Automatic Selection of Machine Learning Models for WCET-aware Compiler Heuristic Generation

Paul Lokuciejewski | Marco Stolpe | Katharina Morik | Peter Marwedel

TU Dortmund University
Department of Computer Science 12, 844221 Dortmund
Germany
Outline

- Introduction
  - Motivation for WCET-aware Compiler Optimizations
  - Compilers and Machine Learning
- Current Workflow for MLB Heuristic Generation
- Automatic Model Selection
  - Learning Algorithms
  - Performance Evaluation
  - Parameter Optimization
- Case Study: Loop Invariant Code Motion
- Experimental Environment & Results
- Conclusions & Future Work
Motivation

- Embedded Systems used as Real-Time Systems
- Worst-case execution time (WCET) is a key parameter
  - Crucial for safety-critical systems
  - Required for task scheduling
  - Helps to design hardware platforms
- Estimated by static timing analyzers

Meeting Real Time Constraints

 😞 Trial-and-error approaches
 😊 Automatic WCET reduction by compilers
  - Integration of timing analyzer into compiler framework
Compiler Developers’ Struggle

- Development of compiler heuristics tedious
  - Requires expertise and extensive trial-and-error tuning
  - Complicated by advent of complex architectures
    - Abstract models don’t exploit target features
  - Optimizations performed in a sequence
    - Interference with possible conflicts

- Which choices do we have?
  - Use conservative assumptions missing optimization potential
  - Tune heuristic for a fixed optimization sequence
Machine Learning Based Compiler Heuristics

- Finding relevant information in high dimensional space
- Help to understand and control complex systems
- ML approaches allow automatic heuristic generation

<static features> \rightarrow \text{heuristic parameters}

Benefits

- Learning results enhance flexibility of compiler
  - Automatic \textit{re-learning} for new target / optimization sequence
- Reduce effort for compiler development
Outline

- Introduction
  - Motivation for WCET-aware Compiler Optimizations
  - Compilers and Machine Learning

- Current Workflow for MLB Heuristic Generation

- Automatic Model Selection
  - Learning Algorithms
  - Performance Evaluation
  - Parameter Optimization

- Case Study: Loop Invariant Code Motion

- Experimental Environment & Results

- Conclusions & Future Work
Current Workflow for Heuristic Generation

Representation of Program in Compiler

Feature Extraction / Label Determination

Machine Learning Algorithm Selection

- Decision Trees
- Naive Bayes
- kNN
- SVM
- Random Forests

Learner: Par1 = c1
Par2 = c2
... ParN = cN

Supervised Learner Model Induction

Learner → Prediction Model / Heuristic
Problem Specification: Model Selection

- Goal: Find induced model with best performance
- But
  - Complex structure of learning algorithms
  - Non-trivial impact on prediction => performance
- Prediction of performance of induced models infeasible
- Evaluate generated heuristics using cross-validation

马桶 Current approach: Trial-and-error
  - Time-consuming, error-prone & benefits of further tuning?
  - Few combinations of learner/parameters tested

Effort shifted from manual tuning to model selection
Outline

- Introduction
  - Motivation for WCET-aware Compiler Optimizations
  - Compilers and Machine Learning
- Current Workflow for MLB Heuristic Generation
- Automatic Model Selection
  - Learning Algorithms
  - Performance Evaluation
  - Parameter Optimization
- Case Study: Loop Invariant Code Motion
- Experimental Environment & Results
- Conclusions & Future Work
Automatic Model Selection

Systematic evaluation of induced models by different learners and parameter settings
Learning Algorithms

- **Decision Trees**: Split training set into sub-trees
  - Splitting criterion, depth of trees, min. # examples in leaf ...
- **Random Forests**: Sets of decision trees + majority vote
  - # of trees, # randomly chosen features for node split ...
- **Linear SVM**: Find hyperplane to separate examples
  - Soft margin for misclassification
- **SVM with RBF kernel**: separation based on Gaussian dist.
  - Soft margin for misclassification, Gaussian’s width
- **k-Nearest Neighbor**: based on nearest neighbors classes
  - Number of considered neighbors
- **Naïve Bayes**: probabil. classifier based on Bayes’ theorem
Performance Evaluation (1)

- Standard performance measurement: **Accuracy**
  - Comparison of predicted and real class using $N$-fold cross validation

- Goal for embedded RT systems: **WCET minimization**

Accuracy the right measure?
Performance Evaluation (2)

Example

- 4 Optimization Decisions:

<table>
<thead>
<tr>
<th></th>
<th>YES</th>
<th>NO</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-10</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>-10</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>-20</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>50</td>
<td>0</td>
</tr>
</tbody>
</table>

Scenario 1:
- A, B, C correct
  - -10
  - -10
  - -20
  - 50 => 10
  - WCET increase
  - Accuracy: 75%

Scenario 2:
- C, D correct
  - 0
  - 0
  - -20
  - 0 => -20
  - WCET decrease
  - Accuracy: 50%

- Accuracy not suitable performance measure
**Performance Evaluation (3)**

- Benchmark-wise cross-validation based on WCET

```plaintext
for all algorithm alg
    for all parameter settings set
        performance = 0
        for all benchmarks b in training set E {
            generateHeuristic_{MLB}(alg, s, E \ b)
            WCET_{MLB} = computeWCET_{MLB}(b)
            performance += (WCET_{MLB} / WCET_{ref})
        }

    evaluation[alg][s] = performance / |E|
```
Parameter Optimization

- Exhaustive search over all combinations of user-defined learner parameters not feasible

Our solution: Evolutionary search

- Genetic algorithm
- Represent each parameter combination as individual
- Performance (fitness) calculation based on WCET
- Reproduction via one-point crossover & mutation
Workflow of Parameter Optimization

Evolutionary Parameter Optimization

Fitness Evaluation of Each Individual

Cross Validation

ML Model

Performance Computation

Best Parameter Combination

Average Performance (Parameter Quality)
Outline

- Introduction
  - Motivation for WCET-aware Compiler Optimizations
  - Compilers and Machine Learning
- Current Workflow for MLB Heuristic Generation
- Automatic Model Selection
  - Learning Algorithms
  - Performance Evaluation
  - Parameter Optimization
- Case Study: Loop Invariant Code Motion
- Experimental Environment & Results
- Conclusions & Future Work
Loop Invariant Code Motion (LICM)

- Well-known ACET optimization
- Moves *loop invariant* computations outside the loop
- Can be applied at source code or assembly level

Positive Effects

- Reduced execution frequencies of shifted invariants
- Positive effects on instruction cache
- Shortens variable live ranges → decreased reg. pressure

Negative Effects

- Moved computations increase register pressure
- Lengthen other CFG paths (above zero-trip test, …)
Heuristics for LICM

- Heuristics for LICM not trivial
  - register pressure dilemma …
- No heuristics in compiler literature
- LICM performed whenever possible

Automatic Generation of LICM Heuristics using Supervised Machine Learning
Outline

- Introduction
  - Motivation for WCET-aware Compiler Optimizations
  - Compilers and Machine Learning
- Current Workflow for MLB Heuristic Generation
- Automatic Model Selection
  - Learning Algorithms
  - Performance Evaluation
  - Parameter Optimization
- Case Study: Loop Invariant Code Motion
- Experimental Environment & Results
- Conclusions & Future Work
Compiler Framework

- **WCC** – WCET-aware C compiler for Infineon TC1796

[Diagram showing the Compiler Framework with WCC highlighted.]

[*http://ls12-www.cs.tu-dortmund.de/research/activities/wcc/]
Experimental Setup

- 39 Benchmarks from DSPstone, MediaBench, UTDSP ...
- 3491 examples (LICM instructions)
- 73 static features to characterize loop-invariant instruction or its environment
  - Structural features, LTA-related, loop-related, reg. pressure, ...
  - Large number of features to support generality of framework

Construction of Training Set

- Feature Extraction: for each instruction before LICM
- Label: Measure WCET before and after LICM (YES|NO)
- Training phase took 50 hours on 2.13 GHz 4-Core
**Evolutionary Parameter Optimization**

- Parameter optimization performed for all 6 learners
  - Finds MLB heuristic with highest reduction of the WCET

- Performance evaluation (fitness) based on benchmark-wise cross validation

- Highly optimized code (O3) w/o LICM used as reference

- Optimization time between 5h - 37h on 2.13 GHz 4-Core

**Final State**

- Best model integrated into compiler framework
- Before LICM, prediction by ML tool
# Results – Parameter Optimization (1)

## Decision Trees

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>max. depth</td>
<td>[1;20]</td>
<td>16</td>
</tr>
<tr>
<td>min. split size</td>
<td>[4;100]</td>
<td>19</td>
</tr>
<tr>
<td>min. leaf size</td>
<td>[2;100]</td>
<td>31</td>
</tr>
<tr>
<td>min. gain</td>
<td>[0;0.03]</td>
<td>0.014</td>
</tr>
<tr>
<td>prepr. altern.</td>
<td>[3;10]</td>
<td>4</td>
</tr>
<tr>
<td>confidence</td>
<td>[0.1;0.5]</td>
<td>0.476</td>
</tr>
</tbody>
</table>

## Random Forests

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>no of trees</td>
<td>[1;100]</td>
<td>7</td>
</tr>
<tr>
<td>features</td>
<td>[1;73]</td>
<td>39</td>
</tr>
</tbody>
</table>

## SVM with RBF kernel

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>[0;10,000]</td>
<td>2405.15</td>
</tr>
<tr>
<td>γ</td>
<td>[0.74]</td>
<td>30.08</td>
</tr>
</tbody>
</table>

## Linear SVM

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>[0;10,000]</td>
<td>616.11</td>
</tr>
</tbody>
</table>

## kNN

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Best</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>[3;100]</td>
<td>11</td>
</tr>
</tbody>
</table>

## Naive Bayes

*no configurable parameters*

- #parameter combinations too large for exhaustive search
Results – Parameter Optimization (2)

- WCET reduction varies between 1.76% and 4.64%
- Compared to standard ACET LICM achieving a WCET reduction of 0.56%, significant improvement of 8.3x
- Parameters have strong impact on learner performance
- No correlation between WCET reduction and accuracy

<table>
<thead>
<tr>
<th>Learner</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision Tree</td>
<td>96.17%</td>
<td>99.78%</td>
<td>97.42%</td>
<td>63.16%</td>
</tr>
<tr>
<td>Random Forests</td>
<td>96.60%</td>
<td>98.96%</td>
<td>97.69%</td>
<td>60.43%</td>
</tr>
<tr>
<td>Linear SVM</td>
<td>98.24%</td>
<td>98.62%</td>
<td>98.34%</td>
<td>53.50%</td>
</tr>
<tr>
<td>SVM with RBF kernel</td>
<td>95.36%</td>
<td>98.80%</td>
<td>97.12%</td>
<td>57.78%</td>
</tr>
<tr>
<td>kNN</td>
<td>97.32%</td>
<td>98.94%</td>
<td>97.98%</td>
<td>67.48%</td>
</tr>
<tr>
<td>Naive Bayes</td>
<td>98.17%</td>
<td>98.17%</td>
<td>98.17%</td>
<td>54.31%</td>
</tr>
</tbody>
</table>
Progress of Evolutionary Parameter Optimization

- Convergence plot for best learner (SVM with RBF kernel)
Results – Relative WCET for Best Model

- 100% corresponds to –O3 without LICM
- Average: Standard 0.56% ; WCET-driven LICM 4.64%
Outline

- Introduction
  - Motivation for WCET-aware Compiler Optimizations
  - Compilers and Machine Learning
- Traditional Function Inlining
- WCET-driven Function Inlining
  - Supervised Learning (Random Forests)
  - Feature / Label Extraction
  - Heuristic Generation
- Experimental Environment & Results
- Conclusions & Future Work
Conclusions & Future Work

- Central issue in MLB heuristic generation: Model Selection
- Choice of learner and its parameters has strong impact on performance of induced model (heuristic)
- Exhaustive search not feasible => Evolutionary search
- Case study on WCET-driven LICM
- Novel optimization outperforms std. LICM by 8.3x

Future Work

- Evaluation of further learning algorithms & optimizations
- Feature selection
Thank you for your attention.