b'$, where $A' \in \mathbb{Q}^{(m-1)\times(n-1)}$, $b' \in \mathbb{Q}^{(m-1)}$, and $x' = (x_1, \ldots, x_{k-1}, x_{k+1}, \ldots, x_n)^T$. We iteratively repeat this approach until our system of inequalities $A^I x^I \leq b^I$ contains no more equalities. As a by-product, we get a system of equalities $A^E x = b^E$ consisting of all equalities we have found. The two systems of constraints $A^I x^I \leq b^I$ and $A^E x = b^E$ together are equivalent to $Ax \leq b$, but $A^I x^I \leq b^I$ contains no equalities while $A^E x = b^E$ contains (at least implicitly) all equalities of $Ax \leq b$. We can now completely eliminate the equalities $A^E x = b^E$ from $Ax \leq b$ by combining this approach with a *diophantine equation handler* [12]. The result is a new system of inequalities that contains no equalities and has an integer solution if and only if $Ax \leq b$ has one.

Extracting equalities has further applications; for instance, the derivation of equalities is needed for the combination of theories by the *Nelson-Oppen method*. We can even check whether an arbitrary equality $a_E^T x = b_E$ is an equality of $Ax \leq b$ by transforming the equalities $A^E x = b^E$ into a substitution and applying this substitution to $a_E^T x = b_E$. The equality $a_E^T x = b_E$ is only contained in $Ax \leq b$ if $a_E^T x = b_E$ simplifies to $0 = 0$ after the substitution.

However, we are still missing one step in our elimination approach: how do we efficiently find an equality $a_i^T x = b_i$ contained in $Ax \leq b$ so that we can substitute with it? The answer are cubes, presented in the below lemma.

Lemma 6. *Let $Ax \leq b$ be a polyhedron. Then, exactly one of the following statements is true:*
(1) $Ax \leq b$ contains an equality $a_E^T x = b_E$ with $a_E \neq \mathbf{0}$, or
(2) $Ax \leq b$ contains a cube with edge length $e > 0$.

Proof. This proof is a case distinction over the sign of x_e for the following slightly simplified version of the largest cube test:

$$\begin{array}{ll} \text{maximize} & x_e \\ \text{subject to} & Ax + a' x_e \leq b, \text{ where } a'_i = \frac{1}{2} \left\| a_i \right\|_1 . \end{array} \tag{5}$$

If the maximum objective value is positive, $Ax \leq b$ contains a cube with edge length $e > 0$. Therefore, we have to prove that $Ax \leq b$ contains no equality $a_E^T x = b_E$ with $a_E \neq 0$, which we will do by contradiction. Assume $Ax \leq b$ contains an equality $a_E^T x = b_E$ with $a_E \neq 0$. Then, by transitivity of the subset relation, the polyhedron consisting of the inequalities $a_E^T x \leq b_E$ and $-a_E^T x \leq -b_E$ must also contain a cube of edge length e. However, applying the transformation from Corollary 3 to this new polyhedron results in two contradicting inequalities: $a_E^T x \leq b_E - \left\| a_E \right\|_1 \cdot \frac{e}{2}$ and $-a_E^T x \leq -b_E - \left\| a_E \right\|_1 \cdot \frac{e}{2}$. Thus, (1) and (2) cannot hold at the same time.

If the maximum objective value is zero, then $Ax \leq b$ is satisfiable but contains no cube with edge length $e > 0$. Therefore, we have to prove that $Ax \leq b$ contains an equality $a_E^T x = b_E$ with $a_E \neq 0$. Consider the dual linear program of (5):

$$\begin{array}{ll} \text{minimize} & y^T b \\ \text{subject to} & y^T A = 0 \,, \\ & y^T a' = 1 \,, \text{ where } a'_i = \frac{1}{2} \left\| a_i \right\|_1 \,, \\ & y \geq 0 \,. \end{array} \tag{6}$$

Due to strong duality, the objectives of the dual and primal linear programs are equal. Therefore, there exists a $y \in \mathbb{R}^m$ that has objective $y^T b = 0$ and that satisfies the dual (6). Since $y^T a' = 1$ and $a'_i \geq 0$ and $y_i \geq 0$ holds, there exists a $k \in \{1, \ldots, m\}$ such that $y_k > 0$. By multiplying $y^T A = 0$ with an $x \in P_b^A$ and isolating $a_k^T x$, we get: $a_k^T x = -\sum_{i=1, i \neq k}^m \left(\frac{y_i}{y_k} a_i^T x \right)$. Using $y_i \geq 0$, and our original inequalities $a_i^T x \leq b_i$, we get a finite lower bound for $a_k^T x$:

$$a_k^T x = -\sum_{i=1, i \neq k}^m \left(\tfrac{y_i}{y_k} a_i^T x \right) \geq -\sum_{i=1, i \neq k}^m \left(\tfrac{y_i}{y_k} b_i \right) .$$

Now, we reformulate $y^T b = 0$ analogously and get: $b_k = -\sum_{i=1, i \neq k}^m \left(\frac{y_i}{y_k} b_i \right)$. Thus, $a_k^T x = b_k$ is an equality contained in the original inequalities $Ax \leq b$.

If the maximum objective value is negative, $Ax \leq b$ is unsatisfiable and contains no cube with edge length $e > 0$. Since P_b^A is now empty, $Ax \leq b$ contains all equalities. □

By Lemma 6 a polyhedron contains a cube with a positive edge length $e > 0$, or an equality. Since e is arbitrarily small, the factor $\frac{e}{2} \|a_i\|_1$ is also arbitrarily small and $a_i^T x + \frac{e}{2} \|a_i\|_1 \leq b_i$ converges to $a_i^T x < b_i$. Therefore, $Ax \leq b$ contains an equality if and only if $Ax < b$ is unsatisfiable. We can solve this system of strict inequalities with the dual simplex algorithm by Dutertre and de Moura [9]. In case $Ax < b$ is unsatisfiable, the algorithm returns an explanation, i.e., a minimal set C of unsatisfiable constraints $a_i^T x < b_i$ from $Ax < b$. If $Ax \leq b$ itself was satisfiable, then we can extract equalities from this explanation: every $a_i^T x < b_i \in C$ implies that $Ax \leq b$ contains the equality $a_i^T x = b_i$.

Finally, we have two ways of handling negated equalities $a_i^T x \neq b_i$. Either we split our set of constraints into two sets of constraints, replacing $a_i^T x \neq b_i$ in the first one with $a_i^T x \leq b_i - 1$ and in the second one with $-a_i^T x \leq -b_i - 1$; or, we ignore all negated equalities during the calculation of the tests themselves and use the negated equalities only to verify the integer solutions returned by the tests.

7 Conclusion

We have presented two tests based on cubes: the largest cube test and the unit cube test. Our tests can be integrated into SMT theory solvers without sacrificing the advantages SMT solvers gain from the incremental structure of subsequent subproblems. Furthermore, our experiments have shown that these tests increase efficiency on certain polyhedra such that previously hard sets of constraints become trivial. We have even shown that major obstacles to our tests, for example equalities, can be handled through generally useful preprocessing steps. Moreover, these preprocessing steps led to an additional application for our tests: finding equalities.

Our future research will investigate further applications of our tests. We expect that we can use cubes not only for the detection of equalities, but also for the detection of (un)bounded directions. We can likely use the largest cube test as a selection strategy for branching by always choosing the branch containing the largest cube. This is in all likelihood a beneficial strategy since the largest cube is a good heuristic for the branch with the most space for integer solutions.

References

1. C. Barrett, C. Conway, M. Deters, L. Hadarean, D. Jovanović, T. King, A. Reynolds, and C. Tinelli. CVC4. In G. Gopalakrishnan and S. Qadeer, editors, *CAV*, volume 6806 of *LNCS*, pages 171–177. Springer, 2011.
2. F. Bobot, S. Conchon, E. Contejean, M. Iguernelala, A. Mahboubi, A. Mebsout, and G. Melquiond. A simplex-based extension of fourier-motzkin for solving linear

integer arithmetic. In *IJCAR 2012*, volume 7364 of *LNCS*, pages 67–81. Springer, 2012.
3. S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.
4. M. Bromberger, T. Sturm, and C. Weidenbach. Linear integer arithmetic revisited. In A. P. Felty and A. Middeldorp, editors, *CADE-25*, volume 9195 of *LNCS*, pages 623–637. Springer, 2015.
5. A. Cimatti, A. Griggio, B. Schaafsma, and R. Sebastiani. The MathSAT5 SMT Solver. In N. Piterman and S. Smolka, editors, *Proceedings of TACAS*, volume 7795 of *LNCS*. Springer, 2013.
6. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In C. Ramakrishnan and J. Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
7. I. Dillig, T. Dillig, and A. Aiken. Cuts from proofs: A complete and practical technique for solving linear inequalities over integers. In A. Bouajjani and O. Maler, editors, *CAV*, volume 5643 of *LNCS*, pages 233–247. Springer, 2009.
8. B. Dutertre. Yices 2.2. In A. Biere and R. Bloem, editors, *Computer-Aided Verification (CAV'2014)*, volume 8559 of *LNCS*. Springer, 2014.
9. B. Dutertre and L. de Moura. A fast linear-arithmetic solver for dpll(t). In T. Ball and R. B. Jones, editors, *CAV*, volume 4144 of *LNCS*, pages 81–94. Springer, 2006.
10. M. Ehrgott. Scalarization techniques. In *Multicriteria Optimization*, pages 97–126. Springer Berlin Heidelberg, 2005.
11. G. Faure, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Sat modulo the theory of linear arithmetic: Exact, inexact and commercial solvers. In H. Kleine Büning and X. Zhao, editors, *SAT 2008*, volume 4996 of *LNCS*, pages 77–90. Springer, 2008.
12. A. Griggio. A practical approach to satisfiability modulo linear integer arithmetic. *JSAT*, 8(1/2):1–27, 2012.
13. F. S. Hillier. Efficient heuristic procedures for integer linear programming with an interior. *Operations Research*, 17(4):600–637, 1969.
14. D. Jovanović and L. de Moura. Cutting to the chase. *JAR*, 51(1):79–108, 2013.
15. M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, and L. A. Wolsey, editors. *50 Years of Integer Programming 1958-2008*. Springer, 2010.
16. R. Kannan and L. Lovász. Covering minima and lattice point free convex bodies. In K. Nori, editor, *FSTTCS*, volume 241 of *LNCS*, pages 193–213. Springer, 1986.
17. N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–396, 1984.
18. C. H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(4):765–768, Oct. 1981.