Sélection automatique d'instructions et ordonnancement d'applications basés sur la programmation par contraintes

Kevin Martin¹, Christophe Wolinski¹,², Krzysztof Kuchcinski³, Antoine Floch¹, and François Charot¹

¹ INRIA Rennes - Bretagne Atlantique, France
² University of Rennes I, Irisa, France
³ Dept. of Computer Science, Lund University, Sweden
• Introduction
• Constraint programming
• Pattern generation
• Pattern selection and scheduling
• Instruction reordering
• Results
• Conclusion
Automatic Instruction-Set Extension

Applications

Software generation

Hardware generation

Goal:
- application speed-up
- power reduction
/* Transformed C code */
void fir(const int x[], const int h[], int y[]) {
  int j;
  int sum;
  for (j = 0; j < 100; j = j + 1) {
    sum = 0;
    for (int i = 0; i < 4; i = i + 1)
      sum += x[i + j] * h[i];
    sum = sum >> 15;
    y[j] = sum;
  }
}
• Introduction

• **Constraint programming**

• Pattern generation

• Pattern selection and scheduling

• Instruction reordering

• Results

• Conclusion
## CP(FD) vs (M)LP

<table>
<thead>
<tr>
<th></th>
<th>CSP</th>
<th>ILP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variables</strong></td>
<td>discret</td>
<td>continue and discret</td>
</tr>
<tr>
<td><strong>Constraints</strong></td>
<td>linear and non-linear, global</td>
<td>linear</td>
</tr>
<tr>
<td><strong>Modeling</strong></td>
<td>flexible, problem structure preserved</td>
<td>linearized model</td>
</tr>
<tr>
<td><strong>Solving</strong></td>
<td>local consistency, Depth-first search</td>
<td>linear relaxation (simplex), Branch and bound</td>
</tr>
<tr>
<td><strong>Search</strong></td>
<td>possible heuristics</td>
<td>standard methods</td>
</tr>
</tbody>
</table>
Constraint programming

Formally, a Constraints Satisfaction Problem (CSP) is defined by a 3-tuple \((V;D;C)\)
Where:
- \(V = x_1; x_2; \ldots; x_n\) is a finite set of finite domain variables (FDV's)
- \(D = D_1; D_2; \ldots; D_n\) is a finite set of domains
- \(C\) is a set of constraints restricting the values that the variables can simultaneously take. In practice, the constraints are defined by equations, inequalities and combinatorial constraints.

Example of a combinatorial constraint: the cumulative constraint
Constraint : Graph Match

Possible Cases of graph matching

Target Graph

Pattern Graph

Target Graph

Target Graph
• Introduction
• Constraint programming
• Pattern generation
• Pattern selection and scheduling
• Instruction reordering
• Results
• Conclusion
Pattern Generation

• What is a pattern ?
• How to generate it ?
What is a pattern?

• A connected graph

Pattern $p$

A match $m$ of pattern $p$ in graph $G$
How to generate patterns?

• Subgraph enumeration
• Generation of computational patterns under architectural and technological constraints
  • Number of inputs
  • Number of outputs
  • Critical path
  • Number of resource
  • ...
• Problem of (sub-)graph matching

• Introduction
• Constraint programming
• Pattern generation
• Pattern selection and scheduling
• Instruction reordering
• Results
• Conclusion
Pattern selection and scheduling – problem

Subject graph

Candidates patterns

<table>
<thead>
<tr>
<th></th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
<th>M5</th>
<th>M6</th>
<th>M7</th>
<th>M8</th>
<th>M9</th>
<th>M10</th>
<th>M11</th>
<th>M12</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n4</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n5</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>n6</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

P1 P2 P3 P4
Pattern selection and graph covering

\[ \forall n \in N : \sum_{m \in \text{matches}_m} m_{\text{sel}} = 1 \]

Execution Time = \[ \sum_{m \in M} m_{\text{sel}} \cdot m_{\text{delay}} \]

\[ m_{\text{sel}} = 1 \Rightarrow \forall e \in m_{\text{in}} : e_{\text{start}} = m_{\text{start}} \land \forall e \in m_{\text{out}} : m_{\text{start}} + m_{\text{delay}} \leq e_{\text{start}} \]
Pattern selection and scheduling: 2 architecture models

Model A

Model B

Control

Merged patterns datapath

Register file

Pattern selection
Clk
Pattern selection and scheduling – match execution time modeling

\[ m_{\text{delay}} = \delta_{\text{in}_m} + \delta_m + \delta_{\text{out}_m} \]
Pattern selection and scheduling – architecture model A

\[ m_{\text{delay}} = \delta_{\text{in}_m} + \delta_m + \delta_{\text{out}_m} \]

where

\[ \delta_{\text{in}_m} = \left( \frac{|\text{pred}(m)|}{\text{in}_\text{PerCycle}} \right) - 1 \]

\[ \delta_{\text{out}_m} = \left( \frac{|\text{last}(m)|}{\text{out}_\text{PerCycle}} \right) - 1 \]

Example for a Nios2:
- \(|\text{pred}(m)| = 2\)
- \(\text{in}_\text{PerCycle} = 2\)
- \(|\text{last}(m)| = 1\)
- \(\text{out}_\text{PerCycle} = 1\)
- \(\delta_m = 1\)

\[ \Rightarrow m_{\text{delay}} = 1 \]
Pattern selection and scheduling – architecture model B

\[ m_{delay} = \delta_{in_m} + \delta_m + \delta_{out_m} \]

\[ \delta_{in_m} = \left[ \frac{IN}{in\_PerCycle} \right] - 1 \]

\[ IN = \sum_{n \in pred1(m)} n_{sel} \]

\[ \delta_{out_m} = \left[ \frac{OUT}{out\_PerCycle} \right] - 1 \]

\[ \forall n \in last(m): \sum_{m \in succ1(n)} m_{sel} > 0 \iff B_n = 1 \]

\[ OUT = \sum_{n \in last(m)} B_n \]
Pattern selection and scheduling – 4-tap FIR filter example

Covered graph

Selected patterns

Kevin Martin - SympA'13
• Introduction
• Constraint programming
• Pattern generation
• Pattern selection and scheduling
• Instruction reordering
• Results
• Conclusion
Instruction reordering

Covered graph

Collapsed graph

Selected patterns

Without reordering

With reordering
Instruction reordering

Nombre de cycles

Latence

\[ \delta_{in_m} \]

\[ \delta_m \]

\[ \delta_{out_m} \]

\[ L_{in_m} \]

\[ d_{in_m/out_m} \]

\[ L_{out_m} \]
• Introduction
• Constraint programming
• Pattern generation
• Pattern selection and scheduling
• Instruction reordering
• Results
• Conclusion
## Results

<table>
<thead>
<tr>
<th>Benchmarks</th>
<th>Nodes</th>
<th>cycles</th>
<th>coef</th>
<th>identified</th>
<th>selected</th>
<th>coverage</th>
<th>cycles</th>
<th>speedup</th>
<th>coef</th>
<th>identified</th>
<th>selected</th>
<th>coverage</th>
<th>cycles</th>
<th>speedup</th>
<th>coef</th>
<th>identified</th>
<th>selected</th>
<th>coverage</th>
<th>cycles</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEG Write BMP Header</td>
<td>34</td>
<td>34</td>
<td>0</td>
<td>6</td>
<td>2</td>
<td>82%</td>
<td>14</td>
<td>2.42</td>
<td>2</td>
<td>82%</td>
<td>14</td>
<td>2.42</td>
<td>0</td>
<td>66</td>
<td>2</td>
<td>88%</td>
<td>12</td>
<td>2.83</td>
<td>3</td>
<td>88%</td>
</tr>
<tr>
<td>JPEG Smooth Downsample</td>
<td>66</td>
<td>78</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>19%</td>
<td>68</td>
<td>1.14</td>
<td>2</td>
<td>19%</td>
<td>68</td>
<td>1.14</td>
<td>0</td>
<td>49</td>
<td>4</td>
<td>95%</td>
<td>44</td>
<td>1.77</td>
<td>4</td>
<td>100%</td>
</tr>
<tr>
<td>JPEG IDCT</td>
<td>250</td>
<td>302</td>
<td>0.5</td>
<td>28</td>
<td>10</td>
<td>76%</td>
<td>214</td>
<td>1.41</td>
<td>10</td>
<td>76%</td>
<td>136</td>
<td>2.25</td>
<td>1.5</td>
<td>254</td>
<td>13</td>
<td>83%</td>
<td>141</td>
<td>2.36</td>
<td>15</td>
<td>89%</td>
</tr>
<tr>
<td>EPIC Collapse</td>
<td>274</td>
<td>287</td>
<td>0</td>
<td>11</td>
<td>8</td>
<td>68%</td>
<td>165</td>
<td>1.74</td>
<td>8</td>
<td>68%</td>
<td>165</td>
<td>1.74</td>
<td>0</td>
<td>111</td>
<td>11</td>
<td>71%</td>
<td>156</td>
<td>1.83</td>
<td>14</td>
<td>71%</td>
</tr>
<tr>
<td>BLOWFISH encrypt</td>
<td>201</td>
<td>169</td>
<td>0.5</td>
<td>11</td>
<td>3</td>
<td>74%</td>
<td>90</td>
<td>1.87</td>
<td>3</td>
<td>74%</td>
<td>90</td>
<td>1.87</td>
<td>0</td>
<td>153</td>
<td>8</td>
<td>90%</td>
<td>81</td>
<td>2.08</td>
<td>7</td>
<td>92%</td>
</tr>
<tr>
<td>SHA transform</td>
<td>53</td>
<td>57</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>64%</td>
<td>28</td>
<td>2.03</td>
<td>3</td>
<td>64%</td>
<td>28</td>
<td>2.03</td>
<td>0</td>
<td>48</td>
<td>8</td>
<td>98%</td>
<td>22</td>
<td>2.59</td>
<td>6</td>
<td>95%</td>
</tr>
<tr>
<td>MESA invert matrix</td>
<td>152</td>
<td>334</td>
<td>0.5</td>
<td>2</td>
<td>2</td>
<td>10%</td>
<td>320</td>
<td>1.04</td>
<td>2</td>
<td>10%</td>
<td>320</td>
<td>1.04</td>
<td>0.5</td>
<td>53</td>
<td>9</td>
<td>65%</td>
<td>262</td>
<td>1.27</td>
<td>9</td>
<td>65%</td>
</tr>
<tr>
<td>FIR unrolled</td>
<td>67</td>
<td>131</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>9%</td>
<td>126</td>
<td>1.04</td>
<td>2</td>
<td>9%</td>
<td>126</td>
<td>1.04</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>94%</td>
<td>98</td>
<td>1.30</td>
<td>2</td>
<td>97%</td>
</tr>
<tr>
<td>FFT</td>
<td>10</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>12</td>
<td>2</td>
<td>60%</td>
<td>10</td>
<td>1.80</td>
<td>2</td>
<td>60%</td>
</tr>
</tbody>
</table>

Average: 50% 1.5 50% 1.7 83% 2 84% 2.3

\[ \text{CP max = 15000} \]
\[ \text{NN max = 5} \]
• Introduction
• Constraint programming
• Pattern generation
• Pattern selection and scheduling
• Instruction reordering
• Results
• Conclusion
Conclusion

- Constraints make it possible to explore solutions that are difficult to examine using specific algorithms.
- Constraints provide flexibility to the defining of the different conditions.
- (Sub-)graph isomorphism constraints offer an easy way of defining design problems.
- Experimental results are very encouraging.
Sélection automatique d'instructions et ordonnancement d'applications basés sur la programmation par contraintes

Merci!