Some Efficient Algorithms for the Tightest UTVPI Polyhedral Over-Approximation Problem

Abhishek A. Patwardhan
Scalable Compilers for Heterogeneous Architectures Lab
Department of Computer Science & Engineering
Indian Institute of Technology Hyderabad, India
cs15mtech11015@iith.ac.in

Ramakrishna Upadrasta
Scalable Compilers for Heterogeneous Architectures Lab
Department of Computer Science & Engineering
Indian Institute of Technology Hyderabad, India
ramakrishna@iith.ac.in

1 Introduction and Motivation
Precise program analysis plays an important role in verifying safety properties of a given input program as well as forming a foundational basis in modern optimizing compilers. However, in order to accurately analyze a given input program, it must be represented in a compact and mathematically analyzable representation.

There exist many mathematical representations that can be handy to statically model the dynamic behavior of the input program. Convex polyhedra are one of the most powerful abstractions that allow effective and exact representation of programs where relationships among program variables are affine. Since they are based on the extremely well understood techniques of linear (rational) and integer linear programming [Sch86], convex polyhedra are also well backed by libraries like PIP [Fea88] and ISL [Ver10].

The polyhedral compilation framework based on (rational and integer) convex polyhedra is well established as a formal and a highly powerful means to automatically parallelize Affine Control Loops (ACLs) of the input program [Fea92, DRV00, BHRS08]. Even in automatic program analysis, like abstract interpretation, convex polyhedra have been widely used for verification of program properties [CC77, CH78].

While having the advantage of precision, the usage of (rational and integer) convex polyhedra faces the scalability limitation, when solving the static analysis problems for millions of lines of code, or unrealistic compile-time when finding schedules for large ACLs. One fundamental issue behind this limitation is the worst-case high-complexity or exponential-time algorithms for basic operations such as solving Feasibility, Optimization, Fourier-Motzkin and Vertex enumeration.

Over the years, many approaches have been proposed to overcome this limitation by using approximations of polyhedra as numerical abstract domains. These special abstract domains have better complexity, or have worst-case-polynomial-time algorithms for solving feasibility and optimization problems. Some examples are Intervals [CC77], Unit-Two-Variables-Per-Inequality (UTVPI) or Octagons [Min06], and Two-Variables-Per-Inequality (TVPI) [SK10]. Thanks to their better complexities, and aided by their closure properties, these abstract domains have been shown to be effective to scale the abstract...
interpretation (and verification) problem(s) for millions of lines of code [CCF+09], like in the Astére analyzer.

Of all the above, Octagons have been extremely successful [CCF+09]—having been found to have a wide-spread usage because being a relational domain, having a unique (and simple) graph-based representation, and supported by libraries like Apron [BCC+03].

The OA algorithm should satisfy the following properties:

- Return the tightest possible OA of a given convex polyhedron (if it exists).
- Should be efficient and easy to implement.

Our paper contributes to both the above properties in the context of the Octagon abstract domain. More specifically, we design two new OA algorithms to find the tightest (rational) Octagon (UTVPI) OA from a given arbitrary convex polyhedron without enumerating its generators.

Our algorithms rely on the following key and simple insights on Octagon polyhedra:

1. Octagons have just quadratic (and hence non-exponential) number of Octagonal directions, and, they have a small number of 1-dimensional faces (edges): namely, \(O(n^2)\) for an \(n\)-dimensional Octagon (refer Figure 1).

   In 2-dimensions, the Octagonal directions are the orthogonal (canonical) directions \(i, j\), and their \(\pm 45\degree\) rotations \((\pm i, \pm j)\) and \((\pm i, -j)\).

2. Given a general polyhedron \(P\), its width/ fatness polyhedron \(W_i\) in a particular Octagonal direction \(v\) is (nothing but!) an interval polyhedron with \(LB\) and \(UB\) being the bounds (lower-bound/min/inf and upper-bound/max/sup respectively); The scalar value \(UB-LB\) constitutes the fatness of \(P\) in the particular Octagonal direction.

3. If \(P\) is a polyhedron and \(P'=OA_{UTVPI}(P)\) is the tightest Octagonal OA of \(P\), then, \(P'\) can be described as the intersection of the width/fatness polyhedra in each of the Octagonal directions.

   In 2-dimensions, there are just four width polyhedra as shown in Figure 1: \(W_i, W_j, W_{i+j}, W_{i-j}\):

   \[P' = W_i \cap W_j \cap W_{i+j} \cap W_{i-j} = OA_{UTVPI}(P)\]

4. Finding the tightest octagonal OA reduces to the problem of minimization of fatness in each of the Octagonal directions.

   Building from the above intuitions, our two algorithms rely on using different techniques—Linear Programming (LP) and Fourier-Motzkin (FM)—for the above mentioned fatness minimization (#4 above) in each of the Octagonal directions.

   More specifically,

   - [Algorithm#1] We propose a Linear Programming based algorithm for fatness minimization in each of the Octagonal directions, with a simple cost function encoding the same. The LP formulation is a result of a particular application of the Farkas’ lemma.
   - [Algorithm#2] We propose a Fourier-Motzkin elimination for fatness estimation along each of the Octagonal directions; in the orthogonal (canonical) directions, it is a regular application of FM, while in the (oblique) Octagonal directions, it is an application of FM after rotating the polyhedron (or the axes) by \(\pm 45\degree\).

**Contributions:** We make the following contributions:

- We present two new algorithms that Over-Approximate an arbitrary convex polyhedron into a (rational) Octagon (formally termed as Unit-Two-Variable-Per-Inequality) polyhedra by relying on elementary polyhedral operations and solely on its Hyperplane representation. Our algorithms improve over the OA algorithm that is implemented in the state-of-the-art libraries like Apron [JM09].
- Our first algorithm is based on Linear Programming; it relies on an application of the affine form of Farkas’ lemma and an objective function that encodes the tightness measure. Our second algorithm is based on Fourier-Motzkin (projections) and affine rotations.
- We do a thorough complexity analysis of our two algorithms along with the one proposed earlier by
Miné [Min06, Min04]. Our algorithms are implemented in (version 0.18) of the ISL library [Ver10, UTVI]. We briefly enumerate applications of our algorithms from program analysis and polyhedral compilation.

This paper is organized as follows: In Section 2, we discuss the necessary background. In Section 3, we introduce our Algorithm#1 which is based on a series of linear programming calls. In Section 4, we discuss our Algorithm#2, which is based on a series of Fourier-Motzkin eliminations and rotations. In Section 5, we do a thorough complexity analysis of both the algorithms along with discussing various perspectives. In Section 6, we discuss some related work. In Section 7, we discuss some applications in polyhedral compilation as well as static analysis. In Section 8, we discuss our implementation details in the ISL library and finally, we discuss some conclusions as well as future work.

2 Background

In this section, we formally describe some basic terminologies and background concepts. First we begin with some basic terminologies, then give a summary of the UTVPI-Over-Approximation algorithm by Antoine Miné. Finally, we give an overview of the setting of our algorithm.

2.1 Some basic terminology

**Polyhedron:** A Polyhedron is a space enclosed within an n-dimensional vector-space which is bounded by finitely many linear inequalities. Each linear inequality defines a half-space; hence polyhedra can be thought of as intersection of finitely many half-spaces.

**Dual representations of Polyhedra:** A Polyhedron in n-dimensional space can be represented and described in two alternative (dual) forms, and these representations are considered to be equivalent to each other.

- Hyperplane (H) representation: A Polyhedron is expressed as an intersection of finitely many affine inequalities \( P = \{ x | Ax \leq b \} \).
- Generator (V) representation: A Polyhedron is expressed as a convex combination of its extremal vertices, a conical combination of its rays and a linear combination of its lines.

\[ P = \{ x | x = V a + R b + L c; a_i \geq 0; \sum_j^a a_i = 1; b_i \geq 0 \}, \]

where V’s columns denote the vertices, R’s columns its rays and L’s columns its lines.

The classic algorithm by Chernikova [Che65] can be used to convert either of these representations to the other. Among others, the PolyLib [Ver92, Wil93], Irs [AF92], PPL [BHZ08], and ISL [Ver10, GGS+17] libraries have an implementation of this algorithm.

**Interval (Box) Polyhedra:** An Interval (or Box) polyhedron is a special case of convex polyhedra, where every constraint is restricted to the form: \( x_i \leq c_i \).

**UTVPI (Octagon) and DBM Polyhedra:** A UTVPI (Octagon) polyhedron is a special case of polyhedra where every affine constraint is restricted to the form: \( \pm x_i \leq c_i \pm x_j \leq c_j \). As the name suggests, every constraint should involve at most two variables and have coefficients to be one of the following \( \{ +1, 0, -1 \} \). A Difference Bound Matrix (DBM) polyhedron is a special case of UTVPI polyhedron where every constraint can only be of the form: \( \pm x_i - x_j \leq c_i \) or \( \pm x_i \leq c_i \); the coefficients are of opposite signs, or one of them is zero. A DBM can be represented in a compact matrixial (and graph) representation.

**Monotonizing Transformation:** Miné [Min06] proposed a monotonizing \( P_{\text{UTVPI}} \rightarrow P_{\text{DBM}} \) conversion that takes an input UTVPI polyhedron \( P_{\text{UTVPI}} \) and returns an equivalent DBM polyhedron \( P_{\text{DBM}} \). When \( P_{\text{UTVPI}} \) is a rational polyhedron (\( P_{\text{UTVPI}} \in \mathbb{Q} \)), the conversion is exact.

This conversion is the key step for solving the tightening and closure properties on the input UTVPI polyhedra using graph algorithms like Bellman-Ford and Floyd-Warshall [CSRL01].

**Fatness of a polyhedron:** The Fatness of a polyhedron \( P \) is the difference between the upper and lower bounds of its projection in a particular direction.

2.2 Some lemmas

**Affine form of Farkas’ lemma:** Let \( D \) be a nonempty polyhedron defined by \( p \) inequalities \( a_k x + b_k \geq 0 \), for any \( k \in \{1, \ldots, p\} \). An affine form \( \Phi \) is non-negative over \( D \) if and only if it is a non-negative affine combination of the affine forms used to define \( D \), meaning:

\[ \Phi(x) \equiv \lambda_0 + \sum_{k=1}^{p} \lambda_k (a_k x + b_k); \forall k \in [0, p] \lambda_k \geq 0 \]

The nonnegative values \( \lambda_k \) are called Farkas’s multipliers. Many seminal results in polyhedral compilation rely on the usage of the above powerful lemma [Sch86].

**Rotation operation of Polyhedra in \( H \)-form:** Given a Polyhedron \( P \) in \( H \)-form, \( P = \{ x | Ax \leq b \} \), where \( x = (x_1, x_2, \ldots, x_n) \), we define the rotation operation within a plane \( \{ x_i, x_j \} \) by 45°, \( \text{ROT}_{45}(P) \), as follows:

\[ f : (x_1, \ldots, x_i, \ldots, x_j, \ldots, x_n) \longrightarrow (x_1, \ldots, x_j + x_j, \ldots, x_i - x_j, \ldots, x_n) \]

\[ \text{ROT}_{45}(P) : P_{\text{rotated}} = \text{Image}(P, f) \]

There is however, a loss of precision [Min06] when \( P_{\text{UTVPI}} \in \mathbb{Z} \) is an integer polyhedron (also called as a Z-Polyhedron), defined over a collection of integer points bounded by affine faces. For instance, in 3-d, rotation about z axis fixes the plane of rotation to xy.
Decomposition of an Octagon into (two) interval polyhedra: Geometrically, a 2-d Octagon can be visualized as a superimposition of two interval (or box) polyhedra (see Figure 1). Each interval polyhedra captures fatness along two unique Octagonal directions. But, as one pair of Octagonal directions aligns the other pair by 45°, the two interval polyhedra must also be aligned with respect to each other by 45°.

If \( P \) is an Octagon in the \((x, y)\)-axes, it can be expressed as an intersection of two interval polyhedra \( P_1 \) and \( P_2 \):

\[
P = P_1 \cap P_2
\]

\[
P_1 = \{LB_x \leq x \leq UB_x \land LB_y \leq y \leq UB_y\}
\]

\[
P_2 = \{LB_{x+y} \leq x + y \leq UB_{x+y} \land LB_{x-y} \leq x - y \leq UB_{x-y}\}
\]

2.3 Miné’s Over-Approximation Algorithm

While proposing the Octagon abstract domain, Miné [Min06, Min04] presented an algorithm that takes a non-empty convex polyhedron \( P \) and returns its UTVPI-OA (\( \text{OA}_{\text{UTVPI}} (P) \)). Miné’s algorithm uses the generator (“Frame”) representation (\( V \)-form) of the polyhedron to be over-approximated. Therefore, it requires application of Chernikova’s algorithm [Che65, Ver92, AF92] to convert \( H \)-form to \( V \)-form.

The intuition behind Miné’s algorithm is to construct the smallest possible octagon which encloses all the vertices from \( P_V \), along with adjusting the upper/ lower bounds to \( \pm \infty \) for the Octagonal constraints which are along the direction of rays and lines.

Formally, the algorithm computes Octagonal union over all the vertices of the polyhedron, i.e., \( \bigcup_{OCT} v_i \), where vertex \( v_i \in P_V \). The subtle property that is being exploited here is that each vertex is trivially an octagon in itself. The final step of the algorithm involves post-processing the DBM resulting from \( \bigcup_{OCT} v_i \) to accommodate for rays and lines. For each ray/line, if the coordinate value for dimension \( x_j \) is non-zero, then the algorithm sets the upper/ lower bound for all the Octagonal constraints involving the variable \( x_j \) to \( \pm \infty \).

Limitation It is well known that enumerating generators of a polyhedron is never a polynomial time process: the simplest case of an \( n \)-dimensional hypercube can be represented with \( 2n \) halfspaces while it has \( 2^n \) many vertices. Hence the interest in a scalable algorithm which can directly build the OA using just the \( H \)-form of the original polyhedron.

2.4 OA Algorithms: the necessity

Convex polyhedra and its limitation A typical usage of an abstract domain—from static analysis [BAG14], polyhedral compilation [Upa13], or performance analysis [BKPS17]—involves collection of a large number of (rational or integer) small-size convex polyhedra\(^3\) that describe the relations between the variables in the input program.\(^4\) This is followed by performing several operations—like Union, Intersection, Emptiness, Optimization, Counting etc.—on these abstract domains which lead to scalability issues.

Our proposal for scalable UTVPI-OA Our premise of using an improved complexity UTVPI-OA rests on the strength of the Octagonal domain (mainly its cubic time-complexity for all its abstract domain operations). It is in the same theme as proposed by Miné’s classic work [Min06, Min04]. But, we propose that many of abstract domain operations, though involving small polyhedra, could cumulatively induce a large unscalability factor. This could either be because of their large number, or the high complexity of the operations.

In this paper, we deal with only (rational) approximations. We do not deal with integer linear approximations and parametric (rational or integer) linear programming approximations. Though these latter problems are harder, we believe that our work will enable the latter.

3 Algorithm#1: LP based OA Algorithm

In his seminal work on affine scheduling, Feautrier proposed [Fea92] to use the affine form of Farkas’ lemma as a means of avoiding the transformation of a polyhedron from \( H \)-form to \( V \)-form.

In this section, we present a new algorithm that relies on the same lemma to construct a search space for valid UTVPI-OA hyperplanes. The algorithm involves a series of linear programming calls (4 of them in total), each of which finds the tightest over-approximating UTVPI hyperplane-pairs which are geometrically opposite to each other. Furthermore, to ensure tightness of the over-approximating hyperplanes, we minimize a cost function which encodes a tightness measure.

3.1 Enabling application of Farkas’ lemma

A generalized UTVPI constraint template looks like \( a_i x_i + b_j x_j \leq c_{ij} \) where \( a_i \) and \( b_j \) can be from \( \{0, 1, -1\} \). For the sake of presentation of the algorithm, we just consider the cases where \( a_i, b_j = \pm 1 \). The cases when \( a_i, b_j = 0 \) are obvious extensions, and will be covered in the formal algorithm.

Consider a UTVPI constraint template of the form: \( \pm x_j \leq c_{ij} \). We are interested in finding a good numeric value for \( c_{ij} \) such that the resulting constraint is satisfied by every point belonging to the original polyhedron \( P \). Geometrically, \( P \) must lie on one side of such a hyperplane so that the latter can help define the over-approximating UTVPI polyhedron. The above criterion is equivalent to the rule that the affine form \( \{H : c_{ij} \pm x_j \geq x_j \pm 0\} \) is positive over \( P \).

\(^3\)Small in either the number of dimensions, or constraints, or both.

\(^4\)This requirement is fundamentally different from our prior work [UC13], which proposed approximated domains for affine scheduling. In that work, the problem involved small number of large-size polyhedra with a limited type of operations: optimization and feasibility.
The following pre-conditions hold true which allow application of the affine form of Farkas’ lemma: (1) \( H \) is affine. (2) \( H \) must be positive over the original convex polyhedron. (3) \( P \) must be non-empty.

While (1) is trivially satisfied, (3) is a basic assumption for program analysis; as part of program analysis or classic array data-flow analysis, the non-emptiness of polyhedra can easily be tested by a single LP call. For (2), we apply affine form of Farkas’ lemma.

\[
\lambda_{ij} \pm x_i \pm x_j \equiv \lambda_0 + \sum_{k=1}^{p} \lambda_k (a_k x + b_k); \quad \forall k \in [0, p] \lambda_k \geq 0
\]

Where each \( \lambda_k \) is a Farkas’ multiplier.

Equating the coefficients of the variables from both the sides, and projecting out the Farkas’ multipliers, results in a search space that captures all feasible values for \( \lambda_{ij} \) and this in turn corresponds to a search space of over-approximating UTVPI hyperplanes.

3.2 Challenge in searching for tightest over-approximating UTVPI hyperplanes

Application of Farkas’ lemma enables characterization of the search space for over-approximating UTVPI hyperplanes. There is however a need to have a selection criterion to choose the hyperplanes so that the UTVPI-OA is the tightest one. From the above discussion, it is clear that searching for the right constant \( \lambda_{ij} \) will lead to the tightest UTVPI-OA.

The selection criterion will however change depending on the nature of the UTVPI-template constraint, whether it is bounding from the lower or the upper side. For the lower-bounding (inf) constraint, it is desirable to select the hyperplane having the maximum possible numeric value; while it is desirable to select the hyperplane having the minimal possible numeric value for the upper-bounding (sup) constraint. Both these selection criteria together result in the tightest over-approximating (lower/upper bounding) UTVPI hyperplane-pairs in that particular Octagonal direction. This can be repeated for each of the \( \binom{n}{2} \) Octagonal directions.

3.3 Joint search space construction and cost function minimization

To apply both these selection criteria together, a joint search space for the two geometrically opposite over-approximating UTVPI constraints can be constructed. A 2-d joint search space \((c_{ij}, c'_{ij})\) can be formed by application of Affine form of Farkas’ lemma to \( x_i + x_j \leq c_{ij} \) and \( -x_i - x_j \leq -c'_{ij} \).

Consider a linear cost function \( f(i, j) = c_{ij} - c'_{ij} \). It represents the distance among two geometrically opposite over-approximating hyperplanes. Minimizing \( f \) with respect to the search space \((c_{ij}, c'_{ij})\), results in tightening of the OA along that Octagonal direction. So, the following objective function can be used while solving a (rational) linear programming problem over the joint search space:

\[
\text{lexmin} \left( c_{ij} - c'_{ij}, c_{ij}, c'_{ij} \right)
\]

That is, to find the tightest OA, minimize \( c_{ij} - c'_{ij} \) with the highest priority. In case of the existence of two possible sets of values for \((c_{ij}, c'_{ij})\) with equal separating distance among them, the one whose coefficients have a lower numeric value is preferred.

Iteratively finding the UTVPI hyperplanes pairs Optimizing the cost function over the search space gives two geometrically opposite UTVPI hyperplanes. For a \( n \)-dimensional (bounded) polyhedra, it is necessary to find \( \binom{n}{2} \) UTVPI over-approximating hyperplanes. However, due to the construction of a joint search space, the above procedure needs to be iterated only \( 4 \binom{n}{2} \) times, once in each Octagonal direction. The intersection of all the constraints found gives the UTVPI OA of the original polyhedron.

Handling of unbounded polyhedra While obtaining a joint search space, the individual search spaces of the two over-approximating UTVPI hyperplanes need to be intersected. If the polyhedron is unbounded along a given direction, then, the over-approximating UTVPI search space turns out to be empty, in turn making the resulting intersection empty. So, before constructing the joint search space, there is a need to ensure non-emptiness of each individual search space. Additionally, the cost function needs to be specialized in order to compensate for the empty search space.

Our complete algorithm is shown in Algorithm 1.

3.4 Relation between cost function minimization and Fatness of the convex polyhedron

Figure 2. Fatness of polyhedron serves as threshold while minimizing the width between over-approximating hyperplanes.

We now investigate the relation between tightness of UTVPI over-approximation and the fatness of the original...
Algorithm 1 LP based UTVPI OA Algorithm

Require: \( P \) ← Polyhedron in \((x_1, x_2, ..., x_n)\) dimensions
1. \( O \) ← Universal Set in n dimensions
2. for each dimension \( x_j \) do
3. for each dimension \( x_j \neq x_i \) do
4. for each \( c_i \in \{-1,0\} \) do
5. for each \( c_j \in \{-1,0,1\} \) do
6. if \((c_i=0 \text{ and } c_j=0)\) or \((c_i=0 \text{ and } c_j=1)\) continue end if
7. \( S_1 \) ← Affine form of Farkas lemma \((P, c_{ij} + c_i x_i + c_j x_j \geq 0)\)
8. \( c'_i \) ← \(-c_i\)
9. \( c'_j \) ← \(c_j\)
10. \( S_2 \) ← Affine form of Farkas lemma \((P, c'_{ij} + c'_i x_i + c'_j x_j \geq 0)\)
11. if (not (Empty\((S_1)\) or Empty\((S_2)\))) then
12. \( S \) ← product space \((S_1, S_2)\)
13. Solve \text{lexmin} \((c_{ij} - c'_{ij}, c_{ij}, c'_{ij})\) over \( S \)
14. \( O \) ← \( O \cap (c_{ij} + c_i x_i + c_j x_j \geq 0) \cap (c'_{ij} + c'_i x_i + c'_j x_j \geq 0) \)
15. else
16. if Empty\((S_2)\) then
17. Solve \text{lexmin} \((c_{ij})\) over \( S_1 \)
18. \( O \) ← \( O \cap (c_{ij} + c_i x_i + c_j x_j \geq 0) \)
19. else
20. Solve \text{lexmin} \((c'_{ij})\) over \( S_2 \)
21. \( O \) ← \( O \cap (c'_{ij} + c'_i x_i + c'_j x_j \geq 0) \)
22. end if
23. end if
24. end for
25. end for
26. end for
27. Return \( O \)

In the LP based OA algorithm, the cost function \( c_{ij} - c'_{ij} \) is being minimized to find the tightest over-approximating UTVPI hyperplanes. However, this function can be minimized only till a threshold value, which essentially corresponds to the fatness of \( P \) along that (Octagonal) direction. Figure 2 illustrates this. The cost function can be seen as minimizing the fatness of the (over-approximating) UTVPI polyhedron thereby making it as close as possible to \( P \). This ensures that the tightness of the over-approximation in that Octagonal direction. When the minimization of fatness is done for each of the 4 \( \binom{4}{2} \) Octagonal directions, it will ensure that the resulting Octagonal OA is the tightest one.

In the next section, we develop another algorithm which is (1) free of linear programming, and (2) uses the relationship between fatness of a convex polyhedron and tightness of its UTVPI-OA. The idea behind our second algorithm lies in measuring the fatness of the original convex polyhedron along all possible Octagonal directions by using the Fourier-Motzkin projection algorithm along with affine rotations.

4 Algorithm#2: FM based OA Algorithm

The Fourier-Motzkin (FM) algorithm [DE73] has been well known in the geometry folklore with its main purpose being a means of eliminating variables from a set of linear constraints using projection operations.

In this section, however, we present a FM based UTVPI-OA algorithm. We provide a unique use-case of FM for measuring fatness of a convex polyhedron along various Octagonal directions so as to compute tightest UTVPI-OA.

The central idea behind our FM based algorithm is to iteratively build the Octagonal OA for a given convex polyhedron by individually constructing \( \binom{4}{2} \) many 2-dimensional Octagonal OAs. Each of the OAs are constructed in such a way that they over-approximate the (exact) shadow of the original polyhedron along the eight Octagonal directions.

The 2-dimensional shadow of the polyhedron can be obtained by projecting out all except the required two dimensions. The projections on the oblique (purely) Octagonal directions can be obtained by rotating the polyhedron itself (or equivalently, the canonical axes) by \( \pm 45° \).
Algorithm 2 FM based UTVPI OA Algorithm

Require: $P \leftarrow$ Polyhedron in $(x_1, x_2, \ldots, x_n)$ dimensions
1. $O \leftarrow$ Universal Set in $n$ dimensions
2. for each dimension $x_j$ do
3. for each dimension $x_j \neq x_i$ do
4. $S \leftarrow$ Project_Out_Except($P, x_i, x_j$)
5. $W_i \leftarrow$ Project_Out($S, x_i$)
6. $W_j \leftarrow$ Project_Out($S, x_j$)
7. $R \leftarrow \{(x_i, x_j) \rightarrow (d_1, d_2): (d_1 = x_i + x_j \land d_2 = x_i - x_j)\}$
8. $S' \leftarrow$ Image($S, R$)
9. $W'_{i+j} \leftarrow$ Project_Out($S', d_1$)
10. $W'_{i-j} \leftarrow$ Project_Out($S', d_2$)
11. $O \leftarrow O \cap W_i \cap W_j \cap W'_{i+j} \cap W'_{i-j}$
12. end for
13. end for
14. Return $O$

In the upcoming sub-sections, we discuss the FM based algorithm in detail.

4.1 Fatness estimation along Orthogonal (Canonical) directions

Given a convex polyhedron $P$, let $S$ be its (exact rational) shadow on a 2-d plane with dimensions $x_i$ and $x_j$.

$$S = \text{Project}_{-}\text{Out}_{-}\text{Except}(P, x_i, x_j)$$

Referring to the lemma on Octagonal decomposition (Section 2.2), the task of finding the UTVPI-OA of $S$ can be decomposed into finding two separate interval (or box) polyhedra having the desired orientations. For this, we first discuss the method to obtain the interval (box) polyhedral OA of the shadow $S$ along canonical (Orthogonal) axes. We then discuss how to extend this method to obtain another interval polyhedron having the desired orientation which again over-approximates $S$.

Finding the interval-OA of a polyhedron is trivial by using linear programming. It can be solved using two LP problems per dimension with min and max as objective functions. But here, instead of using LP, we use FM in a recursive manner. (Remember that the Fourier-Motzkin has already been used to obtain $S$, the 2-d shadow of $P$). To obtain interval-OA for $S$, it can simply be projected on both of its axes (namely, the $x_i$ and $x_j$ axes, along the $(0^\circ, 90^\circ)$ directions). This 2-d to 1-d projections essentially result in estimating the fatness of $S$ along both the dimensions. The constraints obtained after projecting $S$ on each individual axes provide the (lower/upper) bounds for that particular dimension. In this way, a 2-d rectangular bounding box can be constructed for the shadow $S$. This 2-d interval (box) polyhedron gives 4 constraints for the desired $n$-dimensional over-approximating UTVPI polyhedron.

4.2 Fatness estimation along (oblique) Octagonal directions

The remaining 4 constraints of the 2-d Octagonal OA need to specify bounds on $x_i + x_j$ and $x_i - x_j$ ($\pm 45^\circ$) directions. In other words, the fatness of the shadow $S$ along the two directions $d_1$ and $d_2$ needs to be estimated, where $d_1 : x_i + x_j$ and $d_2 : x_i - x_j$.

**Naïve method** One naïve method is to rotate $P$ itself along each of the $\pm 45^\circ$ oblique Octagonal directions, and then do the FM projections. The projections obtained along these oblique Octagonal directions result in the estimation of width or fatness in these directions. The above method has the limitation that (in total), the FM algorithm needs to be applied $2 \binom{n}{2}$ times on polyhedron $P$; once for the $(0^\circ, 90^\circ)$ directional pair, and again for the $\pm 45^\circ$ directional pair of axes. This could be expensive and could be improved.

**Improved method** We propose that $S$, the exact shadow of $P$ that has been obtained as a result of FM, can itself be rotated to obtain the UTVPI-OA. The rotated shadow can be projected on the orthogonal axes to obtain its fatness in the oblique Octagonal directions. These widths can be used to define the tightest over-approximating constraints in the $\pm 45^\circ$ directions.

Consider the following linear transformation:

$$R : \{(x_i, x_j) \rightarrow (d_1, d_2) | (d_1 = x_i + x_j \land d_2 = x_i - x_j)\}$$

The above linear transformation rotates the canonical axes of a shadow by $45^\circ$. Now, the application of Change of Basis operation on the shadow $S$ with respect to the mapping function $R$ will obtain the rotated shadow. Formally, $S_{\text{rotated}} = \text{Image}(S, R)$

The rotated shadow ($S_{\text{rotated}}$) can be projected onto (new) canonical axes ($d_1$ and $d_2$) to estimate its fatness, and also to obtain (lower and upper) bounds along the two dimensions.
(d₁ : xᵢ₊₁ + xⱼ, d₂ : xᵢ − xⱼ)

This operation results in finding the rectangular bounding box in a rotated space. This gives rise to 4 more required constraints for n-dimensional UTVPI-OA. By intersecting the constraints corresponding to the (1) bounding box of the original shadow, and (2) bounding box of the rotated shadow, the tightest 2-d Octagonal (UTVPI) OA for a shadow is obtained.

4.3 Iteratively constructing UTVPI OA

The above described procedure needs to be iterated for each of the 2-d planes from the n-dimensional vector space in which original polyhedron resides. So, for all possible pairs of axes, it needs to be iterated \( \binom{n}{2} \) times.\(^6\)

All the Octagonal OA hyperplanes so obtained can be intersected to obtain the UTVPI OA of the original convex polyhedron. It can be noticed that unbounded polyhedra are handled implicitly, thanks to the resilience of Fourier-Motzkin projection algorithm.

5 Complexity Analysis and Discussion

In this section, we first do a time complexity analysis of our algorithms discussed in Sections 3 and 4. Then, we do a comparative analysis of these two algorithms along with Mini’s algorithm and discuss some perspectives.

Let us assume that the input polyhedron \( P = \{ x \mid Ax \leq b \} \) be a \( m \times n \) constraint system (\( m \) constraints over \( n \) variables), and \( L \) is the maximum size of the numbers (or coefficients) occurring in the input.

5.1 LP based algorithm

In the LP based algorithm, it can easily be seen that in total, \( 4 \binom{n}{2} \) (rational) LP calls are made. It is well established that while there exist asymptotically better algorithms [GLS93], the simplex algorithm is the more widely used one for solving (rational) LP [Sch86]. Also, in the combinatorial optimization community, it is understood that for a normal (well-behaved) constraint system, the complexity of the (rational) LP problem using simplex algorithm is on average\(^7\)

\[ Z(m, n) \approx O((m + n)mnL). \]

So, the overall average complexity of obtaining the tightest UTVPI-OA using the above algorithm is \( O(4 \binom{n}{2} Z(m, n)) = O(4 \binom{n}{2}(m + n)mnL) = O((m + n)mn^2L). \) If we make the usual assumptions that \( m \approx O(n) \), and that the coefficients fit in normal integers (32 or 64 bit), this further simplifies to \( O(n^5) \) on average.

\(^6\)An improvement from the naive method that needs to be run \( 2 \binom{n}{2} \) times. This improvement shows one more instance why Octagons are a unique polyhedra among all abstract domain.

\(^7\)For some discussion on this, please see our earlier work [UC13, Upa13].

5.2 FM based algorithm

The FM based algorithm is based on a series of projections (along the Octagonal directions) and rotations. While the rotation operation is trivially linear in complexity, the FM projection, along with its internal redundancy elimination, is the most time-consuming operation. For a n-dimensional system, \( \binom{n}{2} \) FM projections to 2-d planes need to be made. The 2-d to 1-d projections can be trivially done by a scan.

It is well known that FM has high complexity. At each step of projection, the number of constraints increases quadratically; meaning, the total complexity could theoretically be \( \binom{n}{2}^2 \) which is doubly-exponential [Pug91]. While the above worst-case complexity is for extreme pathological examples, the core FM algorithm remains highly simple, and easily implementable. Moreover, excellent implementations like Omega [Pug91], FMLib [Pou] and ISL [Ver10] exist.

We may assume that for a well-behaved and typical system from polyhedral compilation (with \( m \approx O(n) \)), the complexity to obtain a 2-d projection using FM is on average \( Y(m, n) \approx O(f(\hat{s}, L, n^k)). \) Here, \( \hat{s} \) is the average sparsity of the constraints, and \( k \) is a small constant that is dependent on run-time parameters of the constraint matrix (like redundancy) [Pug91]. So, the overall complexity of the FM based OA algorithm will be \( O(\binom{n}{2} Y(m, n)) \approx O((f(\hat{s}, L, n^{k+2})). \)

With the above assumptions, we believe that the time complexity of obtaining the tightest UTVPI-OA using our FM-based algorithm could be a low order polynomial in \( n \).

5.3 Discussion and Perspectives

Applicability of approximations in polyhedral model:

It is well understood that over/under approximations (OA/UA) that preserve soundness have been accommodated in polyhedral compilation at various phases; dependence analysis needs OA [DRV00], the affine-scheduling needs UA [UC13] while the code-generation again needs an OA [Upa13]. So, our algorithms can directly be applied to dependence analysis and code-generation. However, for affine scheduling, polyhedral duality needs to be exploited to obtain UA of loop transformation search space. (OA of dual results in UA of primal [UC13].) Approximations also have applicability in the performance analysis of ACLs, like Cache Miss Calculations [BKPS17]. For more discussions, please see Section 7.

LP vs. ILP and tightness of the approximation: Our algorithms use rational linear programming (LP) and rational Fourier-Motzkin, not integer based methods (like ILP and integer shadows). This means that while they will return the tightest rational Octagonal OA, it will not be the tightest integer OA; the former will be an OA of the latter.
Redundancy removal: Our algorithms return the tightest (rational) UTVPI-OA without requiring an irredundant description of the input polyhedron. The latter is a hard problem; one approach to solve it uses Chernikova’s algorithm for $H \rightarrow V$ conversion.

Complexity and real-world scalability: The Quintic ($\approx n^5$) complexities of our algorithms indeed look large, but we conjecture that they will be better than the larger (exponential) cost that has to paid for the earlier Chernikova based algorithm. Also, assuming a cubic complexity for linear programming, the polyhedral scheduling itself reduces to quintic complexity when the dependence graph edges are taken into consideration [UC13, Upa13]. So, our algorithms are of comparable complexity when compared to affine scheduling, the most expensive phase of polyhedral compilation.

Limitations: LP Algorithm: The LP algorithm has to rely on a library implementation of simplex. While it makes the overall implementation simpler, as good libraries already exist, it will also incur a static cost every time (for example, to setup the simplex table).

Limitations: FM Algorithm: Though the FM algorithm does not incur static-cost, it heavily relies on the presence of a fast and efficient implementation, which seems definitely possible with the success of libraries like Omega, FMLib and ISL. We also believe that the FM based algorithm will scale well if and only if it is supported by a good implementation: one that does effective and efficient redundancy elimination.

LP and FM: constant-fold improvements: Our algorithms exploit the Octagonal nature of the OA that they aim to construct. Our LP based algorithm, by constructing the joint search space, incurs a cost of only one LP call per Octagonal direction, thereby finding the UTVPI-OA hyperplanes in both the opposite directions. Our FM based algorithm makes projections for pairs of variables. It also makes further improvements by rotating only the shadows (not the original polyhedron), thereby having an additional constant-fold improvement over a naïve FM based algorithm that does projections on all the Octagonal directions. In total, while the LP based algorithm incurs $O \left(4 \binom{n}{2} Z(m, n)\right)$ complexity, the improved FM based algorithm incurs $O \left(\binom{n}{2} Y(m, n)\right)$.

Parallelizability of our algorithms: It is also crucial to note that our two OA algorithms are trivially parallelizable. This is because, the computations in each of the $\binom{n}{2}$ Octagonal directions can proceed independent of each other. So, a simple parallelization of our algorithms could speed them up further when compared to the Chernikova based one.

Completion of tool chain: Both of our algorithms depend on development of the complete toolchain, including linking up with monotizing transformation for $P_{\text{UTVPI}} \rightarrow P_{\text{DBM}}$ conversion and Bellman-Ford to return feasibility.

6 Related Work

(UTVPI Algorithms) The combinatorial optimization community [Sch86] has been fascinated by (UTVPI polyhedra because of their simplicity as well as their improved complexities. Two notable works are by Aspvall-Shiloach [AS80] who gave a polynomial time algorithm, and Hochbaum-Noar [HN94] who gave strongly polynomial time algorithm for checking feasibility of TVPI systems.

Abstract domains Cousot et al. [CC77] were the pioneers in introducing abstract domains for program analysis, the interval abstract domain, as well as the Convex Polyhedra [CH78] abstract domain. This was further extended by Miné [Min06, Min04] who proposed the Octagon (UTVPI) abstract domain.

Over/Under Approximations There has not been considerable work in developing Over or Under approximation strategies from one numerical abstract domain to another.

Over-approximating convex polyhedra into Interval (box) polyhedra is rather trivial and hence folklore.

Antoine Miné [Min06, Min04]—father of the Octagon abstract domain, proposed the first-ever algorithm [Min04, Section 3.5.2, p68], [Min06, Section 4.3] to find the Zone (DBM) and UTVPI OA of a given convex polyhedron. To the best of our knowledge, there does not exist any other algorithm that finds the tightest Octagonal OA other than Miné’s algorithm. The above algorithm however requires enumeration of generators of the original convex polyhedra using well-established methods such as the Chernikova’s algorithm, and is hence less practical.

Our algorithms directly operate on the $H$-form of the given polyhedra, and thereby avoid using Chernikova’s algorithm. They effectively leverage the power of duality and projections.

Simon et al. [SK10] proposed strategies to obtain TVPI-OA of a linear inequality. While the authors admit that their approximations may not be able to find the tightest OA, they do not observe any loss of precision in program analysis.

Upadrastra et al. [UC13] proposed sub-polyhedral scheduling using (UTVPI) polyhedra in order to address the scalability challenges from the affine scheduling problem. They proposed two heuristics—that do not ensure tightness—to obtain the (UTVPI) polyhedral under-approximations; under-approximations of the Farkas’ (scheduling) polyhedra so as to preserve the program semantics. In many of the static analysis problems, there is a need to find over-approximation to preserve the soundness of the analysis.

7 Some Applications

In this preliminary work, we proposed two new algorithms for finding UTVPI/Octagon Over approximations of Convex Polyhedra. We envision various uses for our algorithm both
within the static analysis community as well as in polyhedral compilation. Here are some possibilities.

**Program analysis (Abstract Interpretation)** In the foundational work by Miné [Min06, Min04], the Octagon abstract domain was shown to be effective in addressing the scalability challenges in abstract interpretation. As our algorithms avoid the computation of generator representation of the polyhedra to be over-approximated, they could be used to improve the scalability. While it is true that some problems in static analysis do need Integer UTVPI approximations, there is a scope to use our algorithms which propose rational (and not integer) OA because of their tightest approximation feature. A scalable and tight OA engine also has applications like finding the rank and termination [BAG14].

**Cache modeling of Affine Programs** Bao et al. [BKPS17] recently proposed an analytical modeling of cache misses of ACL programs. The proposed method relies on various (rational and integer) polyhedral operations like Union, Intersection, Difference and Coalesce. Though relying on general convex polyhedra results in impressive results, the authors report scalability with some operations (like Difference and Coalesce). The above can be improved by relying on UTVPI polyhedra on which all the above operations can be accommodated in a cubic worst-case time. There would however be a loss of precision, and it has to be practically seen how the precision vs. scalability trade-off manifests.

**Sub-polyhedral Code-generation** Upadrasta et al. [Upa13, Ch. 9] demonstrated how (U)TVPI Over-Approximations can be used to improve scalability of the classic QRW [QRW, Bas04] code-generation algorithm. Their proposed technique requires computation of (U)TVPI Over-Approximations of the polyhedral domains to be scanned for code-generation. As our algorithms guarantee tightest UTVPI OA, they can directly be used in the proposed UTVPI-QRW algorithm. Since it is well understood that at code-generation time, most of the constraints of polyhedra are mostly (U)TVPI, it helps to rely on our FM based algorithm.

**Index Set Splitting based Parallelization** Griebel et al. [GFL00] pioneered the Index Set Splitting (ISS) based parallelization scheme based on application of Transitive Closure on the Polyhedral Reduced Dependence Graph (PRDG). This was later extended by Bielecki et al. [BP16] in the TRACO project. Verdoolaege et al. [VCB11] study the scalability of exact and approximate transitive closure based parallelization schemes, comparing with the previous work of Kelly et al. [KPRS96], as well as the effectiveness of over (vs. under) approximations of PRDG. Our OA algorithms can easily be applied for computation of approximate transitive closure.

### 8 Implementation, Conclusions and Future Work

We briefly discuss our implementation, and give conclusions and future work.

#### 8.1 Implementation and availability

We have prototype implementation our two algorithms in the Integer Set Library (ISL) version 0.18 [Ver10]. Our LP based algorithm uses ISL’s implementation of Parkas’ lemma to obtain the search space of UTVPI hyperplanes, and ISL’s PIP solver to encode our LP formulation with the objective cost function. Our FM based algorithm uses ISL’s default (integer) FM algorithm to implement the Projection based OA algorithm. The implementation of our two algorithms is available [UTV].

#### 8.2 Conclusions

In this preliminary work, we present two new polyhedral over-approximation algorithms, which rely solely on hyperplane representation and so avoid the usage of vertex enumeration algorithms. Our improvements overcome the limitations of the state-of-the-art algorithm used extensively in static analysis. Both of our algorithms are designed in a way that guarantee computation of tightest over-approximation provided it exists. We feel that our algorithms are unique because they rely on the previously unexplored geometric properties of Octagons. We also provide a preliminary implementation for both of our algorithms in the Integer Set Library [Ver10] and also enumerate its few applications from program analyses and transformations.

#### 8.3 Future work

Our work is preliminary and on-going. Our future work involves building a complete scalable toolchain: (i) Using (rational) FM from FMLib [Pou] avoiding usage of integer FM (that is currently in ISL) (ii) linking up with Bellman-Ford. (iii) doing extensive scalability tests on both real world examples as well as artificial examples to illustrate complexity.

**Acknowledgments:** We are sincerely thankful to Sanjay Rajopadhye for his valuable appreciation and feedback towards both of our algorithms and encouragement towards this paper. We thank Sven Verdoolaege for his clarification on ISL’s integer Fourier-Motzkin implementation. We also thank the reviewers of IMPACT-2019 and Louis-Noël Pouchet for their detailed comments that helped in immensely improving the presentation of the paper. We also thank Sukrut Sridhar Rao and Prateek Kumar for their feedback. This work was partly supported by a faculty grant from AMD.

**References**

Efficient Algorithms for UTVPI Polyhedral Over-Approximation


IMPACT 2019, January 23, 2019, Valencia, Spain