When Iterative Optimization Meets the Polyhedral Model: One-Dimensional Date

Louis-Noël Pouchet

ALCHEMY, LRI - INRIA Futurs
Under the direction of A. Cohen & C. Bastoul

October 9, 2006
Problematic

- Emerging microprocessors introduce more parallelism / deeper memory hierarchies
- Optimizing compilers are mandatory to take advantage of processor architecture

But:
- Processor mechanism is too complex to be modeled entirely
- Cost models for optimization phases are too restrictive

⇒ How can we override these difficulties?
Problematic

- Emerging microprocessors introduce more parallelism / deeper memory hierarchies
- Optimizing compilers are mandatory to take advantage of processor architecture

But:
- Processor mechanism is too complex to be modeled entirely
- Cost models for optimization phases are too restrictive

⇒ How can we override these difficulties?
Problematic

- Emerging microprocessors introduce more parallelism / deeper memory hierarchies
- Optimizing compilers are mandatory to take advantage of processor architecture

But:
- Processor mechanism is too complex to be modeled entirely
- Cost models for optimization phases are too restrictive

⇒ How can we override these difficulties?
Iterative Optimization

- Program transformations can result in unpredictable performance degradation (Bodin et al., 98)

⇒ Instead of statically decide if a transformation is better, run it on the target architecture

Pros:
- Much more accurate than static optimization
- Provide performance improvements
- Enable machine learning techniques to discover accurate transformation parameters (Stephenson et al., 03)
- Optimization space search can be feedback-directed
Iterative Optimization

Program transformations can result in unpredictable performance degradation (Bodin et al., 98)

⇒ Instead of statically decide if a transformation is better, run it on the target architecture

Pros:

• Much more accurate than static optimization
• Provide performance improvements
• Enable machine learning techniques to discover accurate transformation parameters (Stephenson et al., 03)
• Optimization space search can be feedback-directed
Iterative Optimization

- Program transformations can result in unpredictable performance degradation (Bodin et al., 98)
  ⇒ Instead of statically decide if a transformation is better, run it on the target architecture

Pros:
- Much more accurate than static optimization
- Provide performance improvements
- Enable machine learning techniques to discover accurate transformation parameters (Stephenson et al., 03)
- Optimization space search can be feedback-directed
Drawbacks

Limitations:

- The set of combination of transformations is extremely large
- Only a subset of them respects the program semantic

→ Only a (very small) subset of transformation sequences is actually tested

→ The search space is too restrictive or too large due to the bottleneck of the legality condition

⇒ Can we improve the search space construction: model all sequences of transformations, and model only legal ones?
**Drawbacks**

Limitations:

- The set of combination of transformations is extremely large
- Only a subset of them respects the program semantic

→ Only a (very small) subset of transformation sequences is actually tested

→ The search space is too restrictive or too large due to the bottleneck of the legality condition

⇒ Can we improve the search space construction: model all sequences of transformations, and model only legal ones?
Drawbacks

Limitations:

- The set of combination of transformations is extremely large
- Only a subset of them respects the program semantic

→ Only a (very small) subset of transformation sequences is actually tested

→ The search space is too restrictive or too large due to the bottleneck of the legality condition

⇒ Can we improve the search space construction: model all sequences of transformations, and model only legal ones?
Drawbacks

Limitations:

- The set of combination of transformations is extremely large
- Only a subset of them respects the program semantic

→ Only a (very small) subset of transformation sequences is actually tested

→ The search space is too restrictive or too large due to the bottleneck of the legality condition

⇒ Can we improve the search space construction: model all sequences of transformations, and model only legal ones?
Iterative Optimization in the Polyhedral Model

- Focus on a subclass of programs: Static Control Parts
- Use a polyhedral abstraction to represent program information
- Use iterative optimization techniques in the constructed space

In the polyhedral model (Feautrier, 92):
- Composition of transformations are easily expressed
- Transformation legality is easily checked
- Natural expression of parallelism
Iterative Optimization in the Polyhedral Model

- Focus on a subclass of programs: Static Control Parts
- Use a polyhedral abstraction to represent program information
- Use iterative optimization techniques in the constructed space

→ In the polyhedral model (Feautrier, 92):
  - Composition of transformations are easily expressed
  - Transformation legality is easily checked
  - Natural expression of parallelism
Iterative Optimization in the Polyhedral Model

- Focus on a subclass of programs: Static Control Parts
- Use a polyhedral abstraction to represent program information
- Use iterative optimization techniques in the constructed space

In the polyhedral model (Feautrier, 92):
- Composition of transformations are easily expressed
- Transformation legality is easily checked
- Natural expression of parallelism
Iterative Optimization in the Polyhedral Model

- Focus on a subclass of programs: Static Control Parts
- Use a polyhedral abstraction to represent program information
- Use iterative optimization techniques in the constructed space

→ In the polyhedral model (Feautrier, 92):
  - Composition of transformations are easily expressed
  - Transformation legality is easily checked
  - Natural expression of parallelism
The Polyhedral Model

1 Analysis: from code to model

2 Transformation in the model
Here: $\theta(i, j) = t = i + j$

3 Code generation: from model to code

---

October 9, 2006
The Polyhedral Model

1 Analysis: from code to model

2 Transformation in the model
Here: $\theta(j) = t = i + j$

3 Code generation: from model to code

October 9, 2006
The Polyhedral Model

1 Analysis: from code to model

2 Transformation in the model
Here: $\theta(i_j) = t = i + j$

3 Code generation: from model to code

doi = 1, 3
   | doi = 1, 3
   | A(i+j) = ...

dot = 2, 6
   | doi = max(1,t-3), min(t-1,3)
   | A(t) = ...
The Polyhedral Model

1 Analysis: from code to model

2 Transformation in the model
   Here: $\theta(i) = t = i + j$

3 Code generation: from model to code

---

1. The Polyhedral Model

   do i = 1, 3
     do j = 1, 3
       A(i+j) = ...

   1 2 3
   1 2 3 4 5 6
   i
   j

   Transformation in the model

   Here: $\theta(i) = t = i + j$

   do t = 2, 6
     do i = max(1,t-3), min(t-1,3)
       A(t) = ...

   1 2 3
   1 2 3
   2 3 4 5 6
   i
   j
   t
A First Example

**matvect**

```
do i = 0, n
  s(i) = 0
  do j = 0, n
    s(i) = s(i) + a(i,j) * x(j)
  end do
end do
```

Iteration domain of $R$:

- **iteration vector** $\vec{x}_R = (i)$
- $D_R : \{i | 0 \leq i \leq n\}$
- $D_R : \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \cdot (i) + \begin{pmatrix} 0 \\ n \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} i \\ n \end{pmatrix} \geq \vec{0}$
A First Example

**matvect**

\[
\text{do } i = 0, n \\
\quad R \quad s(i) = 0 \\
\quad \quad \text{do } j = 0, n \\
\quad \quad \quad S \quad s(i) = s(i) + a(i,j) \times x(j) \\
\quad \quad \text{end do} \\
\text{end do}
\]

Iteration domain of \( R \):

- *iteration vector* \( \vec{x}_R = (i) \)
- \( \mathcal{D}_R : \{ i \mid 0 \leq i \leq n \} \)
- \( \mathcal{D}_R : \begin{bmatrix} 1 & \end{bmatrix} \cdot (i) + \begin{bmatrix} 0 \\ n \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} i \\ n \end{bmatrix} \geq \vec{0} \)
A First Example

**matvect**

```plaintext
do i = 0, n
R  |  s(i) = 0
   |  do j = 0, n
S  |  s(i) = s(i) + a(i,j) * x(j)
   |  end do
end do
```

Iteration domain of $S$:

- *iteration vector* $\vec{x}_S = (i)$
- $\mathcal{D}_S : \{i, j \mid 0 \leq i \leq n, 0 \leq j \leq n, \}$
- $\mathcal{D}_S : \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix} \cdot \begin{pmatrix} i \\ j \\ n \end{pmatrix} \geq \vec{0}$
## Expressing Transformations

### Interchange Transformation

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

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

transformation function 
\[\vec{y} = T_1 \vec{x}\]

**Do**
- \(i = 1, 2\)
- \(j = 1, 3\)

**Do**
- \(j = 1, 3\)
- \(i = 1, 2\)
Expressing Transformations

**Interchange Transformation**

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

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

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

transformation function

\[
\vec{y} = T_i \vec{x}
\]

```
for i = 1, 2
for j = 1, 3
```

```
for j = 1, 3
for i = 1, 2
```
Expressing Transformations

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

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

transformation function 
\[
\tilde{y} = T_2 \tilde{x}
\]

\[
\begin{array}{l}
do \ i = 1, \ 2 \\
do \ j = 1, \ 3
\end{array}
\]

\[
\begin{array}{l}
do \ i = -1, \ -2, \ -1 \\
do \ j = 1, \ 3
\end{array}
\]
Expressing Transformations

Compound Transformation

The transformation matrix is the composition of an interchange and reversal

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

transformation function

\[
y = T \bar{x} = T_1 T_2 \bar{x}
\]

October 9, 2006
Expressing Transformations

Compound Transformation

The transformation matrix is the composition of an interchange and reversal

\[
\begin{pmatrix}
-1 & 0 \\
0 & 1 \\
0 & -1
\end{pmatrix}
\begin{pmatrix}
i \\
j
\end{pmatrix}
+ \begin{pmatrix}
-1 \\
2 \\
-1
\end{pmatrix}
\geq \begin{pmatrix}
0 \\
-1 \\
0
\end{pmatrix}
\begin{pmatrix}
i \\
j
\end{pmatrix}
\Rightarrow

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

(a) original polyhedron
\[A\bar{x} + \bar{a} \geq \bar{0}\]
(b) transformation function
\[
\bar{y} = T\bar{x} = T_1 T_2 \bar{x}
\]
(c) target polyhedron
\[(AT^{-1})\bar{y} + \bar{a} \geq \bar{0}\]

\begin{align*}
do & i = 1, 2 \\
do & j = 1, 3
\end{align*}

\begin{align*}
do & j = -1, -3, -1 \\
do & i = 1, 2
\end{align*}
Scheduling a Program

**Definition (Schedule)**

A schedule of a program is a function which associates a timestamp to each instance of each instruction. It can be written, for a statement $S$ ($T$ is a constant matrix):

$$
\theta_S(\vec{x}_S) = T \begin{pmatrix} \vec{x}_S \\ n \\ 1 \end{pmatrix}
$$

Example:

$$
\theta_R(\vec{x}_R) = \begin{bmatrix} 1 \end{bmatrix} \cdot (i)
$$

$$
\theta_S(\vec{x}_S) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot (j)
$$

Is the original lexicographic order for $R$ and $S$. 
Objectives

- Focus on one-dimensional schedules ($T$ is a constant row matrix)
- Build the set of all *legal* program versions (i.e. which respects all the data dependence of the program)

→ Perform an exact dependence analysis
→ Build the set of all possible values of $T$

⇒ The resulting space represents all the distinct possible ways to *legally reschedule* the program, using arbitrarily complex sequence of transformations.
Objectives

- Focus on one-dimensional schedules ($T$ is a constant row matrix)
- Build the set of all legal program versions (i.e. which respects all the data dependence of the program)

→ Perform an exact dependence analysis
→ Build the set of all possible values of $T$

⇒ The resulting space represents all the distinct possible ways to legally reschedule the program, using arbitrarily complex sequence of transformations.
Objectives

- Focus on one-dimensional schedules ($T$ is a constant row matrix)
- Build the set of all *legal* program versions (i.e. which respects all the data dependence of the program)

→ Perform an exact dependence analysis
→ Build the set of all possible values of $T$

⇒ The resulting space represents all the distinct possible ways to *legally reschedule* the program, using arbitrarily complex sequence of transformations.
Dependence Expression

- Need to represent the *exact* set of instances in dependence
- Exact computation made possible thanks to the SCoP and Static reference assumptions (Bastoul, 04)
- Use a subset of the Cartesian product of iteration domains:

\[
\begin{align*}
R & : \quad \text{do } i = 1, 3 \\
& \quad s(i) = 0 \\
& \quad \text{do } j = 1, 3 \\
& \quad s(i) = s(i) + a(i,j) \cdot x(j)
\end{align*}
\]

\[
D_{R \delta S} : 
\begin{bmatrix}
1 & -1 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & -1 & 1 & 0
\end{bmatrix} \cdot \begin{pmatrix}
i_R \\
i_S \\
i_S \\
n_1
\end{pmatrix} \geq 0
\]
Formal Definition [1/2]

Assuming \( R\delta S \), \( \mathcal{D}_{R\delta S} \) is the exact set of instances of \( R \) and \( S \) where the dependence exists.

A schedule is **legal** iff, \( \forall \overrightarrow{x}_R \times \overrightarrow{x}_S \in \mathcal{D}_{R\delta S}, \theta_R(\overrightarrow{x}_R) < \theta_S(\overrightarrow{x}_S) \).

**Legal Schedule**

\[ \Rightarrow \text{Assuming } R\delta S, \theta_R(\overrightarrow{x}_R) \text{ and } \theta_S(\overrightarrow{x}_S) \text{ are legal iff:} \]

\[ \Delta_{R,S} = \theta_S(\overrightarrow{x}_S) - \theta_R(\overrightarrow{x}_R) - 1 \]

Is non-negative for each point in \( \mathcal{D}_{R\delta S} \).
Formal Definition [2/2]

→ We can express the legality condition as a set of affine non-negative functions over $\mathcal{D}_{R\delta S}$

Lemma (Affine form of Farkas lemma)

Let $\mathcal{D}$ be a nonempty polyhedron defined by the inequalities $A\vec{x} + \vec{b} \geq \vec{0}$. Then any affine function $f(\vec{x})$ is non-negative everywhere in $\mathcal{D}$ iff it is a positive affine combination:

$$f(\vec{x}) = \lambda_0 + \vec{\lambda}^T (A\vec{x} + \vec{b}), \text{ with } \lambda_0 \geq 0 \text{ and } \vec{\lambda} \geq \vec{0}.$$ 

$\lambda_0$ and $\vec{\lambda}^T$ are called the Farkas multipliers.
Formal Definition [2/2]

We can express the legality condition as a set of affine non-negative functions over $\mathcal{D}_{R \delta S}$

Lemma (Affine form of Farkas lemma)

Let $\mathcal{D}$ be a nonempty polyhedron defined by the inequalities $A\vec{x} + \vec{b} \geq \vec{0}$. Then any affine function $f(\vec{x})$ is non-negative everywhere in $\mathcal{D}$ iff it is a positive affine combination:

$$f(\vec{x}) = \lambda_0 + \vec{\lambda}^T (A\vec{x} + \vec{b}), \text{ with } \lambda_0 \geq 0 \text{ and } \vec{\lambda} \geq \vec{0}.$$  

$\lambda_0$ and $\vec{\lambda}^T$ are called the Farkas multipliers.
Formal Definition [2/2]

→ We can express the legality condition as a set of affine non-negative functions over $\mathcal{D}_{R\delta S}$

Lemma (Affine form of Farkas lemma)

Let $\mathcal{D}$ be a nonempty polyhedron defined by the inequalities $A\vec{x} + \vec{b} \geq \vec{0}$. Then any affine function $f(\vec{x})$ is non-negative everywhere in $\mathcal{D}$ iff it is a positive affine combination:

$$f(\vec{x}) = \lambda_0 + \vec{\lambda}^T (A\vec{x} + \vec{b}), \text{ with } \lambda_0 \geq 0 \text{ and } \vec{\lambda} \geq \vec{0}.$$

$\lambda_0$ and $\vec{\lambda}^T$ are called the Farkas multipliers.

⇒ We can express the set of affine, non-negative functions over $\mathcal{D}_{R\delta S}$
Formal Definition [2/2]

Lemma (Affine form of Farkas lemma)

Let $\mathcal{D}$ be a nonempty polyhedron defined by the inequalities $A\vec{x} + \vec{b} \geq \vec{0}$. Then any affine function $f(\vec{x})$ is non-negative everywhere in $\mathcal{D}$ iff it is a positive affine combination:

$$f(\vec{x}) = \lambda_0 + \vec{\lambda}^T (A\vec{x} + \vec{b}), \text{ with } \lambda_0 \geq 0 \text{ and } \vec{\lambda} \geq \vec{0}.$$ 

$\lambda_0$ and $\vec{\lambda}^T$ are called the Farkas multipliers.

$\Rightarrow$ We just need to equate the coefficients:

$$\theta_S(\vec{x}_S) - \theta_R(\vec{x}_R) - 1 = \lambda_0 + \vec{\lambda}^T \left( \mathcal{D}_R \delta_S \left( \vec{x}_R \right) + \vec{d}_R \delta_S \right) \geq 0$$
An example

<table>
<thead>
<tr>
<th>R</th>
<th>s(i) = 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>do j = 1, 3</td>
</tr>
<tr>
<td>S</td>
<td>s(i) = s(i) + a(i, j) * x(j)</td>
</tr>
</tbody>
</table>

The two prototype affine schedules for $R$ and $S$ are:

$$
\theta_R(x_R) = t_{1R} \cdot i_R + t_{2R} \cdot n + t_{3R} \cdot 1
$$

$$
\theta_S(x_S) = t_{1S} \cdot i_S + t_{2S} \cdot j_S + t_{3S} \cdot n + t_{4S} \cdot 1
$$

We get the following system for $R_\delta S$:

$$
\begin{aligned}
D_{R_\delta S} & \\
 i_R & : & -t_{1R} & = \lambda_{D_{1,1}} - \lambda_{D_{1,2}} + \lambda_{D_{1,7}} \\
i_S & : & t_{1S} & = \lambda_{D_{1,3}} - \lambda_{D_{1,4}} - \lambda_{D_{1,7}} \\
j_S & : & t_{2S} & = \lambda_{D_{1,5}} - \lambda_{D_{1,6}} \\
n & : & t_{3S} - t_{2R} & = \lambda_{D_{1,2}} + \lambda_{D_{1,4}} - \lambda_{D_{1,6}} \\
1 & : & t_{4S} - t_{3R} - 1 & = \lambda_{D_{1,0}}
\end{aligned}
$$

We need to solve this system, to get $D^R_t S$. 

October 9, 2006
An example

<table>
<thead>
<tr>
<th>do i = 1, 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
</tr>
<tr>
<td>s(i) = 0</td>
</tr>
<tr>
<td>do j = 1, 3</td>
</tr>
<tr>
<td>S</td>
</tr>
<tr>
<td>s(i) = s(i) + a(i,j) * x(j)</td>
</tr>
</tbody>
</table>

The two prototype affine schedules for $R$ and $S$ are:

\[
\begin{align*}
\theta_R(x_R) &= t_{1R}i_R + t_{2R}n + t_{3R}1 \\
\theta_S(x_S) &= t_{1S}i_S + t_{2S}j_S + t_{3S}n + t_{4S}1
\end{align*}
\]

We get the following system for $R\delta S$:

\[
\begin{align*}
D_{R\delta S} & \quad i_R : & -t_{1R} &= \lambda_{D_{1,1}} - \lambda_{D_{1,2}} + \lambda_{D_{1,7}} \\
& \quad i_S : & t_{1S} &= \lambda_{D_{1,3}} - \lambda_{D_{1,4}} - \lambda_{D_{1,7}} \\
& \quad j_S : & t_{2S} &= \lambda_{D_{1,5}} - \lambda_{D_{1,6}} \\
& \quad n : & t_{3S} - t_{2R} &= \lambda_{D_{1,2}} + \lambda_{D_{1,4}} + \lambda_{D_{1,6}} \\
& \quad 1 : & t_{4S} - t_{3R} - 1 &= \lambda_{D_{1,0}}
\end{align*}
\]

\[\rightarrow\] We need to solve this system, to get $D_t^{R\delta S}$. 

October 9, 2006
Construction Algorithm

- Need to build the intersection of all constraints obtained for each dependence, so for $k$ dependences:

$$\mathcal{D}_t = \bigcap_{k} \mathcal{D}_t^k$$

- Need to bound the space, since the set of possible transformations can be infinite.

⇒ To each (integral) point in $\mathcal{D}_t$ corresponds a different version of the original program where the semantic is preserved.
Construction Algorithm

- Need to build the intersection of all constraints obtained for each dependence, so for $k$ dependences:

$$D_t = \bigcap_{k} D_t^k$$

- Need to bound the space, since the set of possible transformations can be infinite

⇒ To each (integral) point in $D_t$ corresponds a different version of the original program where the semantic is preserved.
Construction Algorithm

- Need to build the intersection of all constraints obtained for each dependence, so for $k$ dependences:

$$D_t = \bigcap_{k} D_t^k$$

- Need to bound the space, since the set of possible transformations can be infinite

$\Rightarrow$ To each (integral) point in $D_t$ corresponds a different version of the original program where the semantic is preserved.
## Discussions

- **Expression of the set of all legal, arbitrarily long sequences of transformation** (reversal, skewing, interchange, peeling, shifting, fusion, distribution)
- Multiple orders of magnitude reduction in the size of the search space compared to state-of-the-art techniques
- On small kernels, the search space is small enough to be exhaustively computed, yielding a method to find **The best transformation** within the model

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#Dep</th>
<th>#St</th>
<th>Bounds</th>
<th>#Sched</th>
<th>#Legal</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>matvect</td>
<td>5</td>
<td>2</td>
<td>−1, 1</td>
<td>3⁴</td>
<td>129</td>
<td>0.024</td>
</tr>
<tr>
<td>locality</td>
<td>2</td>
<td>2</td>
<td>−1, 1</td>
<td>3¹⁰</td>
<td>6561</td>
<td>0.022</td>
</tr>
<tr>
<td>matmul</td>
<td>7</td>
<td>2</td>
<td>−1, 1</td>
<td>3⁹</td>
<td>912</td>
<td>0.029</td>
</tr>
<tr>
<td>gauss</td>
<td>18</td>
<td>2</td>
<td>−1, 1</td>
<td>3¹⁰</td>
<td>506</td>
<td>0.047</td>
</tr>
<tr>
<td>crout</td>
<td>26</td>
<td>4</td>
<td>−3, 3</td>
<td>7¹⁷</td>
<td>798</td>
<td>0.046</td>
</tr>
</tbody>
</table>
Discussions

- Expression of the set of all legal, arbitrarily long sequences of transformation (reversal, skewing, interchange, peeling, shifting, fusion, distribution)
- Multiple orders of magnitude reduction in the size of the search space compared to state-of-the-art techniques
- On small kernels, the search space is small enough to be exhaustively computed, yielding a method to find the best transformation within the model

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#Dep</th>
<th>#St</th>
<th>Bounds</th>
<th>#Sched</th>
<th>#Legal</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>matvect</td>
<td>5</td>
<td>2</td>
<td>−1, 1</td>
<td>3'</td>
<td>129</td>
<td>0.024</td>
</tr>
<tr>
<td>locality</td>
<td>2</td>
<td>2</td>
<td>−1, 1</td>
<td>3^{10}</td>
<td>6561</td>
<td>0.022</td>
</tr>
<tr>
<td>matmul</td>
<td>7</td>
<td>2</td>
<td>−1, 1</td>
<td>3^{9}</td>
<td>912</td>
<td>0.029</td>
</tr>
<tr>
<td>gauss</td>
<td>18</td>
<td>2</td>
<td>−1, 1</td>
<td>3^{10}</td>
<td>506</td>
<td>0.047</td>
</tr>
<tr>
<td>crout</td>
<td>26</td>
<td>4</td>
<td>−3, 3</td>
<td>7^{17}</td>
<td>798</td>
<td>0.046</td>
</tr>
</tbody>
</table>
Discussions

- Expression of the set of all legal, arbitrarily long sequences of transformation (reversal, skewing, interchange, peeling, shifting, fusion, distribution)
- Multiple orders of magnitude reduction in the size of the search space compared to state-of-the-art techniques
- On small kernels, the search space is small enough to be exhaustively computed, yielding a method to find The best transformation within the model

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#Dep</th>
<th>#St</th>
<th>Bounds</th>
<th>#Sched</th>
<th>#Legal</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>matvect</td>
<td>5</td>
<td>2</td>
<td>−1, 1</td>
<td>3'</td>
<td>129</td>
<td>0.024</td>
</tr>
<tr>
<td>locality</td>
<td>2</td>
<td>2</td>
<td>−1, 1</td>
<td>3^{10}</td>
<td>6561</td>
<td>0.022</td>
</tr>
<tr>
<td>matmul</td>
<td>7</td>
<td>2</td>
<td>−1, 1</td>
<td>3^{9}</td>
<td>912</td>
<td>0.029</td>
</tr>
<tr>
<td>gauss</td>
<td>18</td>
<td>2</td>
<td>−1, 1</td>
<td>3^{10}</td>
<td>506</td>
<td>0.047</td>
</tr>
<tr>
<td>crout</td>
<td>26</td>
<td>4</td>
<td>−3, 3</td>
<td>7^{17}</td>
<td>798</td>
<td>0.046</td>
</tr>
</tbody>
</table>

October 9, 2006
Performance Distribution [1/2]

Figure: Performance distribution for matmul, locality, mvt and crout
Performance Distribution [2/2]

- Regularities are observable
- Exhaustive scan may achievable on (very) small kernels
- High peak performance discovered thanks to optimization enabling
- The best transformation depends on the compiler, the target architecture, and even the compiler options

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Compiler</th>
<th>Options</th>
<th>Parameters</th>
<th>#Improved</th>
<th>ID best</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>h264</td>
<td>PathCC</td>
<td>-O2</td>
<td>none</td>
<td>11</td>
<td>352</td>
<td>36.1%</td>
</tr>
<tr>
<td>h264</td>
<td>GCC</td>
<td>-O3</td>
<td>none</td>
<td>19</td>
<td>234</td>
<td>13.3%</td>
</tr>
<tr>
<td>h264</td>
<td>ICC</td>
<td>-O2</td>
<td>none</td>
<td>27</td>
<td>290</td>
<td>12.9%</td>
</tr>
<tr>
<td>h264</td>
<td>ICC</td>
<td>-O2</td>
<td>none</td>
<td>27</td>
<td>290</td>
<td>12.9%</td>
</tr>
<tr>
<td>h264</td>
<td>GCC</td>
<td>-O2</td>
<td>none</td>
<td>19</td>
<td>234</td>
<td>13.3%</td>
</tr>
<tr>
<td>h264</td>
<td>ICC</td>
<td>-O2</td>
<td>none</td>
<td>27</td>
<td>290</td>
<td>12.9%</td>
</tr>
<tr>
<td>MVT</td>
<td>PathCC</td>
<td>-O2</td>
<td>N=2000</td>
<td>5652</td>
<td>4934</td>
<td>27.4%</td>
</tr>
<tr>
<td>MVT</td>
<td>GCC</td>
<td>-O2</td>
<td>N=2000</td>
<td>3526</td>
<td>13301</td>
<td>18.0%</td>
</tr>
<tr>
<td>MVT</td>
<td>GCC</td>
<td>-O2</td>
<td>N=2000</td>
<td>3526</td>
<td>13301</td>
<td>18.0%</td>
</tr>
<tr>
<td>MVT</td>
<td>ICC</td>
<td>-O2</td>
<td>N=2000</td>
<td>5826</td>
<td>14093</td>
<td>24.0%</td>
</tr>
<tr>
<td>MVT</td>
<td>ICC</td>
<td>-O2</td>
<td>N=2000</td>
<td>5826</td>
<td>14093</td>
<td>24.0%</td>
</tr>
<tr>
<td>matmul</td>
<td>PathCC</td>
<td>-O2</td>
<td>N=250</td>
<td>402</td>
<td>283</td>
<td>308.1%</td>
</tr>
<tr>
<td>matmul</td>
<td>GCC</td>
<td>-O2</td>
<td>N=250</td>
<td>318</td>
<td>284</td>
<td>38.6%</td>
</tr>
<tr>
<td>matmul</td>
<td>GCC</td>
<td>-O2</td>
<td>N=250</td>
<td>318</td>
<td>284</td>
<td>38.6%</td>
</tr>
<tr>
<td>matmul</td>
<td>ICC</td>
<td>-O2</td>
<td>N=250</td>
<td>390</td>
<td>311</td>
<td>56.6%</td>
</tr>
<tr>
<td>matmul</td>
<td>ICC</td>
<td>-O2</td>
<td>N=250</td>
<td>390</td>
<td>311</td>
<td>56.6%</td>
</tr>
</tbody>
</table>

October 9, 2006
Exhaustive vs Heuristic Scan

Propose a decoupling heuristic:

- The general “form” of the schedule is embedded in the iterator coefficients
- Parameters and constant coefficients can be seen as a refinement

→ On some distributions a random heuristic may converge faster

**Figure:** Heuristic convergence

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#Schedules</th>
<th>Heuristic.</th>
<th>#Runs</th>
<th>%Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>locality</td>
<td>6561</td>
<td>Rand, DH</td>
<td>125, 123</td>
<td>96.1%, 98.3%</td>
</tr>
<tr>
<td>matmul</td>
<td>912</td>
<td>Rand, DH</td>
<td>170, 170</td>
<td>99.9%, 99.8%</td>
</tr>
<tr>
<td>mvt</td>
<td>16641</td>
<td>Rand, DH</td>
<td>30, 31</td>
<td>93.3%, 99.0%</td>
</tr>
</tbody>
</table>
**Internship Summary: Internship Overview**

**What, When, with Who?**

- **Constant talks with Nicolas Vasilache (PhD student)**
- **Advised and oriented by Cedric Bastoul**
- **Theoretical fruitful discussions with Albert Cohen**

October 9, 2006
Scientific Contribution

- New approach of the search space for iterative optimization
- Mathematically well founded algorithm for the construction of the *legal* transformation space in the polyhedral model
- Better formulation of the Fourier-Motzkin algorithm

- First exhaustive exploration of the performance space in the polyhedral model, for one-dimensional schedules
- Usual mathematical models sub-optimality brought to light
- Many observations on the performance space distribution
Scientific Contribution

- New approach of the search space for iterative optimization
- Mathematically well founded algorithm for the construction of the *legal* transformation space in the polyhedral model
- Better formulation of the Fourier-Motzkin algorithm

- First exhaustive exploration of the performance space in the polyhedral model, for one-dimensional schedules
- Usual mathematical models sub-optimality brought to light
- Many observations on the performance space distribution
Ongoing and Future Work

Ongoing research:
- Expression of equivalence between parts of the search space
- Simulation of multidimensional schedules with correction / completion
- New exploration heuristics
- Feedback directed exploration

PhD objectives:
- Extend the method to multidimensional schedules
- Develop exploration methods for the search space (statistic, machine learning, . . .)
Conclusion

- Very exciting and fruitful internship
- Many applications and collaborative works will be issued
- Novel iterative compilation method

⇒ The polyhedral model contributes to accelerate the convergence of iterative methods and to discover significant opportunities for performance improvements.
**Conclusion**

- Very exciting and fruitful internship
- Many applications and collaborative works will be issued
- Novel iterative compilation method

⇒ The polyhedral model contributes to accelerate the convergence of iterative methods and to discover significant opportunities for performance improvements.
Conclusion

- Very exciting and fruitful internship
- Many applications and collaborative works will be issued
- Novel iterative compilation method

⇒ The polyhedral model contributes to accelerate the convergence of iterative methods and to discover significant opportunities for performance improvements.
Conclusion

- Very exciting and fruitful internship
- Many applications and collaborative works will be issued
- Novel iterative compilation method

⇒ The polyhedral model contributes to accelerate the convergence of iterative methods and to discover significant opportunities for performance improvements.
### A Transformation Example

**Optimal Transformation for mvt, GCC 4 -O2**

| S1: x1[i] = 0 | for (i = 0; i <= M; i++) { 
| S2: x2[i] = 0 |   S2(i); 
| S3: x1[i] += a[i][j] * y1[j] |   for (c1 = 1; c1 <= M-1; c1++) 
| S4: x2[i] += a[j][i] * y2[j] |     for (i = 0; i <= M; i++) { 
|   for (j = 0; j <= M; j++) { 
|     S3(i,j); 
|     S4(i,j); 
|   } |       S4(i,c1-1); 
| } | } 

| for (i = 0; i <= M; i++) { 
|   S1(i); 
|   S2(i); 
|   for (j = 0; j <= M; j++) { 
|     S3(i,j); 
|     S4(i,j); 
|   } | for (i = 0; i <= M; i++) { 
|   S1(i); 
|   S4(i,M-1); 
| } | for (i = 1 ; i <= M; i++) { 
|   S4(i,M); 
|   for (c1 = M+2; c1 <= 3*M+1; c1++) 
|     for (i = max(c1-2*M-1,0); i <= min(M,c1-M-1); i++) { 
|       S3(i,c1-i-M-1); 
|     } 

---

October 9, 2006