A Case for Strongly Polynomial Time Sub-Polyhedral Scheduling Using Two-Variable-Per-Inequality Polyhedra

Ramakrishna Upadrasta[1,2] Albert Cohen[1]

[1]PARKAS group, INRIA, École Normale Supérieure

[2]Université Paris-Sud 11

January the 23rd, 2012

Outline

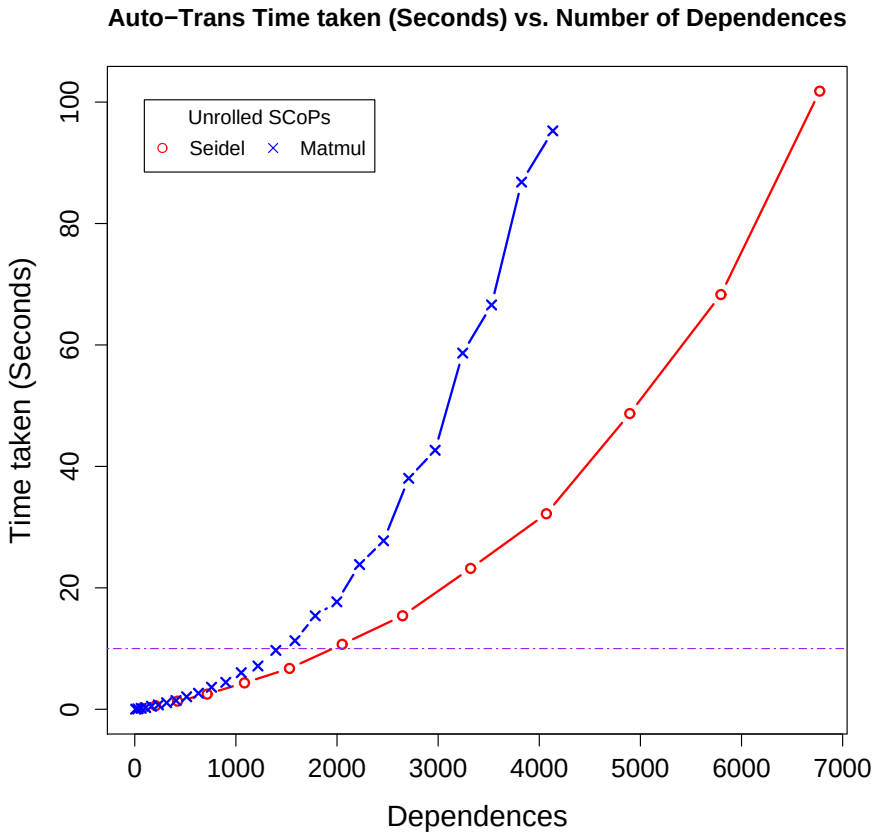
- Unscalability of Polyhedral Scheduling
- Sub-Polyhedral Varieties and Summary (TVPI and UTVPI)
- Sub-Polyhedral Scheduling
- Under-Approximation Algorithms: TVPI
  - The problem of finding Under-Approximation
  - Convexity, TVPI-UA and per-constraint methods
  - Median method and LP based methods
  - Theoretical comparison
- Experimental Results
- Conclusions and Future Work

Unscalability of Polyhedral Scheduling: A Motivation

Our goal: Compile larger and larger SCoPs using polyhedral model.

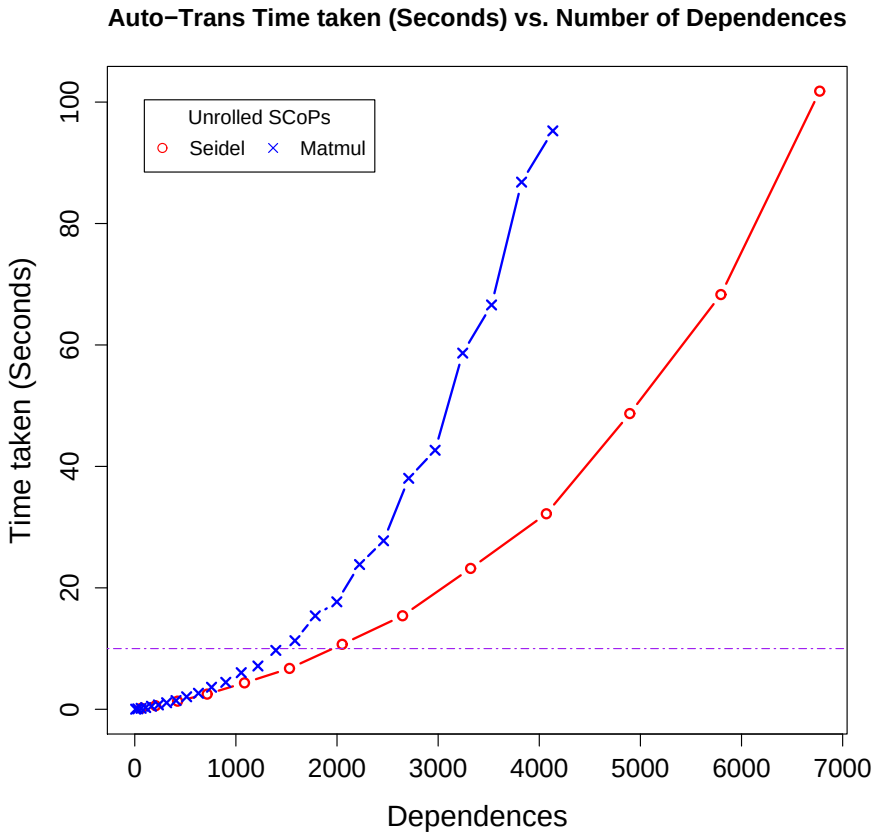
Unscalability of Polyhedral Scheduling: A Motivation

Our goal: Compile larger and larger SCoPs using polyhedral model.



Unscalability of Polyhedral Scheduling: A Motivation

Our goal: Compile larger and larger SCoPs using polyhedral model.

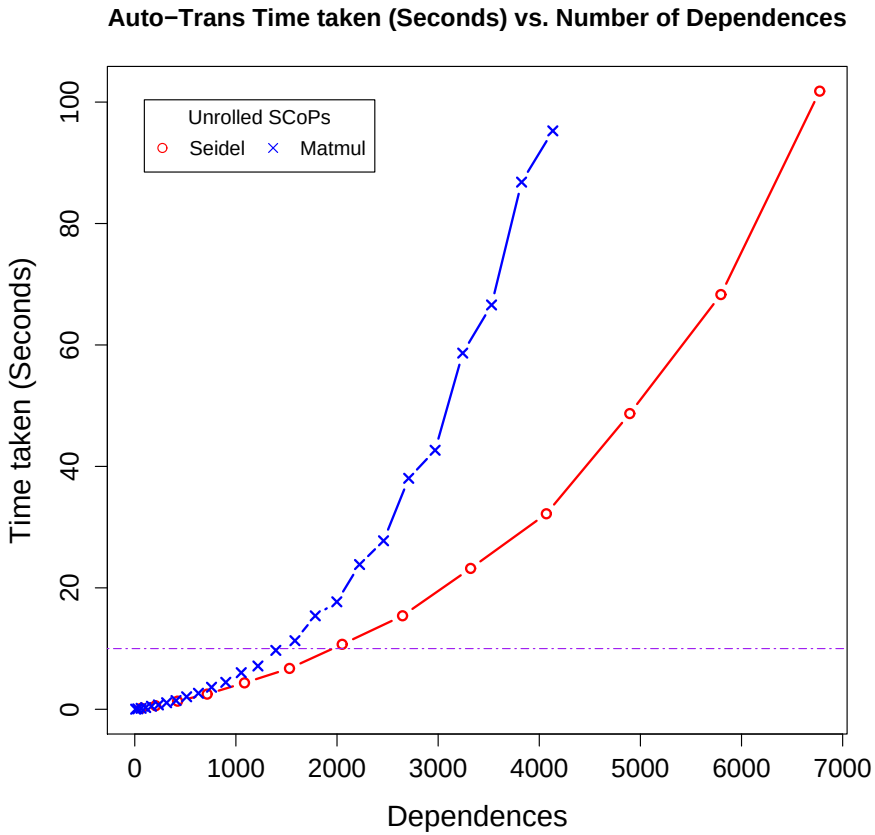


- $\approx |V|^5$ increase
  ($|V|$: #statements)
- $>$ 10s compilation limit

Scalability problem in scheduling
(general/real-time compilers)

Unscalability of Polyhedral Scheduling: A Motivation

Our goal: Compile larger and larger SCoPs using polyhedral model.



- $\approx |V|^5$ increase
  ($|V|$: #statements)
- $>$ 10s compilation limit

Scalability problem in scheduling
(general/real-time compilers)

Our solution: Algorithms using approximations of polyhedra.

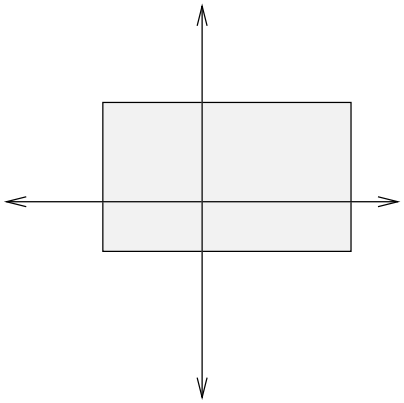
Sub-Polyhedra

Sub-Polyhedral Varieties

Theme: Trade precision for cost.

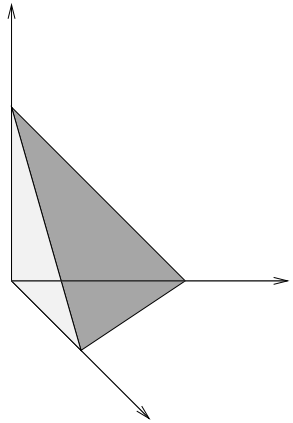
Sub-Polyhedral Varieties

Theme: Trade precision for cost.



Interval

$$a \leq x_i \leq b$$

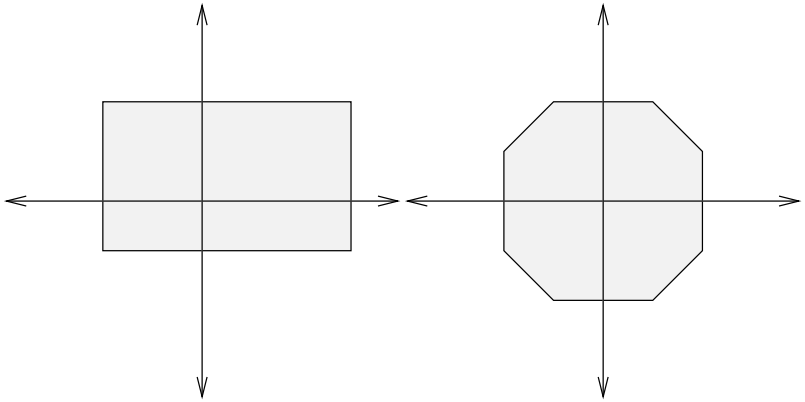


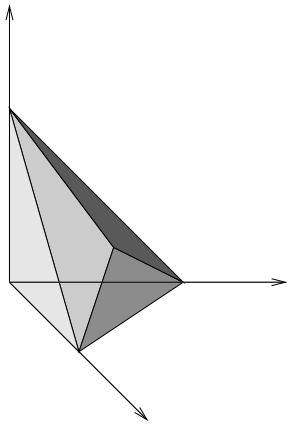
Convex Polyhedra

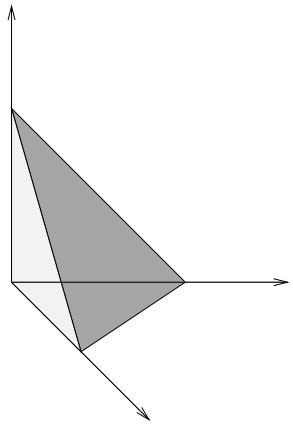
$$\sum a_i x_i \leq c$$

Sub-Polyhedral Varieties

Theme: Trade precision for cost.



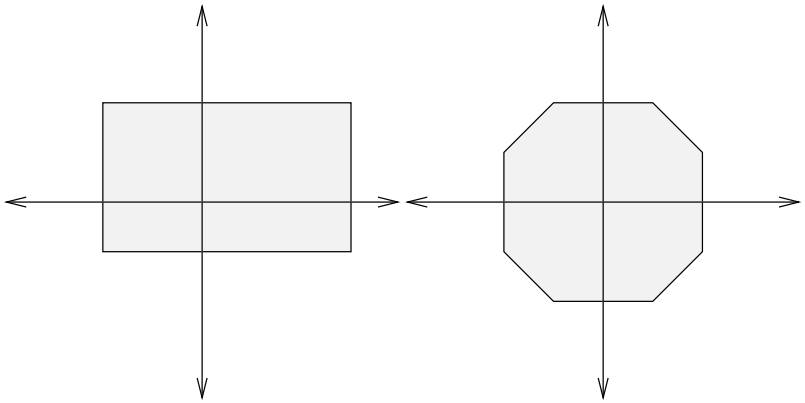


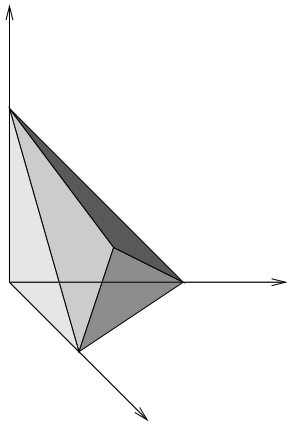


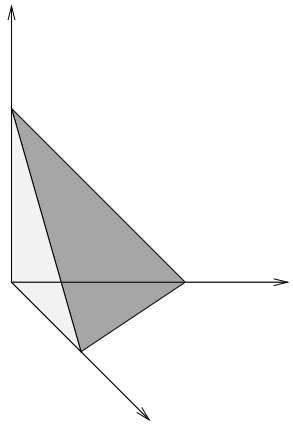
Interval	Octagon (UTVPI)	TVPI	Convex Polyhedra
	Unit Two Variable Per Inequality	Two Variable Per Inequality	
$a \leq x_i \leq b$	$\pm x_i \pm x_j \leq c$	$ax_i + bx_j \leq c$	$\sum a_i x_i \leq c$

Sub-Polyhedral Varieties

Theme: Trade precision for cost.







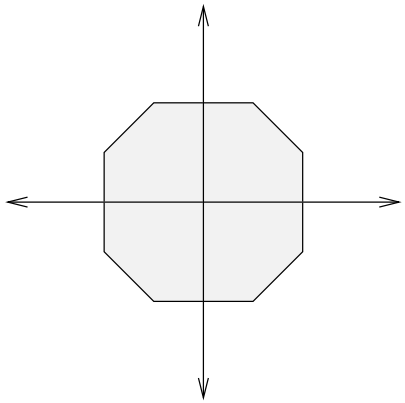
Interval	Octagon (UTVPI)	TVPI	Convex Polyhedra
	Unit Two Variable Per Inequality	Two Variable Per Inequality	
$a \leq x_i \leq b$	$\pm x_i \pm x_j \leq c$	$ax_i + bx_j \leq c$	$\sum a_i x_i \leq c$

$$\xrightarrow[\text{Precision}]{}$$

$$\text{Intervals} \subset \text{Octagons (UTVPI)} \subset \text{TVPI} \subset \text{Conv.Poly}$$

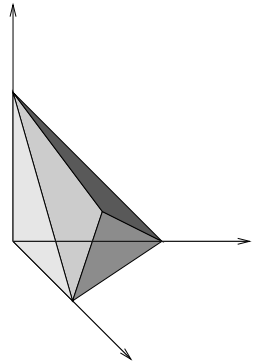
$$\xrightarrow[\text{Cost}]{}$$

Sub-Polyhedra Summary: TVPI and UTVPI



UTVPI (Octagon)

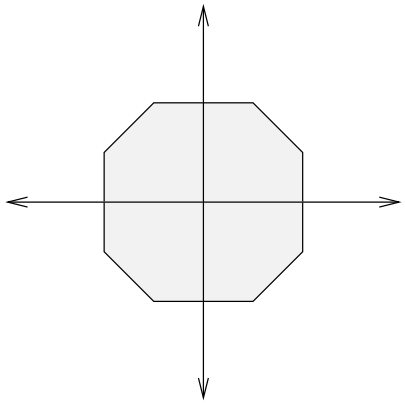
$\pm x_i \pm x_j \leq c$



TVPI

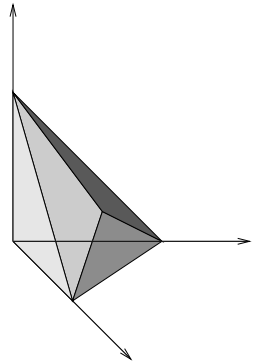
$ax_i + bx_j \leq c$

Sub-Polyhedra Summary: TVPI and UTVPI



UTVPI (Octagon)

$\pm x_i \pm x_j \leq c$

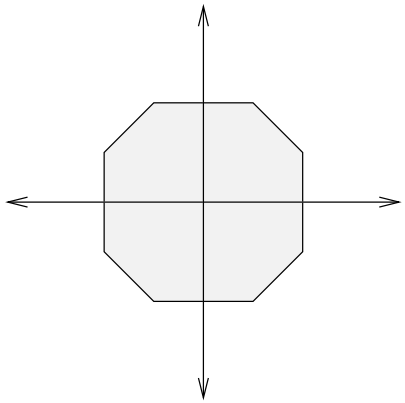


TVPI

$ax_i + bx_j \leq c$

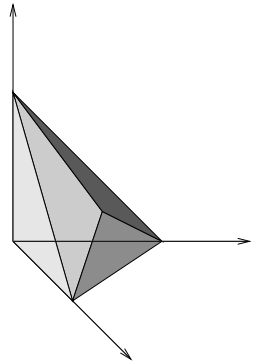
Feasibility	$\Theta(mn)$ (Bellman-Ford)	$\Theta(mn^2 \log m)$ Hochbaum-Naor-94 (Fourier-Motzkin special case)

Sub-Polyhedra Summary: TVPI and UTVPI



UTVPI (Octagon)

$\pm x_i \pm x_j \leq c$



TVPI

$ax_i + bx_j \leq c$

	UTVPI (Octagon)	TVPI
Feasibility	$\Theta(mn)$ (Bellman-Ford)	$\Theta(mn^2 \log m)$ Hochbaum-Naor-94 (Fourier-Motzkin special case)
Optimization	$\Theta(mn)$ (Bellman-Ford)	$\Theta(m^3 n^2 \log A)$ Wayne-99 (Min-cost flow)

Better worst case time complexities than general (convex) polyhedra.

Sub-Polyhedral Scheduling

Schedule Polyhedron Approximation: Motivation

Reminder: Feautrier's scheduler:

Input: Dependence polyhedron per e: $\mathcal{D}_e(\lambda, \mathcal{I}, \mathcal{N})$

- Apply Farkas lemma $\Longrightarrow$ Farkas polyhedron $\mathcal{P}_e(\mu, \lambda)$ on edge e.
- Construct overall system $\mathcal{P} = \bigcap_{e \in E} \mathcal{P}_e$.
- Use LP techniques to solve $\mathcal{P}$ for feasible solution points.

Schedule Polyhedron Approximation: Motivation

Reminder: Feautrier's scheduler:

Input: Dependence polyhedron per e: $\mathcal{D}_e(\lambda, \mathcal{I}, \mathcal{N})$

- Apply Farkas lemma $\Longrightarrow$ Farkas polyhedron $\mathcal{P}_e(\mu, \lambda)$ on edge e.
- Construct overall system $\mathcal{P} = \bigcap_{e \in E} \mathcal{P}_e$.
- Use LP techniques to solve $\mathcal{P}$ for feasible solution points.

Our method: Under-Approximate (UA) the Farkas system $\mathcal{P}$.

Schedule Polyhedron Approximation: Motivation

Reminder: Feautrier's scheduler:

Input: Dependence polyhedron per e: $\mathcal{D}_e(\lambda, \mathcal{I}, \mathcal{N})$

- Apply Farkas lemma $\Longrightarrow$ Farkas polyhedron $\mathcal{P}_e(\mu, \lambda)$ on edge e.
- Construct overall system $\mathcal{P} = \bigcap_{e \in E} \mathcal{P}_e$.
- Use LP techniques to solve $\mathcal{P}$ for feasible solution points.

Our method: Under-Approximate (UA) the Farkas system $\mathcal{P}$.

- Legal because $\mathcal{P}$ describes the set of solution points.
  (UA $\Longrightarrow$ we will be throwing away some of them.)
- Works for any *Feautrier-family* of affine transformations.
  (Ex: Forward Communications Only transformation in PLuTo.)
- Each $\mathcal{P}_e$ is *very* small $\Longrightarrow$ UA can be done per dependence edge.

Scheduling with Sub-Polyhedra

New constraint system: $\mathcal{P}_a = \bigcap_{e \in E} \mathrm{UA}(\mathcal{P}_e)$

Scheduling with Sub-Polyhedra

New constraint system: $\mathcal{P}_a = \bigcap_{e \in E} \mathrm{UA}(\mathcal{P}_e)$

Scheduling algorithm using TVPI approximation of constraints

- Under-approximate each $\mathcal{P}_e \Longrightarrow \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$
- Construct under-approximated system $\mathcal{P}_a = \bigcap_{e \in E} \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$
- Use Sub-Polyhedral methods to find feasible points in $\mathcal{P}_a$.

Scheduling with Sub-Polyhedra

New constraint system: $\mathcal{P}_a = \bigcap_{e \in E} \mathrm{UA}(\mathcal{P}_e)$

Scheduling algorithm using TVPI approximation of constraints

- Under-approximate each $\mathcal{P}_e \Longrightarrow \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$
- Construct under-approximated system $\mathcal{P}_a = \bigcap_{e \in E} \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$
- Use Sub-Polyhedral methods to find feasible points in $\mathcal{P}_a$.

Overall complexity of scheduling: $T_{sched} = T_{UA} + T_{OPT}(\mathcal{P}_a)$

- worstcase strongly polynomial time
- $\approx \mathcal{O}(mn)$ in average/common case.

Scheduling with Sub-Polyhedra

New constraint system: $\mathcal{P}_a = \bigcap_{e \in E} \mathrm{UA}(\mathcal{P}_e)$

Scheduling algorithm using TVPI approximation of constraints

- Under-approximate each $\mathcal{P}_e \Longrightarrow \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$
- Construct under-approximated system $\mathcal{P}_a = \bigcap_{e \in E} \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}_e)$
- Use Sub-Polyhedral methods to find feasible points in $\mathcal{P}_a$.

Overall complexity of scheduling: $T_{sched} = T_{UA} + T_{OPT}(\mathcal{P}_a)$

- worstcase strongly polynomial time
- $\approx \mathcal{O}(mn)$ in average/common case.

Question: How to find $\mathcal{P}_a$?

TVPI Under-Approximation Algorithms

The Problem of Finding Under-Approximation

Input: General convex polyhedron $\mathcal{P}$ in constraint form.
Goal: Compute $\mathcal{P}_a = \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}) \subseteq \mathcal{P}$

- A guarantee that $\mathcal{P}_a \neq \phi$ for non-trivial cases.
- Simple algorithm:
  - ~~generator representation~~
  - using linear programming?

The Problem of Finding Under-Approximation

Input: General convex polyhedron $\mathcal{P}$ in constraint form.
Goal: Compute $\mathcal{P}_a = \mathrm{UA}_{\mathrm{TVPI}}(\mathcal{P}) \subseteq \mathcal{P}$

- A guarantee that $\mathcal{P}_a \neq \phi$ for non-trivial cases.
- Simple algorithm:
  - ~~generator representation~~
  - using linear programming?

Our solution strategy : Use dual transformation

$$\mathcal{P} \rightarrow \mathcal{P}^* \rightsquigarrow \mathrm{OA}(\mathcal{P}^*) \rightarrow \mathrm{UA}(\mathcal{P})$$

- Only a *re-interpretation* of $\mathcal{P}$'s constraints in dual space.
- An approximation scheme $\Rightarrow$ family of approximation algorithms.

Convexity Basics and TVPI Approximation

Primal Space

Dual Space

Convexity Basics and TVPI Approximation

Primal Space		Dual Space
Constraints	$\equiv$	Generators (Vertices, Rays)

Convexity Basics and TVPI Approximation

Primal Space		Dual Space
Constraints	$\equiv$	Generators (Vertices, Rays)
$\mathcal{P} = \{\mathbf{x} \mid A\mathbf{x} + \mathbf{b} \geq \mathbf{0}\}$	$\equiv$	$\mathcal{K} = \text{cone}\begin{pmatrix} A^T & \mathbf{0} \\ \mathbf{b}^T & 1 \end{pmatrix}$
$\mathcal{P} = \left\{\mathbf{x} \mid \begin{pmatrix} A & \mathbf{b} \\ \mathbf{0}^T & 1 \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \geq \mathbf{0}\right\}$		

Convexity Basics and TVPI Approximation

Primal Space		Dual Space
Constraints	$\equiv$	Generators (Vertices, Rays)
$\mathcal{P} = \{\mathbf{x} \mid A\mathbf{x} + \mathbf{b} \geq \mathbf{0}\}$	$\equiv$	$\mathcal{K} = \text{cone}\begin{pmatrix} A^T & \mathbf{0} \\ \mathbf{b}^T & 1 \end{pmatrix}$
$\mathcal{P} = \left\{\mathbf{x} \mid \begin{pmatrix} A & \mathbf{b} \\ \mathbf{0}^T & 1 \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \geq \mathbf{0}\right\}$		
TVPI Two-Variable-Per-Inequality	$\equiv$	TCPV Two-Components-Per-Vector

Convexity Basics and TVPI Approximation

Primal Space		Dual Space
Constraints	$\equiv$	Generators (Vertices, Rays)
$\mathcal{P} = \{\mathbf{x} \mid A\mathbf{x} + \mathbf{b} \geq \mathbf{0}\}$	$\equiv$	$\mathcal{K} = \text{cone}\begin{pmatrix} A^T & \mathbf{0} \\ \mathbf{b}^T & 1 \end{pmatrix}$
$\mathcal{P} = \left\{\mathbf{x} \mid \begin{pmatrix} A & \mathbf{b} \\ \mathbf{0}^T & 1 \end{pmatrix}\begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} \geq \mathbf{0}\right\}$		
TVPI Two-Variable-Per-Inequality	$\equiv$	TCPV Two-Components-Per-Vector
UA of system $\mathcal{P}$ (constraint form) $\mathcal{P} \supseteq \mathcal{P}_a$	$\equiv$	OA of system $\mathcal{K}$ (generator form) $\mathcal{K} \subseteq \mathcal{K}_a$

Per-Constraint Approximation Using Duality

Polarity framework: $\mathcal{P} \supseteq \mathcal{P}_a$

Per-Constraint Approximation Using Duality

Polarity framework: $\mathcal{P} \supseteq \mathcal{P}_a \iff \mathcal{K} \subseteq \mathcal{K}_a$

Per-Constraint Approximation Using Duality

Polarity framework: $\mathcal{P} \supseteq \mathcal{P}_a \iff \mathcal{K} \subseteq \mathcal{K}_a$

Given:

$$\mathcal{P} = \{\mathbf{x} | \mathbf{a}_1\mathbf{x} \geq b_1; \ldots; \mathbf{a}_j\mathbf{x} \geq b_j; \ldots; \mathbf{a}_m\mathbf{x} \geq b_m;\}$$

finding

$$\mathcal{P}_a = \{\mathbf{x} | \mathbf{a}_1\mathbf{x} \geq b_1; \ldots; \mathrm{UA}(\mathbf{a}_j\mathbf{x} \geq b_j); \ldots; \mathbf{a}_m\mathbf{x} \geq b_m;\}$$

Per-Constraint Approximation Using Duality

Polarity framework: $\mathcal{P} \supseteq \mathcal{P}_a \iff \mathcal{K} \subseteq \mathcal{K}_a$

Given:

$$\mathcal{P} = \{\mathbf{x} | \mathbf{a}_1\mathbf{x} \geq b_1; \ldots; \mathbf{a}_j\mathbf{x} \geq b_j; \ldots; \mathbf{a}_m\mathbf{x} \geq b_m;\}$$

finding

$$\mathcal{P}_a = \{\mathbf{x} | \mathbf{a}_1\mathbf{x} \geq b_1; \ldots; \mathrm{UA}(\mathbf{a}_j\mathbf{x} \geq b_j); \ldots; \mathbf{a}_m\mathbf{x} \geq b_m;\}$$

$\equiv$ Given:

$$\mathcal{K} = \mathrm{cone}\left\{\begin{pmatrix} \mathbf{a}_1^T \\ b_1 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{a}_j^T \\ b_j \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{a}_m^T \\ b_m \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}\right\}$$

finding

$$\mathcal{K}_a = \mathrm{cone}\left\{\begin{pmatrix} \mathbf{a}_1^T \\ b_1 \end{pmatrix}, \ldots, \mathrm{OA}\left(\begin{pmatrix} \mathbf{a}_j^T \\ b_j \end{pmatrix}\right), \ldots, \begin{pmatrix} \mathbf{a}_m^T \\ b_m \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}\right\}$$

A Per-Generator Over-Approximation Method

Problem: How to find the TCPV-OA of a non-TCPV vector $\mathcal{K}^j$?

A Per-Generator Over-Approximation Method

Problem: How to find the TCPV-OA of a non-TCPV vector $\mathcal{K}^j$?

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix}$$

A Per-Generator Over-Approximation Method

Problem: How to find the TCPV-OA of a non-TCPV vector $\mathcal{K}^j$?

$$\underbrace{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix}}_{\text{non-TCPV}} = \begin{pmatrix} \frac{1}{2}a_1 \\ \frac{1}{2}a_2 \\ 0 \\ \frac{b}{3} \end{pmatrix} + \begin{pmatrix} \frac{1}{2}a_1 \\ 0 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{1}{2}a_2 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}$$

A Per-Generator Over-Approximation Method

Problem: How to find the TCPV-OA of a non-TCPV vector $\mathcal{K}^j$?

$$\underbrace{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix}}_{\text{non}-\text{TCPV}} = \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ \frac{1}{2}a_2 \\ 0 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ 0 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{2}a_2 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}}$$

A Per-Generator Over-Approximation Method

Problem: How to find the TCPV-OA of a non-TCPV vector $\mathcal{K}^j$?

$$\underbrace{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix}}_{\text{non-TCPV}} = \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ \frac{1}{2}a_2 \\ 0 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ 0 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{2}a_2 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}}$$

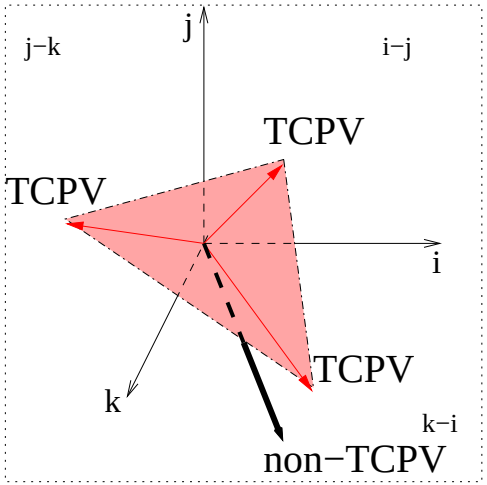
$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix} \in \text{cone} \begin{pmatrix} \frac{1}{2}a_1 & \frac{1}{2}a_1 & 0 \\ \frac{1}{2}a_2 & 0 & \frac{1}{2}a_2 \\ 0 & \frac{1}{2}a_3 & \frac{1}{2}a_3 \\ \frac{b}{3} & \frac{b}{3} & \frac{b}{3} \end{pmatrix}$$

A Per-Generator Over-Approximation Method

Problem: How to find the TCPV-OA of a non-TCPV vector $\mathcal{K}^j$?

$$\underbrace{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix}}_{\text{non-TCPV}} = \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ \frac{1}{2}a_2 \\ 0 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ 0 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{2}a_2 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}}$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix} \in \text{cone} \begin{pmatrix} \frac{1}{2}a_1 & \frac{1}{2}a_1 & 0 \\ \frac{1}{2}a_2 & 0 & \frac{1}{2}a_2 \\ 0 & \frac{1}{2}a_3 & \frac{1}{2}a_3 \\ \frac{b}{3} & \frac{b}{3} & \frac{b}{3} \end{pmatrix}$$

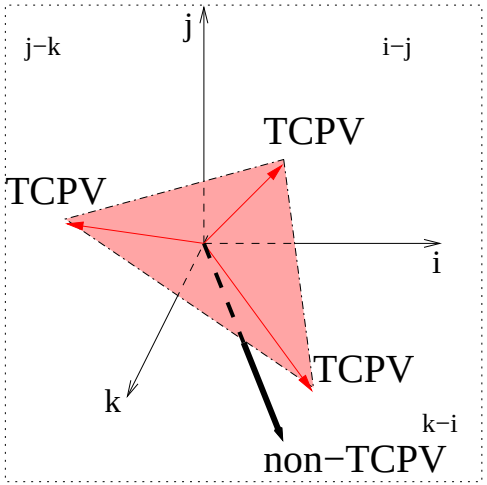


A Per-Generator Over-Approximation Method

Problem: How to find the TCPV-OA of a non-TCPV vector $\mathcal{K}^j$?

$$\underbrace{\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix}}_{\text{non-TCPV}} = \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ \frac{1}{2}a_2 \\ 0 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} \frac{1}{2}a_1 \\ 0 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{2}a_2 \\ \frac{1}{2}a_3 \\ \frac{b}{3} \end{pmatrix}}_{\text{TCPV}}$$

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b \end{pmatrix} \in \text{cone} \begin{pmatrix} \frac{1}{2}a_1 & \frac{1}{2}a_1 & 0 \\ \frac{1}{2}a_2 & 0 & \frac{1}{2}a_2 \\ 0 & \frac{1}{2}a_3 & \frac{1}{2}a_3 \\ \frac{b}{3} & \frac{b}{3} & \frac{b}{3} \end{pmatrix}$$



Solution: The decomposition is a TCPV-OA of the vector!

Median Method for Under-Approximation

The Median Method of Approximation

Problem: How to find the OA of $\mathcal{K}$?

The Median Method of Approximation

Problem: How to find the OA of $\mathcal{K}$?

Solution: Use the above strategy for each non-TCPV generator of $\mathcal{K}$.

$$\mathcal{K}_a = \text{cone}\left\{ \begin{pmatrix} \mathbf{a}_1^T \\ b_1 \end{pmatrix}, \ldots, \begin{pmatrix} \frac{1}{2}a_{j,1} & \frac{1}{2}a_{j,1} & 0 \\ \frac{1}{2}a_{j,2} & 0 & \frac{1}{2}a_{j,2} \\ 0 & \frac{1}{2}a_{j,3} & \frac{1}{2}a_{j,3} \\ \frac{b_j}{3} & \frac{b_j}{3} & \frac{b_j}{3} \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{a}_m^T \\ b_m \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \right\}$$

The Median Method of Approximation

Problem: How to find the OA of $\mathcal{K}$?

Solution: Use the above strategy for each non-TCPV generator of $\mathcal{K}$.

$$\mathcal{K}_a = \text{cone}\left\{ \begin{pmatrix} \mathbf{a}_1^T \\ b_1 \end{pmatrix}, \ldots, \begin{pmatrix} \frac{1}{2}a_{j,1} & \frac{1}{2}a_{j,1} & 0 \\ \frac{1}{2}a_{j,2} & 0 & \frac{1}{2}a_{j,2} \\ 0 & \frac{1}{2}a_{j,3} & \frac{1}{2}a_{j,3} \\ \frac{b_j}{3} & \frac{b_j}{3} & \frac{b_j}{3} \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{a}_m^T \\ b_m \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \right\}$$

$\Longleftrightarrow$ Use dual of the above strategy for each non-TVPI constraint in $\mathcal{P}$!

The Median Method of Approximation

Problem: How to find the OA of $\mathcal{K}$?
Solution: Use the above strategy for each non-TCPV generator of $\mathcal{K}$.

$$\mathcal{K}_a = \text{cone}\left\{ \begin{pmatrix} \mathbf{a}_1^T \\ b_1 \end{pmatrix}, \ldots, \begin{pmatrix} \frac{1}{2}a_{j,1} & \frac{1}{2}a_{j,1} & 0 \\ \frac{1}{2}a_{j,2} & 0 & \frac{1}{2}a_{j,2} \\ 0 & \frac{1}{2}a_{j,3} & \frac{1}{2}a_{j,3} \\ \frac{b_j}{3} & \frac{b_j}{3} & \frac{b_j}{3} \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{a}_m^T \\ b_m \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix} \right\}$$

$\Longleftrightarrow$ Use dual of the above strategy for each non-TVPI constraint in $\mathcal{P}$!

$$\mathcal{P}_a = \left\{ \mathbf{x} | \ldots; \frac{1}{2} \begin{pmatrix} a_{j,1} & a_{j,2} & 0 \\ a_{j,1} & 0 & a_{j,3} \\ 0 & a_{j,2} & a_{j,3} \end{pmatrix} \mathbf{x} \geq \frac{1}{3} \begin{pmatrix} b_j \\ b_j \\ b_j \end{pmatrix}; \ldots; \right\}$$

LP based Method for Under-Approximation

LP based Method

Main idea: Use homogenizing dimension $\Longrightarrow$ an LP problem.
(Instead of $\frac{b_j}{3} + \frac{b_j}{3} + \frac{b_j}{3} = b_j$, we use new variables $\mathbf{p}$.)

$$\mathcal{P}_a = \left\{ (\mathbf{x}, \mathbf{p}) | \ldots ; \frac{1}{2} \begin{pmatrix} a_{j,1} & a_{j,2} & 0 \\ a_{j,1} & 0 & a_{j,3} \\ 0 & a_{j,2} & a_{j,3} \end{pmatrix} \mathbf{x} \geq \begin{pmatrix} p_{j,1} \\ p_{j,2} \\ p_{j,3} \end{pmatrix} ; \ldots ; \sum \mathbf{p}^j = b_j \right\}$$

LP based Method

Main idea: Use homogenizing dimension $\Longrightarrow$ an LP problem.
(Instead of $\frac{b_j}{3} + \frac{b_j}{3} + \frac{b_j}{3} = b_j$, we use new variables $\mathbf{p}$.)

$$\mathcal{P}_a = \left\{ (\mathbf{x}, \mathbf{p}) | \ldots ; \frac{1}{2} \begin{pmatrix} a_{j,1} & a_{j,2} & 0 \\ a_{j,1} & 0 & a_{j,3} \\ 0 & a_{j,2} & a_{j,3} \end{pmatrix} \mathbf{x} \geq \begin{pmatrix} p_{j,1} \\ p_{j,2} \\ p_{j,3} \end{pmatrix} ; \ldots ; \sum \mathbf{p}^j = b_j \right\}$$

- $\sum \mathbf{p}^j = b_j$ is *context constraint.*
- $\mathbf{p}$-vector values can found by LP problem.
- Results in proper UA.

A Comparison of UA Algorithms

Comparison of Algorithms

	Median	LP	
Variants	✗	✔	One-shot, Independent

Comparison of Algorithms

	Median	LP	
Variants	✗	✔	One-shot, Independent
Cost (Strongly polynomial)	✔	✗	1 LP call per $(\mathcal{P}, m)$

Comparison of Algorithms

	Median	LP	
Variants	✗	✔	One-shot, Independent
Cost (Strongly polynomial)	✔	✗	1 LP call per $(\mathcal{P}, m)$
Feasibility guarantee	✗	✔	One-shot

Comparison of Algorithms

	Median	LP	
Variants	✗	✔	One-shot, Independent
Cost (Strongly polynomial)	✔	✗	1 LP call per $(\mathcal{P}, m)$
Feasibility guarantee	✗	✔	One-shot
Size increase	$m \times n \rightsquigarrow \hat{s}^2 m \times n$ ($\hat{s}$: average sparsity factor)		

Comparison of Algorithms

	Median	LP	
Variants	✗	✔	One-shot, Independent
Cost (Strongly polynomial)	✔	✗	1 LP call per $(\mathcal{P}, m)$
Feasibility guarantee	✗	✔	One-shot
Size increase	$m \times n \rightsquigarrow \hat{s}^2 m \times n$ ($\hat{s}$: average sparsity factor)		
Simple to implement	✔	✔	

Empirical Evaluation

Empirical Framework

Framework:

- Experiments done in PLuTo[1].
- PolyBench[2] benchmarks
- Code in C/C++ (uses PIP/PolyLib/FMLib)

[1] http://pluto-compiler.sourceforge.net/
[2] http://www.cse.ohio-state.edu/~pouchet/software/polybench

Empirical Framework

Framework:

- Experiments done in PLuTo[1].
- PolyBench[2] benchmarks
- Code in C/C++ (uses PIP/PolyLib/FMLib)

Polyhedral characteristics:

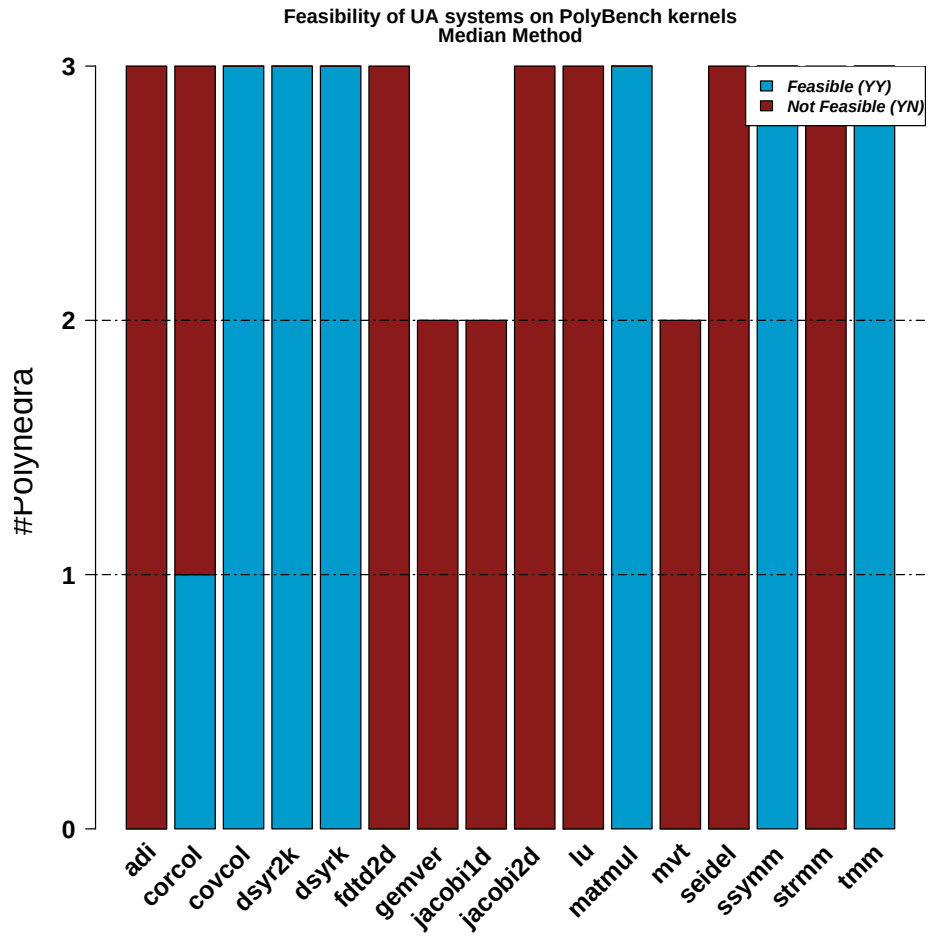
- Most dependence-polyhedral constraints are TVPI.
- A TVPI constraint is *always* a UTVPI constraint.
- Low sparsity $\hat{s} \simeq 4$ & independent of n (#dimensions).
- Need for duplication elimination (even syntactic).

Testing:

- Feasibility: PIP
- Schedule: using $\mathrm{UA}(\mathcal{P})$, usual cost function and PIP

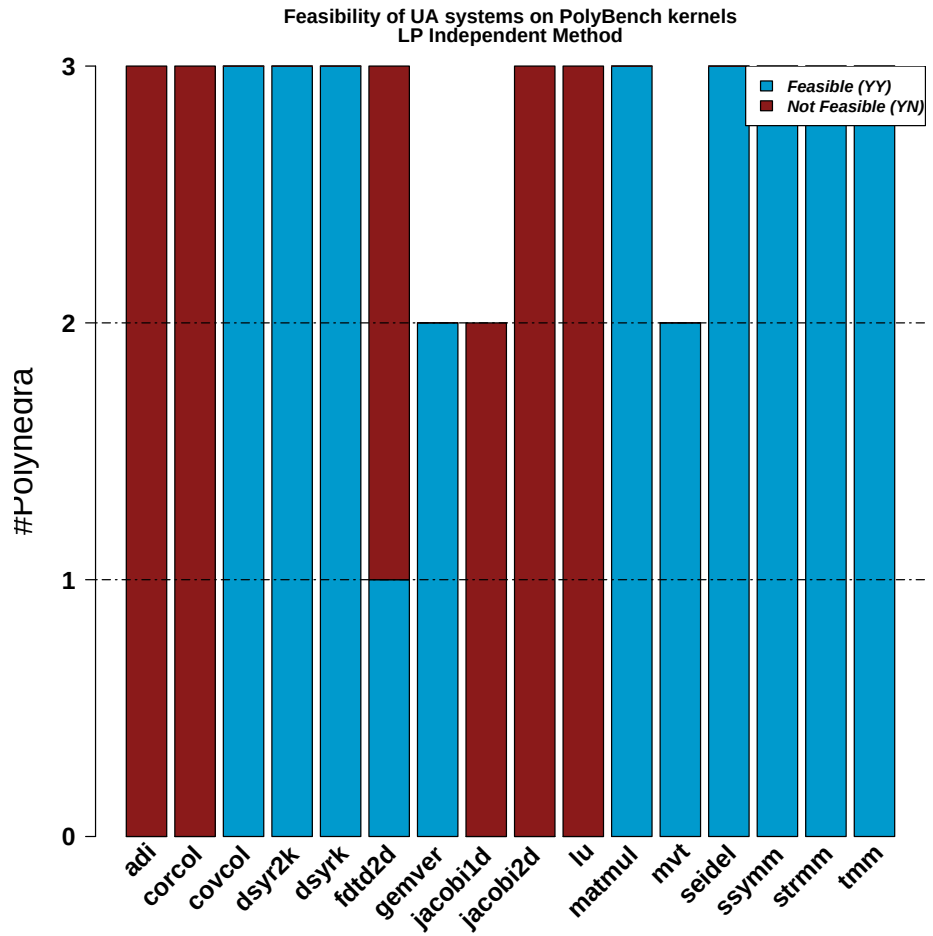
[1] http://pluto-compiler.sourceforge.net/
[2] http://www.cse.ohio-state.edu/~pouchet/software/polybench

Feasibility of $\mathrm{UA}(\mathcal{P})$: Median Method



- Total 45 Polyhedra: 19 Feasible vs. 26 Not Feasible UAs
- 6 out of 16 PolyBench benchmarks

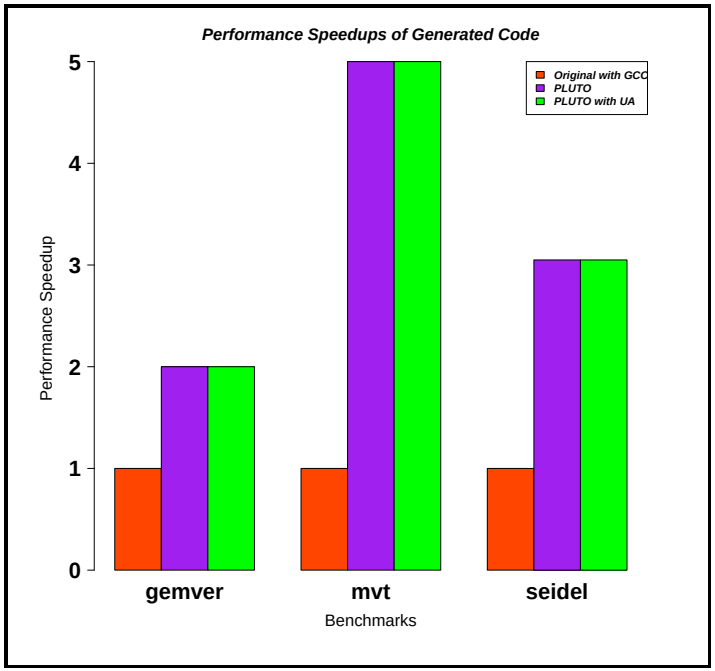
Feasibility of $\mathrm{UA}(\mathcal{P})$: LP-Independent Method



- Total 45 Polyhedra: 29 Feasible vs. 16 Not Feasible UAs
- 10 out of 16 PolyBench benchmarks

Generated Code Comparison (Par)

Details: Schedule found using $\mathrm{UA}(\mathcal{P})$, usual cost function and PIP.



Note: Implementation in progress!

Wrapping up

Summary and Contributions

Summary:

- There exists a scalability problem in polyhedral scheduling.
- Sub-Polyhedra (TVPI/UTVPI) offer better guarantees
  $\Longrightarrow$ scalability in scheduling (precision vs. cost tradeoff)

Summary and Contributions

Summary:

- There exists a scalability problem in polyhedral scheduling.
- Sub-Polyhedra (TVPI/UTVPI) offer better guarantees
  $\Longrightarrow$ scalability in scheduling (precision vs. cost tradeoff)

Contributions: Sub-Polyhedral (TVPI/UTVPI) UA algorithms

- Median method and LP based methods
  - (-) increase in system size
    $(m \times n \rightsquigarrow \hat{s}^2 m \times n)$
  - (+) ~~vertex method~~ ~~quadratic formulation~~
    Linear formulation (polynomial time)
  - (+) $\mathrm{UA}(\mathcal{P}) \neq \phi$ for considerable (29/45) cases
  - (+) generate code using UA systems
    $T_{exec}(\mathcal{P}) \cong T_{exec}(\mathrm{UA}(\mathcal{P}))$
- Experimental evidence

Challenges and Future Work (Beaucoup!)

Challenges:

- A scalable guarantee for $\mathrm{UA}(\mathcal{P}) \neq \phi$.
- No strongly polynomial time approximation possible?

Challenges and Future Work (Beaucoup!)

Challenges:

- A scalable guarantee for $\mathrm{UA}(\mathcal{P}) \neq \phi$.
- No strongly polynomial time approximation possible?

Future work:

- UTVPI approximation $\Longrightarrow$ Bellman-Ford polyhedra
  $\Longrightarrow \Theta(mn)$ polyhedral scheduling: our estimate of lower bound.
- Per dependence implementation.
- Code generation for all feasible cases.

Challenges and Future Work (Beaucoup!)

Challenges:

- A scalable guarantee for $\mathrm{UA}(\mathcal{P}) \neq \phi$.
- No strongly polynomial time approximation possible?

Future work:

- UTVPI approximation $\Longrightarrow$ Bellman-Ford polyhedra
  $\Longrightarrow \Theta(mn)$ polyhedral scheduling: our estimate of lower bound.
- Per dependence implementation.
- Code generation for all feasible cases.

Take home message: We think that sub-polyhedral scheduling

- is possible using (U)TVPI systems
- will help beat the scalability challenge.

Merci et Questions

Merci Beaucoup!

- Paul Feautrier
- Cédric Bastoul, Armin Größlinger
- Reviewers (IMPACT 2011, 2012)

"Truth is much too complicated to allow anything but approximations" – John von Neumann

Questions?

Backup

Homogenization-Polarity Framework

$$P^{\mathcal{H}} \xrightarrow{\text{homog}} C^{\mathcal{H}} \xrightarrow{\text{polar}} K^{\mathcal{V}} \xrightarrow{\text{OA}} K_a^{\mathcal{V}} \xrightarrow{\text{depolar}} C_a^{\mathcal{H}} \xrightarrow{\text{dehomog}} P_a^{\mathcal{H}}$$

- [Step 0] $P^{\mathcal{H}}$: Input P, the Polyhedron to be under-approximated.
- [Step 1] $P^{\mathcal{H}} \xrightarrow{\text{homog}} C^{\mathcal{H}}$: $C = \text{Homog}(P)$.
- [Step 2] $C^{\mathcal{H}} \xrightarrow{\text{polar}} K^{\mathcal{V}}$: $K = C^*$ (the Polar of C).
- [Step 3] $K^{\mathcal{V}} \xrightarrow{\text{OA}} K_a^{\mathcal{V}}$: $K \subseteq K_a$.
- [Step 4] $K_a^{\mathcal{V}} \xrightarrow{\text{depolar}} C_a^{\mathcal{H}}$:
- [Step 5] $C_a^{\mathcal{H}} \xrightarrow{\text{dehomog}} P_a^{\mathcal{H}}$:

Approximation Scheme

Homogenization-polarity framework: $P \supseteq P_a \iff K \subseteq K_a$

$$K = \text{cone}\left\{\begin{pmatrix} \mathbf{a}_1^T \\ b_1 \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{a}_j^T \\ b_j \end{pmatrix}, \ldots, \begin{pmatrix} \mathbf{a}_m^T \\ b_m \end{pmatrix}, \begin{pmatrix} \mathbf{0} \\ 1 \end{pmatrix}\right\}$$

$$K \in \text{cone}\,(K_a)$$

$$K^j = \begin{pmatrix} a_{j,1} \\ a_{j,2} \\ a_{j,3} \\ b_j \end{pmatrix} \in \text{cone} \begin{pmatrix} t_1^{j,1} & t_2^{j,1} & 0 \\ t_1^{j,2} & 0 & t_3^{j,2} \\ 0 & t_2^{j,3} & t_3^{j,3} \\ p_1^j & p_2^j & p_3^j \end{pmatrix} = \text{cone}(T^j)$$

Context constraints:

$$T^j = \{t_1^{j,1} + t_2^{j,1} = a_{j,1};\ t_1^{j,2} + t_3^{j,2} = a_{j,2};\ t_2^{j,3} + t_3^{j,3} = a_{j,3};\ p_1^j + p_2^j + p_3^j = b_j\}$$

Polyhedral Characteristics: FPH

	SCoP			FPH						
Bench	#L	#S	#D	P_{FPH}	n	$\hat{m}$	$\hat{m}_T$	$\| \hat{\mathcal{S}}_{m_k} \|$	$\hat{m}_E$	$\hat{m}_U$
adi	12	4	54	3	21	200	65	5	1844	614
corcol	12	6	14	3	23	22	13	5	130	77
covcol	13	7	26	3	25	18	14	4	29	55
dsyr2k	3	1	3	3	8	8	6	3	11	18
dsyrk	3	1	3	3	8	8	7	3	11	18
fdtd-2d	11	4	48	3	21	96	35	6	1010	367
gemver	7	4	13	2	15	21	15	4	48	47
jacobi-1d	4	2	10	2	11	32	14	5	232	104
jacobi-2d	6	2	14	3	13	65	15	7	1135	212
lu	5	2	10	3	11	35	17	5	232	106
matmul	3	1	3	3	10	9	7	4	20	24
mvt	4	2	11	2	10	20	12	3	46	52
seidel	3	1	27	3	9	33	15	5	168	179
ssymm	8	3	15	3	15	15	10	3	22	36

Polyhedral Characteristics: DS/DIR

SCoP				DS/DIR				
Bench	#L	#S	#D	P_{DS}	P_{DIR}	n	$\hat{m}$	$\hat{m}_T$
adi	12	4	54	90	564	10	20	19
corcol	12	6	14	38	194	10	17	16
covcol	13	7	26	41	228	16	25	24
dsyr2k	3	1	3	9	18	9	18	17
dsyrk	3	1	3	9	18	9	18	17
fdtd-2d	11	4	48	39	168	11	22	21
gemver	7	4	13	29	161	7	13	12
jacobi-1d	4	2	10	16	88	8	14	13
jacobi-2d	6	2	14	21	84	10	19	18
lu	5	2	10	12	60	12	23	22
matmul	3	1	3	9	18	11	21	21
mvt	4	2	11	31	68	7	13	12
seidel	3	1	27	37	162	13	24	23
ssymm	8	3	15	33	126	9	19	19

Execution Time Comparison

		Perf. Comparison (Seconds)			
Benchmark	Orig	Par Cur	Par New	Tiled Cur.	Tiled New
gemver	0.31	0.15	0.15	0.15	0.15
mvt	1.40	0.27	0.28	0.42	0.43
seidel	11.8	3.6	3.6	11.5	11.5

More Future Work

Algorithmic:

- Direct UTVPI-UA algorithm similar to LP method
- Integration with Bellman-Ford
- Cost function approximation
- Closeness of UA wrt. original
- Parametrization?

Implementational:

- Per-dependence method integration and feasibility stats
- Duplicate constraints elimination
- Closer integration with PLuTo: YY cases
- Offline/Online method