Loop Transformations: Convexity, Pruning and Optimization

Louis-Noël Pouchet¹ Uday Bondhugula² Cédric Bastoul³ Albert Cohen³ J. Ramanujam⁴ P. Sadayappan¹ Nicolas Vasilache⁵

¹ The Ohio State University
² IBM T.J. Watson Research Center
³ ALCHEMY group, INRIA Saclay / University of Paris-Sud 11
⁴ Louisiana State University
⁵ Reservoir Labs, Inc.

January 28, 2011
ACM 2011 Symposium on Principles of Programming Languages
Austin, TX
Compiler Optimizations for Performance

- High-level loop transformations are critical for performance...
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)
Compiler Optimizations for Performance

- **High-level loop transformations are critical for performance...**
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)

- **... But deciding the best sequence of transformations is hard!**
  - Conflicting objectives: more SIMD implies less locality, etc.
  - It is machine-dependent and of course program-dependent
  - Expressive search spaces are required, but challenge the search!
Compiler Optimizations for Performance

- High-level loop transformations are critical for performance...
  - Coarse-grain parallelism (OpenMP)
  - Fine-grain parallelism (SIMD)
  - Data locality (reduce cache misses)

- ... But deciding the best sequence of transformations is hard!
  - Conflicting objectives: more SIMD implies less locality, etc.
  - It is machine-dependent and of course program-dependent
  - Expressive search spaces are required, but challenge the search!

- Our approach:
  - Convexity: model optimization spaces as convex set (ILP, scan, project, etc.)
  - Pruning: make our spaces contain all and only semantically equivalent programs in our framework
  - Optimization: decompose in two more tractable sub-problems without any loss of expressiveness, empirical search + ILP models
Spaces of Affine Loop transformations

All unique bounded affine multidimensional schedules

All unique semantics-preserving fusion / distribution / code motion choices

All unique semantics-preserving affine multidimensional schedules
Spaces of Affine Loop transformations

All unique bounded affine multidimensional schedules

All unique semantics-preserving fusion / distribution / code motion choices

All unique semantics-preserving affine multidimensional schedules

Bounded: $10^{200}$
Legal: $10^{50}$
Empirical search: 10
Spaces of Affine Loop transformations

All unique bounded affine multidimensional schedules

1 point $\leftrightarrow$ 1 unique transformed program
Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)
Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)
- Iteration domain: represented as integer polyhedra

```plaintext
for (i=1; i<=n; ++i)
  for (j=1; j<=n; ++j)
    if (i<=n-j+2)
      ... s[i] = ...
```

$$\mathcal{P}_{S1} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ -1 & -1 & 1 & 2 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix} \geq \vec{0}$$

Diagram showing the iteration domain of $$S_1$$.
Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)
- Iteration domain: represented as integer polyhedra
- Memory accesses: static references, represented as affine functions of \( \vec{x}_S \) and \( \vec{p} \)

```cpp
for (i=0; i<n; ++i) {
    s[i] = 0;
    for (j=0; j<n; ++j)
        s[i] = s[i] + a[i][j]*x[j];
}
```

\[
f_s(\vec{x}_S) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \cdot (\begin{bmatrix} x_S^n \\ n \\ 1 \end{bmatrix})
\]

\[
f_a(\vec{x}_S) = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \cdot (\begin{bmatrix} x_S^n \\ n \\ 1 \end{bmatrix})
\]

\[
f_x(\vec{x}_S) = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \cdot (\begin{bmatrix} x_S^n \\ n \\ 1 \end{bmatrix})
\]
Polyhedral Representation of Programs

Static Control Parts

- Loops have affine control only (over-approximation otherwise)
- Iteration domain: represented as integer polyhedra
- Memory accesses: static references, represented as affine functions of $\vec{x_S}$ and $\vec{p}$
- Data dependence between S1 and S2: a subset of the Cartesian product of $\mathcal{D}_{S1}$ and $\mathcal{D}_{S2}$ (exact analysis)

```python
for (i=1; i<=3; ++i) {
    . s[i] = 0;
    . for (j=1; j<=3; ++j)
        . s[i] = s[i] + 1;
}
```

\[
\mathcal{D}_{S1 \delta S2} : \begin{bmatrix}
1 & -1 & 0 & 0 \\
1 & 0 & 0 & -1 \\
-1 & 0 & 0 & 3 \\
0 & 1 & 0 & -1 \\
0 & -1 & 0 & 3 \\
0 & 0 & 1 & -1 \\
0 & 0 & -1 & 3
\end{bmatrix} \cdot \begin{bmatrix}
i_{S1} \\
i_{S2} \\
i_{S2} \\
1
\end{bmatrix} \geq 0
\]

$S1$ iterations

$S2$ iterations
Affine Transformations for Iteration Reordering

The transformation matrix is the identity with a permutation of two rows.

\[
\begin{bmatrix}
1 & 0 \\
-1 & 0 \\
0 & 1 \\
0 & -1
\end{bmatrix}
\begin{bmatrix}
i \\
j
\end{bmatrix}
+ \begin{bmatrix}
-1 \\
2 \\
-1 \\
3
\end{bmatrix} \geq 0
\]
\[\begin{bmatrix}
i' \\
j'
\end{bmatrix} = \begin{bmatrix}
0 & 1 \\
1 & 0
\end{bmatrix}
\begin{bmatrix}
i \\
j
\end{bmatrix}
\]
\[
\begin{bmatrix}
0 & 1 \\
0 & -1 \\
1 & 0 \\
-1 & 0
\end{bmatrix}
\begin{bmatrix}
i' \\
j'
\end{bmatrix}
+ \begin{bmatrix}
-1 \\
2 \\
-1 \\
3
\end{bmatrix} \geq 0
\]

(a) original polyhedron  (b) transformation function  (c) target polyhedron

\[\text{do } i = 1, 2 \]
\[\text{do } j = 1, 3 \]
\[S(i,j)\]

\[\text{do } i' = 1, 3 \]
\[\text{do } j' = 1, 2 \]
\[S(i'=j',j'=i')\]
Affine Transformations for Iteration Reordering

Reversal Transformation

The transformation matrix is the identity with one diagonal element replaced by $-1$.

\[ \begin{bmatrix} 1 & 0 \\ -1 & 0 \\ 0 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} i' \\ j' \end{bmatrix} \geq 0 \]

(a) original polyhedron

(b) transformation function

(c) target polyhedron

\[ \begin{bmatrix} -1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} i' \\ j' \end{bmatrix} + \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix} \geq 0 \]

\[
\begin{align*}
\text{do } i &= 1, 2 \\
\text{do } j &= 1, 3 \\
S(i, j)
\end{align*}
\]

\[
\begin{align*}
\text{do } i' &= -1, -2, -1 \\
\text{do } j' &= 1, 3 \\
S(i=3-i', j=j')
\end{align*}
\]
Affine Transformations for Iteration Reordering

The transformation matrix is the composition of an interchange and reversal.

\[
\begin{pmatrix}
1 & 0 \\
-1 & 0 \\
0 & 1 \\
0 & -1
\end{pmatrix}
\begin{pmatrix}
\begin{pmatrix}
i \\
j
\end{pmatrix}
+ \\
\begin{pmatrix}
-1 \\
2 \\
-1 \\
3
\end{pmatrix}
\geq \vec{0}
\end{pmatrix}
= 
\begin{pmatrix}
0 & -1 \\
1 & 0
\end{pmatrix}
\begin{pmatrix}
i \\
j
\end{pmatrix}
\]

\[
\begin{pmatrix}
0 & -1 \\
0 & 1 \\
1 & 0 \\
-1 & 0
\end{pmatrix}
\begin{pmatrix}
\begin{pmatrix}i'\end{pmatrix}
+ \\
\begin{pmatrix}-1 \\
2 \\
-1 \\
3
\end{pmatrix}
\geq \vec{0}
\end{pmatrix}
\]

(a) original polyhedron
(b) transformation function
(c) target polyhedron

do \(i = 1, 2\)
do \(j = 1, 3\)
\(S(i, j)\)
do \(j' = -1, -3, -1\)
do \(i' = 1, 2\)
\(S(\text{i}=4-\text{j}', \text{j}=\text{i}')\)
Affine Transformations for Iteration Reordering

The transformation matrix is the composition of an interchange and reversal

\[
\begin{pmatrix}
1 & 0 \\
-1 & 0 \\
0 & 1 \\
0 & -1
\end{pmatrix}
\begin{pmatrix} i \\ j \end{pmatrix}
\begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix}
\geq \vec{0}
\]

\[
\begin{pmatrix} i' \\ j' \end{pmatrix} = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}
\begin{pmatrix} i \\ j \end{pmatrix}
\begin{pmatrix} -1 \\ 2 \\ -1 \\ 3 \end{pmatrix}
\geq \vec{0}
\]

(a) original polyhedron  
(b) transformation function  
(c) target polyhedron

\[
\begin{align*}
do & i = 1, 2 \\
do & j = 1, 3 \\
S & (i, j)
\end{align*}
\]

\[
\begin{align*}
do & j' = -1, -3, -1 \\
do & i' = 1, 2 \\
S & (i=4-j', j=i')
\end{align*}
\]
Affine Schedule

Definition (Affine multidimensional schedule)

Given a statement $S$, an affine schedule $\Theta^S$ of dimension $m$ is an affine form on the $d$ outer loop iterators $\vec{x}_S$ and the $p$ global parameters $\vec{n}$. $\Theta^S \in \mathbb{Z}^{m \times (d+p+1)}$ can be written as:

$$\Theta^S(\vec{x}_S) = \begin{pmatrix} \theta_{1,1} & \cdots & \theta_{1,d+p+1} \\ \vdots & \ddots & \vdots \\ \theta_{m,1} & \cdots & \theta_{m,d+p+1} \end{pmatrix} \cdot \begin{pmatrix} \vec{x}_S \\ \vec{n} \\ 1 \end{pmatrix}$$

$\Theta^S_k$ denotes the $k^{th}$ row of $\Theta^S$.

Definition (Bounded affine multidimensional schedule)

$\Theta^S$ is a bounded schedule if $\theta^S_{i,j} \in [x, y]$ with $x, y \in \mathbb{Z}$.
Space of Semantics-Preserving Affine Schedules

1 point $\leftrightarrow$ 1 unique semantically equivalent program (up to affine iteration reordering)
Semantics Preservation

Definition (Causality condition)

Given $\Theta^R$ a schedule for the instances of $R$, $\Theta^S$ a schedule for the instances of $S$. $\Theta^R$ and $\Theta^S$ preserve the dependence $\mathcal{D}_{R,S}$ if $\forall \langle \vec{x}_R, \vec{x}_S \rangle \in \mathcal{D}_{R,S}$:

$$\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$$

$\prec$ denotes the lexicographic ordering.

$$(a_1, \ldots, a_n) \prec (b_1, \ldots, b_m) \text{ iff } \exists i, 1 \leq i \leq \min(n, m) \text{ s.t. } (a_1, \ldots, a_{i-1}) = (b_1, \ldots, b_{i-1}) \text{ and } a_i < b_i$$
Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \succ \vec{0}$
Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) \succ \vec{0}$
- Considering the row $p$ of the scheduling matrices:

\[
\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) \geq \delta_p
\]
Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) \prec \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) > \vec{0}$
- Considering the row $p$ of the scheduling matrices:

$$\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) \geq \delta_p$$

- $\delta_p \geq 1$ implies no constraints on $\delta_k, k > p$
- $\delta_p \geq 0$ is required if $\forall k < p, \delta_k \geq 1$
Lexico-positivity of Dependence Satisfaction

- $\Theta^R(\vec{x}_R) < \Theta^S(\vec{x}_S)$ is equivalently written $\Theta^S(\vec{x}_S) - \Theta^R(\vec{x}_R) > 0$
- Considering the row $p$ of the scheduling matrices:

$$\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) \geq \delta_p$$

- $\delta_p \geq 1$ implies no constraints on $\delta_k, k > p$
- $\delta_p \geq 0$ is required if $\not\exists k < p, \delta_k \geq 1$

- Schedule lower bound:

**Lemma (Schedule lower bound)**

*Given $\Theta^R_k$, $\Theta^S_k$ such that each coefficient value is bounded in $[x, y]$. Then there exists $K \in \mathbb{Z}$ such that:*

$$\Theta^S_k(\vec{x}_S) - \Theta^R_k(\vec{x}_R) > -K \cdot \vec{n} - K$$
Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension $m$, the program semantics is preserved if the three following conditions hold:

(i) $\forall D_R, S, \delta_p^{D_{R,S}} \in \{0, 1\}$

(ii) $\forall D_R, S, \sum_{p=1}^{m} \delta_p^{D_{R,S}} = 1$

(iii) $\forall D_R, S, \forall p \in \{1, \ldots, m\}, \forall \langle \bar{x}_R, \bar{x}_S \rangle \in D_{R,S}$
Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension $m$, the program semantics is preserved if the three following conditions hold:

(i) $\forall D_R, S, \delta_p^{D_R,S} \in \{0, 1\}$

(ii) $\forall D_R, S, \sum_{p=1}^{m} \delta_p^{D_R,S} = 1$

(iii) $\forall D_R, S, \forall p \in \{1, \ldots, m\}, \forall \langle \bar{x}_R, \bar{x}_S \rangle \in D_{R,S}$
Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules \( \Theta^R, \Theta^S \ldots \) of dimension \( m \), the program semantics is preserved if the three following conditions hold:

1. \( \forall D_{R,S}, \quad \delta_p^{D_{R,S}} \in \{0, 1\} \)
2. \( \forall D_{R,S}, \quad \sum_{p=1}^{m} \delta_p^{D_{R,S}} = 1 \)
3. \( \forall D_{R,S}, \forall p \in \{1, \ldots, m\}, \forall (\vec{x}_R, \vec{x}_S) \in D_{R,S}, \)

\[ \Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) \geq \delta_p^{D_{R,S}} \]
Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension $m$, the program semantics is preserved if the three following conditions hold:

(i) $\forall D_{R,S}, \delta_p^{D_{R,S}} \in \{0, 1\}$

(ii) $\forall D_{R,S}, \sum_{p=1}^{m} \delta_p^{D_{R,S}} = 1$

(iii) $\forall D_{R,S}, \forall p \in \{1, \ldots, m\}, \forall \langle \bar{x}_R, \bar{x}_S \rangle \in D_{R,S}$,

$$\Theta^S_p(\bar{x}_S) - \Theta^R_p(\bar{x}_R) \geq \delta_p^{D_{R,S}} - \sum_{k=1}^{p-1} \delta_k^{D_{R,S}} . (K.\bar{n} + K)$$
Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension $m$, the program semantics is preserved if the three following conditions hold:

(i) $\forall D_R, S, \delta_p^{D_{R,S}} \in \{0, 1\}$

(ii) $\forall D_R, S, \sum_{p=1}^{m} \delta_p^{D_{R,S}} = 1$

(iii) $\forall D_R, S, \forall p \in \{1, \ldots, m\}, \forall \langle \vec{x}_R, \vec{x}_S \rangle \in D_{R,S},$

$$\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) - \delta_p^{D_{R,S}} + \sum_{k=1}^{p-1} \delta_k^{D_{R,S}}(K.\vec{n} + K) \geq 0$$

→ Use Farkas lemma to build all non-negative functions over a polyhedron (here, the dependence polyhedra) [Feautrier,92]
Convex Form of All Bounded Affine Schedules

Lemma (Convex form of semantics-preserving affine schedules)

Given a set of affine schedules $\Theta^R, \Theta^S \ldots$ of dimension $m$, the program semantics is preserved if the three following conditions hold:

(i) $\forall D_{R,S}, \delta_p^{D_{R,S}} \in \{0, 1\}$

(ii) $\forall D_{R,S}, \sum_{p=1}^{m} \delta_p^{D_{R,S}} = 1$

(iii) $\forall D_{R,S}, \forall p \in \{1, \ldots, m\}, \forall \langle \vec{x}_R, \vec{x}_S \rangle \in D_{R,S},$

$$\Theta^S_p(\vec{x}_S) - \Theta^R_p(\vec{x}_R) - \delta_p^{D_{R,S}} + \sum_{k=1}^{p-1} \delta_k^{D_{R,S}}(K.n + K) \geq 0$$

→ Use Farkas lemma to build all non-negative functions over a polyhedron (here, the dependence polyhedra) [Feautrier,92]

→ Bounded coefficients required [Vasilache,07]
Space of Semantics-Preserving Fusion Choices

1 point $\leftrightarrow$ 1 unique semantically equivalent program (up to "partial" statement reordering)
Fusion in the Polyhedral Model

for (i = 0; i <= N; ++i) {
    Blue(i);
    Red(i);
}

Perfectly aligned fusion
Fusion in the Polyhedral Model

```
Blue(0);
for (i = 1; i <= N; ++i) {
    Blue(i);
    Red(i-1);
}
Red(N);
```

Fusion with shift of 1
Not all instances are fused
Fusion in the Polyhedral Model

```
for (i = 0; i < P; ++i)
    Blue(i);
for (i = P; i <= N; ++i) {
    Blue(i);
    Red(i-P);
}
for (i = N+1; i <= N+P; ++i)
    Red(i-P);
```

Fusion with parametric shift of P
Automatic generation of prolog/epilog code
Fusion in the Polyhedral Model

for (i = 0; i < P; ++i)
    Blue(i);
for (i = P; i <= N; ++i) {
    Blue(i);
    Red(i-P);
}
for (i = N+1; i <= N+P; ++i)
    Red(i-P);

Many other transformations may be required to enable fusion: interchange, skewing, etc.
Affine Constraints for Fusibility

- Two statements can be fused if their timestamp can overlap

**Definition (Generalized fusibility check)**

Given $v_R$ (resp. $v_S$) the set of vertices of $D_R$ (resp. $D_S$). $R$ and $S$ are fusible at level $p$ if, $\forall k \in \{1 \ldots p\}$, there exist two semantics-preserving schedules $\Theta^R_k$ and $\Theta^S_k$ such that

$$\exists (\vec{x}_1, \vec{x}_2, \vec{x}_3) \in v_R \times v_S \times v_R, \quad \Theta^R_k(\vec{x}_1) \leq \Theta^S_k(\vec{x}_2) \leq \Theta^R_k(\vec{x}_3)$$

- Intersect $\mathcal{L}$ with fusibility and distribution constraints
- **Completeness**: if the test fails, then there is no sequence of affine transformations that can implement this fusion structure
Fusion / Distribution / Code Motion

Our strategy:

1. Build a set containing all unique fusion / distribution / code motion combinations
2. Prune all combinations that do not preserve the semantics

Given two statements R and S, three choices:

1. R is \textit{fully before} S $\rightarrow$ distribution + code motion
2. R is \textit{fully after} S $\rightarrow$ distribution + code motion
3. otherwise $\rightarrow$ fusion

$\Rightarrow$ It corresponds to all total preorders of R and S
Affine Encoding of Total Preorders

Principle:

► Model a total preorder with 3 binary variables

\[ p_{i,j} : i < j \quad s_{i,j} : i > j \quad e_{i,j} : i = j \]

► Enforce totality and mutual exclusion

► Enforce all cases of transitivity through affine inequalities connecting some variables. Ex: \( e_{i,j} = 1 \land e_{j,k} = 1 \Rightarrow e_{i,k} = 1 \)
Affine Encoding of Total Preorders

Principle:

- Model a total preorder with 3 binary variables
  
  \( p_{i,j}: i < j \quad s_{i,j}: i > j \quad e_{i,j}: i = j \)

- Enforce totality and mutual exclusion

- Enforce all cases of transitivity through affine inequalities connecting some variables. Ex: \( e_{i,j} = 1 \land e_{j,k} = 1 \Rightarrow e_{i,k} = 1 \)

- This set contains one and only one point per distinct total preorder of \( n \) elements
Affine Encoding of Total Preorders

Principle:

- Model a total preorder with 3 binary variables
  \[ p_{i,j} : i < j \quad s_{i,j} : i > j \quad e_{i,j} : i = j \]
- Enforce totality and mutual exclusion
- Enforce all cases of transitivity through affine inequalities connecting some variables. Ex: \( e_{i,j} = 1 \land e_{j,k} = 1 \Rightarrow e_{i,k} = 1 \)

- This set contains one and only one point per distinct total preorder of \( n \) elements
- Easy pruning: just bound the sum of some variables
  \[ \text{e.g., } e_{1,2} + e_{4,5} + e_{8,12} < 3 \]
- Automatic removal of supersets of unfusable sets
Convex set of All Unique Total Preorders

\[ O = \begin{cases} 
0 \leq p_{i,j} \leq 1 \\
0 \leq e_{i,j} \leq 1 \\
0 \leq s_{i,j} \leq 1 
\end{cases} \]

constrained to:

\[ O = \begin{cases} 
0 \leq p_{i,j} \leq 1 \\
0 \leq e_{i,j} \leq 1 \\
p_{i,j} + e_{i,j} \leq 1 \\
\forall k \in ]j,n[ \quad e_{i,j} + e_{i,k} \leq 1 + e_{j,k} \\
\forall k \in ]i,j[ \quad p_{i,k} + p_{k,j} \leq 1 + p_{i,j} \\
\forall k \in ]j,n[ \quad e_{i,j} + p_{i,k} \leq 1 + p_{j,k} \\
\forall k \in ]i,j[ \quad e_{i,j} + p_{j,k} \leq 1 + p_{i,k} \\
\forall k \in ]j,n[ \quad e_{k,j} + p_{i,k} \leq 1 + p_{i,j} \\
\forall k \in ]j,n[ \quad e_{i,j} + p_{i,j} + p_{j,k} \leq 1 + p_{i,k} + e_{i,k} 
\end{cases} \]

- Variables are binary
- Relaxed mutual exclusion
- Basic transitivity on \( e \)
- Basic transitivity on \( p \)
- Complex transitivity on \( p \) and \( e \)
- Complex transitivity on \( s \) and \( p \)

▶ Systematic construction for a given \( n \), needs \( n^2 \) Boolean variables
▶ **Enable ILP modeling, enumeration, etc.**
▶ Extension to multidimensional total preorders (i.e., multi-level fusion)
Pruning for Semantics Preservation

Intuition: enumerate the smallest sets of unfusible statements

- Use an intermediate structure to represent sets of statements
  - Graph representation of maybe-unfusible sets (1 node per statement)
  - Enumerate sets from the smallest to the largest

- Leverage dependence graph + properties of fusion / distribution

- Compute properties by intersecting $L$ with additional fusion / distribution / code motion affine constraints

- Any individual point can be removed from $O$
Space of Semantics-Preserving Fusion Choices

1 point $\leftrightarrow$ 1 unique semantically equivalent program (up to statement reordering)
Space of Semantics-Preserving Fusion Choices

1 point $\leftrightarrow$ \textbf{many} unique semantically equivalent programs (up to iteration reordering)
Space of Semantics-Preserving Fusion Choices

1 point $\leftrightarrow$ 1 unique semantically equivalent program (up to limited iteration reordering)
Objectives for Effective Optimization

Objectives:

► Achieve efficient coarse-grain parallelization

► Combine iterative search of profitable transformations for tiling
  → loop fusion and loop distribution

Tiling Hyperplane method [Bondhugula,08]

► Model-driven approach for automatic parallelization + locality improvement

► Tiling-oriented

► Poor model-driven heuristic for the selection of loop fusion (not portable)

► Overly relaxed definition of fused statements
Fusibility Restricted to Non-negative Schedules

- Fusibility is not a transitive relation!
  - Example: sequence of matrix-by-vector products $x = Ab$, $y = Bx$, $z = Cy$
  - $x = Ab$, $y = Bx$ can be fused, also $y = Bx$, $z = Cy$
  - They cannot be fused all together

- Determining the Fusibility of a group of statements is reducible to exhibiting compatible pairwise loop permutations
  - Extremely easy to compute all possible loop permutations that lead to fuse a pair of statements
  - Never check $\mathcal{L}$ on more than two statements!

- Stronger definition of fusion
  - Guarantee at most $c$ instances are not fused
    \[-c < \Theta^R_k(\vec{0}) - \Theta^S_k(\vec{0}) < c\]
  - No combinatorial choice
The Optimization Algorithm in a Nutshell

Proceeds from the outer-most loop level to the inner-most:

1. Compute the space of valid fusion/distribution/code motion choices

2. Select a fusion/distribution/code motion scheme in this space

3. Compute an affine schedule that implements this scheme
   - Static cost model to select the schedule
   - Compound of skewing, shifting, fusion, distribution, interchange, tiling and parallelization (OpenMP)
   - Maximize locality for each set of statements to be fused
### Experimental Results

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#loops</th>
<th>#stmts</th>
<th>#refs</th>
<th>#dim</th>
<th>#cst</th>
<th>#points</th>
<th>#dim</th>
<th>#cst</th>
<th>#points</th>
<th>Time</th>
<th>perf-Intel</th>
<th>perf-AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>advect3d</td>
<td>12</td>
<td>4</td>
<td>32</td>
<td>12</td>
<td>58</td>
<td>75</td>
<td>9</td>
<td>43</td>
<td>26</td>
<td>0.82s</td>
<td>1.47×</td>
<td>5.19×</td>
</tr>
<tr>
<td>atax</td>
<td>4</td>
<td>4</td>
<td>10</td>
<td>12</td>
<td>58</td>
<td>75</td>
<td>6</td>
<td>25</td>
<td>16</td>
<td>0.06s</td>
<td>3.66×</td>
<td>1.88×</td>
</tr>
<tr>
<td>bicg</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>12</td>
<td>58</td>
<td>75</td>
<td>10</td>
<td>52</td>
<td>26</td>
<td>0.05s</td>
<td>1.75×</td>
<td>1.40×</td>
</tr>
<tr>
<td>gemver</td>
<td>7</td>
<td>4</td>
<td>19</td>
<td>12</td>
<td>58</td>
<td>75</td>
<td>6</td>
<td>28</td>
<td>8</td>
<td>0.06s</td>
<td>1.34×</td>
<td>1.33×</td>
</tr>
<tr>
<td>ludcmp</td>
<td>9</td>
<td>14</td>
<td>35</td>
<td>182</td>
<td>3003</td>
<td>≈10^{12}</td>
<td>40</td>
<td>443</td>
<td>8</td>
<td>0.54s</td>
<td>1.98×</td>
<td>1.45×</td>
</tr>
<tr>
<td>doitgen</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>22</td>
<td>13</td>
<td>3</td>
<td>10</td>
<td>4</td>
<td>0.08s</td>
<td>15.35×</td>
<td>14.27×</td>
</tr>
<tr>
<td>varcovar</td>
<td>7</td>
<td>7</td>
<td>26</td>
<td>42</td>
<td>350</td>
<td>47293</td>
<td>22</td>
<td>193</td>
<td>96</td>
<td>0.09s</td>
<td>7.24×</td>
<td>14.83×</td>
</tr>
<tr>
<td>correl</td>
<td>5</td>
<td>6</td>
<td>12</td>
<td>30</td>
<td>215</td>
<td>4683</td>
<td>21</td>
<td>162</td>
<td>176</td>
<td>0.09s</td>
<td>3.00×</td>
<td>3.44×</td>
</tr>
</tbody>
</table>

**Table:** Search space statistics and performance improvement

- **Performance portability:** empirical search on the target machine of the optimal fusion structure
- Outperforms state-of-the-art cost models
- Full implementation in the source-to-source polyhedral compiler PoCC
Conclusion

Take-home message:

⇒ Clear formalization of loop fusion in the polyhedral model
⇒ Formal definition of all semantically equivalent programs up to:
  ▶ statement reordering
  ▶ limited affine iteration reordering
  ▶ arbitrary affine iteration reordering

⇒ Effective and portable hybrid empirical optimization algorithm
  (parallelization + data locality)

Future work:

▶ Develop static cost models for fusion / distribution / code motion
▶ Use statistical techniques to learn optimization algorithms