Iterative Optimization in the Polyhedral Model: Part I, One-Dimensional Time

Louis-Noel Pouchet, Cedric Bastoul, Albert Cohen and Nicolas Vasilache

Presented by: Tom St. John

Dept of Computer & Information Sciences
University of Delaware
Introduction

- Focus on Loop Nest Optimization for regular loops
- Automatic method for parallelism extraction/loop transformation
- Combine iterative methods with the polyhedral model
- Solution is independent of the compiler and the target machine
Contribution

Search Space Construction

- One point in the search space maps to one distinct legal program version
- Suitable for various exploration methods

Performance

- 99% of best speedup attained within 20 runs of a dedicated heuristic
- Wall clock optimal transformation discoverable on small kernels
A Motivating Example
One-Dimensional Scheduling

Original Schedule

\[
\begin{align*}
&\text{for } (i=0; i<n; ++i) \{ \\
&\quad \text{. } S1(i); \\
&\quad \text{. for } (j=0; j<n; ++j) \\
&\quad \quad \text{. } S2(i,j); \\
&\& \\left \{ \begin{array}{l}
\theta_{S1} = i \\
\theta_{S2} = i
\end{array} \right.
\end{align*}
\]

- Specify the outer-most loop only
- Initial outer-most loop is \( i \)
Distribute Loops

\[
\text{for } (i=0; i<n; ++i) \{
  . \ S1(i);
  . \text{for } (j=0; j<n; ++j)
  . \ . \ S2(i,j);
\}
\]

\[
\begin{cases}
\theta_{S1} = i \\
\theta_{S2} = i+n
\end{cases}
\]

- Specify the outer-most loop only
- All instances of S1 execute before the first instance of S2
Distribute Loops and Loop Interchange for S2

for (i=0; i<n; ++i) {
  . S1(i);
  . for (j=0; j<n; ++j)
  .  . S2(i,j);
}

\[
\begin{align*}
\theta_{S1} &= i \\
\theta_{S2} &= j + n
\end{align*}
\]

for (i=0; i<n; ++i) {
  . S1(i);
  . for (j=n; j<2*n; ++j)
  .  . S2(i,j-n);
}

- Specify the outer-most loop only
- The outer-most loop for S2 becomes \( j \)
for (i=0; i<n; ++i) {
    . S1(i);
    . for (j=0; j<n; ++j)
    .   S2(i,j);
}

\[
\begin{align*}
\theta_{S1} &= i \\
\theta_{S2} &= j + n
\end{align*}
\]

for (i=0; i<n; ++i) {
    . S1(i);
    . for (j=n; j<2*n; ++j)
    .   S2(i,j-n);
}

<table>
<thead>
<tr>
<th>Transformation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reversal</td>
<td>Changes the direction in which a loop traverses its iteration range</td>
</tr>
<tr>
<td>skewing</td>
<td>Makes the bounds of a given loop depend on an outer loop counter</td>
</tr>
<tr>
<td>interchange</td>
<td>Exchanges two loops in a perfectly nested loop, a.k.a. permutation</td>
</tr>
<tr>
<td>peeling</td>
<td>Extracts one iteration of a given loop</td>
</tr>
<tr>
<td>shifting</td>
<td>Allows to reorder loops</td>
</tr>
<tr>
<td>fusion</td>
<td>Fuses two loops, a.k.a. jamming</td>
</tr>
<tr>
<td>distribution</td>
<td>Splits a single loop nest into many, a.k.a. fission or splitting</td>
</tr>
</tbody>
</table>
for (i=0; i<n; ++i) {
    S1(i);
    for (j=0; j<n; ++j)
        S2(i, j);
}

- A schedule is an affine function of the iteration vector and the parameters
  \[
  \theta_{S1}(\bar{x}_{S1}) = t_{1s1} \cdot i_{s1} + t_{2s1} \cdot n + t_{3s1} \cdot 1
  \]
  \[
  \theta_{S2}(\bar{x}_{S2}) = t_{1s2} \cdot i_{s2} + t_{2s2} \cdot j_{s2} + t_{3s2} \cdot n + t_{4s2} \cdot 1
  \]
A schedule is an affine function of the iteration vector and the parameters:

\[
\begin{align*}
\theta_{S1}(\vec{x}_{S1}) &= t_{1s1} \cdot i_{S1} + t_{2s1} \cdot n + t_{3s1} \cdot 1 \\
\theta_{S2}(\vec{x}_{S2}) &= t_{1s2} \cdot i_{S2} + t_{2s2} \cdot j_{S2} + t_{3s2} \cdot n + t_{4s2} \cdot 1
\end{align*}
\]

For \(-1 \leq t \leq 1\), there are \(3^7 = 2187\) possible schedules.
for (i=0; i<n; ++i) {
  S1(i);
  for (j=0; j<n; ++j)
    S2(i,j);
}

- A schedule is an affine function of the iteration vector and the parameters
  \[
  \theta_{S1}(\vec{x}_{S1}) = t_{1s1} \cdot i_{S1} + t_{2s1} \cdot n + t_{3s1} \cdot 1
  \]
  \[
  \theta_{S2}(\vec{x}_{S2}) = t_{1s2} \cdot i_{S2} + t_{2s2} \cdot j_{S2} + t_{3s2} \cdot n + t_{4s2} \cdot 1
  \]

- For \(-1 \leq t \leq 1\), there are \(3^7 = 2187\) possible schedules
- However, only 129 legal distinct schedules
Overview
Efficiently construct a space of all legal, distinct affine schedules

<table>
<thead>
<tr>
<th></th>
<th>matmult</th>
<th>locality</th>
<th>fir</th>
<th>h264</th>
<th>crout</th>
</tr>
</thead>
<tbody>
<tr>
<td>i-Bounds</td>
<td>−1,1</td>
<td>−1,1</td>
<td>0,1</td>
<td>−1,1</td>
<td>−3,3</td>
</tr>
<tr>
<td>c-Bounds</td>
<td>−1,1</td>
<td>−1,1</td>
<td>0,3</td>
<td>0,4</td>
<td>−3,3</td>
</tr>
<tr>
<td>#Sched.</td>
<td>$1.9 \times 10^4$</td>
<td>$5.9 \times 10^4$</td>
<td>$1.2 \times 10^7$</td>
<td>$1.8 \times 10^8$</td>
<td>$2.6 \times 10^{15}$</td>
</tr>
</tbody>
</table>
• Efficiently construct a space of all legal, distinct affine schedules

<table>
<thead>
<tr>
<th></th>
<th>matmult</th>
<th>locality</th>
<th>fir</th>
<th>h264</th>
<th>crout</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t$-Bounds</td>
<td>-1,1</td>
<td>-1,1</td>
<td>0,1</td>
<td>-1,1</td>
<td>-3,3</td>
</tr>
<tr>
<td>$c$-Bounds</td>
<td>-1,1</td>
<td>-1,1</td>
<td>0,3</td>
<td>0,4</td>
<td>-3,3</td>
</tr>
<tr>
<td>#Sched.</td>
<td>$1.9 \times 10^4$</td>
<td>$5.9 \times 10^4$</td>
<td>$1.2 \times 10^7$</td>
<td>$1.8 \times 10^8$</td>
<td>$2.6 \times 10^{15}$</td>
</tr>
<tr>
<td>#Legal</td>
<td>6561</td>
<td>912</td>
<td>792</td>
<td>360</td>
<td>798</td>
</tr>
</tbody>
</table>
**Search Space Construction**

- Efficiently construct a space of all legal, distinct affine schedules

<table>
<thead>
<tr>
<th></th>
<th>matmult</th>
<th>locality</th>
<th>fir</th>
<th>h264</th>
<th>crout</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{f}$-Bounds</td>
<td>$-1,1$</td>
<td>$-1,1$</td>
<td>$0,1$</td>
<td>$-1,1$</td>
<td>$-3,3$</td>
</tr>
<tr>
<td>$c$-Bounds</td>
<td>$-1,1$</td>
<td>$-1,1$</td>
<td>$0,3$</td>
<td>$0,4$</td>
<td>$-3,3$</td>
</tr>
<tr>
<td>#Sched.</td>
<td>$1.9 \times 10^4$</td>
<td>$5.9 \times 10^4$</td>
<td>$1.2 \times 10^7$</td>
<td>$1.8 \times 10^8$</td>
<td>$2.6 \times 10^{15}$</td>
</tr>
<tr>
<td>#Legal</td>
<td>6561</td>
<td>912</td>
<td>792</td>
<td>360</td>
<td>798</td>
</tr>
</tbody>
</table>

- Rely on polyhedral model and integer linear programming to guarantee completeness and correctness of the space properties

- Search space will encompass unique, distinct compositions of reversal, skewing, interchange, fusion, peeling, shifting, distribution
Search Space Exploration

- Perform exhaustive scan to discover wall clock optimal schedule, and evidence of intricacy of the best transformation
- Build an efficient heuristic to accelerate the space traversal
Search Space Construction
Polyhedral Representation

Static Control Parts (SCoP)

- Loops have affine control only
Static Control Parts (SCoP)

- Loops have affine control only
- Iteration domain – represented as integer polyhedra

\[
D_{S_1} = \begin{bmatrix}
1 & 0 & 0 & -1 \\
-1 & 0 & 1 & 0 \\
0 & 1 & 0 & -1 \\
0 & -1 & 1 & 0 \\
-1 & -1 & 1 & 2
\end{bmatrix} \begin{pmatrix}
i \\
j \\
n \\
1
\end{pmatrix} \geq \bar{0}
\]
Polyhedral Representation

Static Control Parts (SCoP)

- Loops have affine control only
- Iteration domain – represented as integer polyhedra
- Memory accesses – static references, represented as affine functions of $\vec{x}$ and $\vec{p}$

\[
\begin{align*}
  f_s(\vec{x}_{S_2}) &= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{pmatrix} \vec{x}_{S_2} \\ n \\ 1 \end{pmatrix} \\
  f_a(\vec{x}_{S_2}) &= \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{pmatrix} \vec{x}_{S_2} \\ n \\ 1 \end{pmatrix} \\
  f_a(\vec{x}_{S_2}) &= \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{pmatrix} \vec{x}_{S_2} \\ n \\ 1 \end{pmatrix}
\end{align*}
\]

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

CISC 879 - Machine Learning for Solving Systems Problems
Polyhedral Representation

Static Control Parts (SCoP)

- Loops have affine control only
- 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 $D_{S1}$ and $D_{S2}$

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

CISC 879 - Machine Learning for Solving Systems Problems
Polyhedral Representation

Static Control Parts (SCoP)

- Loops have affine control only
- Iteration domain – represented as integer polyhedra
- Memory accesses – static references, represented as affine functions of $\bar{x}_S$ and $\bar{p}$
- Data dependence between S1 and S2 – a subset of the Cartesian product of $D_{S1}$ and $D_{S2}$
- Reduced dependence graph labelled by dependence polyhedra
Property (Causality condition for schedules)

Given \( R \delta S \), \( \theta_R \) and \( \theta_S \) are legal iff for each pair of instances in dependence:

\[
\theta_R(x_R) < \theta_S(x_S)
\]

Equivalently: \( \Delta_{R,S} = \theta_S(x_S) - \theta_R(x_R) - 1 \geq 0 \)
Lemma (Affine form of Farkas lemma)

Let $\mathcal{D}$ be a nonempty polyhedron defined by $A\vec{x} + \vec{b} \geq \vec{0}$. 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.
Search Space Construction

Affine Schedules

- Causality condition
- Farkas Lemma

Valid Farkas Multipliers

Many to one

Legal Distinct Schedules

CISC 879 - Machine Learning for Solving Systems Problems
\[ \theta_S(\vec{x}_S) - \theta_R(\vec{x}_R) - 1 = \lambda_0 + \vec{\lambda}^T \left( D_{R,S}(\vec{x}_R) + \vec{d}_{R,S} \right) \geq 0 \]

\[
\begin{align*}
D_{R\delta S} & : \\
i_R & : \lambda_{D_{1,1}} - \lambda_{D_{1,2}} + \lambda_{D_{1,7}} \\
i_S & : \lambda_{D_{1,3}} - \lambda_{D_{1,4}} - \lambda_{D_{1,7}} \\
j_s & : \lambda_{D_{1,5}} - \lambda_{D_{1,6}} \\
n & : \lambda_{D_{1,2}} + \lambda_{D_{1,4}} + \lambda_{D_{1,6}} \\
n & : \lambda_{D_{1,0}}
\end{align*}
\]
\[
\theta_S(\tilde{x}_S) - \theta_R(\tilde{x}_R) - 1 = \lambda_0 + \tilde{\lambda}^T \left( D_{R,S} \left( \begin{array}{c} \tilde{x}_R \\ \tilde{x}_S \end{array} \right) + \tilde{d}_{R,S} \right) \geq 0
\]

\[
\begin{align*}
D_{R,s} & : & i_R & : & -t_{1_R} & = & \lambda_{D_{1,1}} - \lambda_{D_{1,2}} + \lambda_{D_{1,7}} \\
i_S & : & t_{1_S} & = & \lambda_{D_{1,3}} - \lambda_{D_{1,4}} - \lambda_{D_{1,7}} \\
& : & t_{2_S} & = & \lambda_{D_{1,5}} - \lambda_{D_{1,6}} \\
& : & t_{3_S} - t_{2_R} & = & \lambda_{D_{1,2}} + \lambda_{D_{1,4}} + \lambda_{D_{1,6}} \\
& : & t_{4_S} - t_{3_R} - 1 & = & \lambda_{D_{1,0}}
\end{align*}
\]
- Solve the constraint system
- Use optimized Fourier-Motzkin projection algorithm
  - Reduces redundancy
  - Detects implicit equalities
Search Space Construction

- Affine Schedules
  - Causality condition
  - Farkas Lemma

- Valid Farkas Multipliers
  - Identification
  - Projection

- Valid Transformation Coefficients

- Legal Distinct Schedules

CISC 879 - Machine Learning for Solving Systems Problems
Search Space Construction

- Affine Schedules
  - Causality condition
  - Farkas Lemma

- Valid Farkas Multipliers
  - Identification
  - Projection

- Valid Transformation Coefficients

- Legal Distinct Schedules

- Bijection

- One point in the search space $\leftrightarrow$ one set of legal schedules w.r.t. the dependence
Search Space Construction

Algorithm

- Add constraints obtained for each dependence
- Bound the search space
- Search space – represented by a set of linear constraints on the schedule coefficients (Z-polytope)
- Every integral point in the search space corresponds to a distinct program version where the semantics are preserved

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>$\bar{z}$-Bounds</th>
<th>#Sched</th>
<th>#Legal</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>matmult</td>
<td>$-1, 1$</td>
<td>$1.9 \times 10^4$</td>
<td>912</td>
<td>0.029</td>
</tr>
<tr>
<td>locality</td>
<td>$-1, 1$</td>
<td>$5.9 \times 10^4$</td>
<td>6561</td>
<td>0.022</td>
</tr>
<tr>
<td>fir</td>
<td>$0, 1$</td>
<td>$1.2 \times 10^7$</td>
<td>792</td>
<td>0.047</td>
</tr>
<tr>
<td>h264</td>
<td>$-1, 1$</td>
<td>$1.8 \times 10^8$</td>
<td>360</td>
<td>0.024</td>
</tr>
<tr>
<td>crout</td>
<td>$-3, 3$</td>
<td>$2.6 \times 10^{15}$</td>
<td>798</td>
<td>0.046</td>
</tr>
</tbody>
</table>
Search Space Exploration
• CLooG – [http://cloog.org](http://cloog.org)
• PipLib – [http://piplib.org](http://piplib.org)
• PolyLib - [http://icps.u-strasbg.fr/polylib/](http://icps.u-strasbg.fr/polylib/)
Exhaustive Scan

Performance Distribution (1)

Figure: Performance distribution for matmult and locality

CISC 879 - Machine Learning for Solving Systems Problems
Exhaustive Scan

Performance Distribution (2)

(a) GCC -O3

(b) ICC -fast

Figure: The effect of the compiler

CISC 879 - Machine Learning for Solving Systems Problems
Exhaustive Scan

Performance Comparison

Figure: Best Version vs Original

CISC 879 - Machine Learning for Solving Systems Problems
Propose a decoupling heuristic:

- The general form of the schedule is embedded in the iterator coefficients
- Decouple the schedule
- Parameter and constant coefficients are less critical, but can be used to refine the search

\[ \theta_S(\vec{x}_S) = (\vec{i} \, \vec{p} \, c) \begin{pmatrix} \vec{x}_S \\ \vec{n} \\ 1 \end{pmatrix} \]
Heuristic Scan

Addressing scalability to larger SCoP

- Impose a static or dynamic maximum on the number of runs (limit exploration to the domain)
- Replace the exhaustive enumeration of combinations with a limited set of random trials in the domain
Heuristic Scan

Results

Figure: Comparison between random and decoupling heuristics

CISC 879 - Machine Learning for Solving Systems Problems
Conclusions

- Implemented optimization and transformation framework on top of the compiler
- Achieved promising speedup and fast heuristic convergence
- Optimal transformation can be discovered for small kernels
Ongoing/Future Work

- Combine with state-of-the-art feedback-directed iterative methods
- Part II – Multidimensional Schedules (PLDI 08)
- Integrate into GCC GRAPHITE branch