Using Machine Learning to Focus Iterative Optimization


Institute of Computer Systems Architecture
Institute of Adaptive and Neural Computing

School of Informatics
University of Edinburgh
UK

March, 2006
Overview

- **Background + Motivation**
  - Embedded applications where performance is critical
  - Using predictive modelling to guide search/global optimisation

- **Models to focus search**
  - Examined two standard search algorithms: Random + GA
  - Propose two models: IID and Markov to focus search
  - Learning model using nearest neighbour classification

- **Evaluation**
  - An exhaustively enumerated small space $14^5$
  - A large test space $80^{20}$

- **Summary and Future work**
Compilers are unable to effectively exploit hardware resources
  - Fundamentally this is due to the complexity of the architecture

Static analysis based approaches try to model the space with simple models/heuristic on a piecemeal basis
  - Experiments show that the optimisation space is massively non-linear.
  - Furthermore architectures evolve faster compiler writers can react

Try a new approach iterative compilation: try different optimisations, run them - select the best.
UltraSparc for mm $N = 512$. 
Iterative compilation is now a well known technique:
  - Search the space using random, GA, hill climbing techniques
  - It gives good results but takes a long time - a barrier to use in general purpose setting

Basic idea is focus search on areas of space likely to be good

We determine these areas by learning form other programs.
  - So, if my program A is similar to previously searched program B, can I use knowledge of its space to focus my search?
Focus reduces search: Adpcm on TI C6713
How learning helps: TI C6713

The graph shows the percent of max improvement available versus evaluations for two different approaches: RANDOM and FOCUSED. The graph indicates that the FOCUSED approach reaches 86% of the max improvement available at around 10 evaluations, whereas the RANDOM approach reaches 38% at the same point. This suggests that focused learning can be more effective in a shorter number of evaluations compared to random learning.
Optimization Space $14^5$

- Embedded system application
  - UTDSP benchmarks: Compute intensive DSP
    - **AMD** Au1500 - gcc 3.2.1 -O3, **TI** C6713 v2.21-O3

- Exhaustively enumerated an interesting search space
  - 14 transformations selected.
  - All combinations of length 5 evaluated

- Allows comparison of techniques
  - How near the minima each technique approaches
  - Rate of improvement
  - Characterization of the space
Generating an Exhaustive Space

- Generate length 1 to 5 in order.
- If $ST = S$, record and prune subtree
Exhaustive enumeration: $14^5$

<table>
<thead>
<tr>
<th>Prog.</th>
<th>TI</th>
<th>AMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>fft</td>
<td>3.64%</td>
<td>3nm</td>
</tr>
<tr>
<td>fir</td>
<td>45.5%</td>
<td>4</td>
</tr>
<tr>
<td>iir</td>
<td>16.3%</td>
<td>3h</td>
</tr>
<tr>
<td>latnrm</td>
<td>0.34%</td>
<td>nsch</td>
</tr>
<tr>
<td>lmsfir</td>
<td>0.39%</td>
<td>1s</td>
</tr>
<tr>
<td>mult</td>
<td>0.00%</td>
<td>{}</td>
</tr>
<tr>
<td>adpcm</td>
<td>24.0%</td>
<td>1ish</td>
</tr>
<tr>
<td>compress</td>
<td>39.1%</td>
<td>4s</td>
</tr>
<tr>
<td>edge</td>
<td>5.06%</td>
<td>3</td>
</tr>
<tr>
<td>histogram</td>
<td>0.00%</td>
<td>{}</td>
</tr>
<tr>
<td>lpc</td>
<td>10.7%</td>
<td>sn2</td>
</tr>
<tr>
<td>spectral</td>
<td>7.46%</td>
<td>n4</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>12.7%</strong></td>
<td></td>
</tr>
</tbody>
</table>
How does blind search perform? TI
How does blind search perform? AMD

Percent of Max Improvement Available

Evaluations

RANDOM

GA
Two models to help focus search

We want models that summarise the space that
- Can be applied to similar programs to focus search
- Are cheap to learn and don’t overfit

Examine two basic models
- Identically independent distribution - very naive
  - Just note how often a transformation occurs in a good sequence
- Markov model - slightly smarter.
  - Considers limited interactions
Models: IID and Markov

- IID: Does not consider interactions

\[ P(s_1, s_2, \ldots, s_L) = \prod_{i=1}^{L} P(s_i). \]

- Markov: Considers previous transformation. Not location aware

\[ P(s_1, s_2, \ldots, s_L) = P(s_1) \prod_{i=2}^{L} P(s_i | s_{i-1}). \]
Models as oracles

- Want to check they are useful before trying to learn them
  - So we exhaustively enumerated space to learn each model
  - 14 transformations upto 5 in length - $14^5$
- IID has a 14 element vector.
  - One probability value per transformation
- Markov a $14 \times 14$ matrix.
  - For each transformation what is the probability of the next one.
- Used the model of the space to guide search of this space as a sanity check
  - Similar to hardware oracles
Oracle vs blind search

Average results on TI. Markov better than IID
Learning

- Used over 30 features to characterise programs
  - Then PCA to see which were relevant
  - Reduced to 5
- Used nearest neighbour as learning mechanism
- Evaluated mechanism on small and large $80^{20}$ space
  - Learnt on large space using 1000 training examples
- Compared against random over first 50 evaluations.
Performance on small space

 TI: Markov is best
## Performance on large space

<table>
<thead>
<tr>
<th></th>
<th>2 Evaluations</th>
<th>5 Evaluations</th>
<th>50 Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R</td>
<td>M</td>
<td>I</td>
</tr>
<tr>
<td>TI</td>
<td>1.10</td>
<td>1.25</td>
<td>1.26</td>
</tr>
<tr>
<td>AMD</td>
<td>1.08</td>
<td>1.24</td>
<td>1.27</td>
</tr>
</tbody>
</table>

- Significant improvement in first 2 evaluations (1.26, 1.27)
  - R = Random, M = Markov, I = IID
- Focus gives an order of magnitude improvement.
  - Greater performance after 5 evaluations vs 50 of random
- IID outperforms Markov on large space
  - Learning an 80 element vector vs $80 \times 80$ matrix with 1000 samples
Summary and Future Work

- Learning models to focus search works
  - More sophisticated models need more training data
- For continuous optimisation switch models as certain point
- Can be used with other work to reduce cost of each evaluation.
  - Automatically choose the space for self-tuning
- Ultimate goal is to use ML to make iterative compilation as cheap as profile-directed schemes
- Best single transformation across $14^5$
  - himc3 on AMD - speedup 1.11
  - No single best on TI
- Effective space: measure of pruning
  - Varies: 0.85% on histogram, 15.4%
- PCA: 5 vectors of 26 weights. Account for over 95% of variance
  - No clear feature dominating.
  - Loop structure important
### Nearest Neighbour

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Nearest Neighbor</th>
</tr>
</thead>
<tbody>
<tr>
<td>fft</td>
<td>lpc</td>
</tr>
<tr>
<td>fir</td>
<td>compress</td>
</tr>
<tr>
<td>iir</td>
<td>fir</td>
</tr>
<tr>
<td>latnrm</td>
<td>iir</td>
</tr>
<tr>
<td>lmsfir</td>
<td>iir</td>
</tr>
<tr>
<td>mult</td>
<td>compress</td>
</tr>
<tr>
<td>adpcm</td>
<td>fir</td>
</tr>
<tr>
<td>compress</td>
<td>fir</td>
</tr>
<tr>
<td>edge</td>
<td>iir</td>
</tr>
<tr>
<td>histogram</td>
<td>fir</td>
</tr>
<tr>
<td>lpc</td>
<td>spectral</td>
</tr>
<tr>
<td>spectral</td>
<td>lpc</td>
</tr>
</tbody>
</table>