Active Learning Accelerated Automatic Heuristic Construction for Parallel Program Mapping

William F. Ogilvie
University of Edinburgh
s0198982@inf.ed.ac.uk

Pavlos Petoumenos
University of Edinburgh
ppetoume@inf.ed.ac.uk

Zheng Wang
Lancaster University
z.wang@lancaster.ac.uk

Hugh Leather
University of Edinburgh
hleather@inf.ed.ac.uk

ABSTRACT
Building effective optimization heuristics is a challenging task which often takes developers several months if not years to complete. Predictive modelling has recently emerged as a promising solution, automatically constructing heuristics from training data, however, obtaining this data can take months per platform. This is becoming an ever more critical problem as the pace of change in architecture increases. Indeed, if no solution is found we shall be left with out of date heuristics which cannot extract the best performance from modern machines.

In this work, we present a low-cost predictive modelling approach for automatic heuristic construction which significantly reduces this training overhead. Typically in supervised learning the training instances are randomly selected to evaluate regardless of how much useful information they carry, but this wastes effort on parts of the space that contribute little to the quality of the produced heuristic. Our approach, on the other hand, uses active learning to select and only focus on the most useful training examples and thus reduces the training overhead.

We demonstrate this technique by automatically creating a model to determine on which device to execute four parallel programs at differing problem dimensions for a representative CPU-GPU based system. Our methodology is remarkably simple and yet effective, making it a strong candidate for wide adoption. At high levels of classification accuracy the average learning speed-up is 3x, as compared to the state-of-the-art.

Categories and Subject Descriptors
D.3.4 [Programming Languages]: Processors—compilers, optimization

Keywords
Active Learning; Machine Learning; Compilers

1. INTRODUCTION
Creating analytical models on which optimization heuristics can be based has become harder as processor complexity has increased. Compiler developers often have to spend months if not years to get a model perfected for a single target architecture, and since modern compilers often support a wide range of disparate platforms most have been found to be out of date.

Machine Learning based predictive modelling has rapidly emerged as a viable means of automating heuristic construction\cite{1, 3}; by running example programs (optimized in different ways) and observing how the variations affect program run-time a machine learning tool can predict good settings with which to compile new, as yet unseen, programs. This new research area is promising, having the potential to fundamentally change the way compiler heuristics are designed, but suffers from a number of issues. One major concern is the cost of collecting training examples. While machine learning allows us to automatically construct heuristics with little human involvement, the cost of generating training examples (that allow a learning algorithm to accumulate knowledge) is often very expensive.

In this work we present a novel, low-cost predictive modelling approach that can significantly reduce the overhead of collecting training examples for parallel program mapping without sacrificing prediction accuracy. The usual procedure in supervised learning is to passively collect training examples at random, regardless of how useful they might be to the learner. We propose using active learning instead, which is a method by which the learning algorithm itself is able to iteratively choose the training instances it believes carry the greatest information based upon whatever knowledge it has already accumulated. Specifically, we use active learning to automatically construct a heuristic to determine which processor will give the better performance on a CPU-GPU based heterogeneous system at differing problem sizes for a given program. We evaluate our system by comparing it with the typical random sampling methodology, used in the bulk of prior work. The experimental results show that our technique accelerates training by a factor of 3x on average.

2. OUR APPROACH
We use a heterogeneous Query-by-Committee (Qbc) implementation of active learning, which is so-named because it requires that a number of different models be generated us-
Table 1: The sizes of the input spaces for each benchmark. Dim indicates the number of dimensions – each dimension is then treated in the same way for our case study. Min gives the minimum value of each dimension. Max gives the maximum value of each dimension. Step gives the step value on each dimension. Size is the total number of points in the input space. Cand is the number of points in the candidate set for each benchmark.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Dim</th>
<th>Min</th>
<th>Max</th>
<th>Step</th>
<th>Size</th>
<th>Cand</th>
</tr>
</thead>
<tbody>
<tr>
<td>HotSpot</td>
<td>2</td>
<td>1</td>
<td>128</td>
<td>1</td>
<td>10,000</td>
<td>10,000</td>
</tr>
<tr>
<td>MatMul</td>
<td>3</td>
<td>1</td>
<td>256</td>
<td>1</td>
<td>1.62*10^4</td>
<td>10,000</td>
</tr>
<tr>
<td>Pathfinder</td>
<td>2</td>
<td>2</td>
<td>1024</td>
<td>1</td>
<td>1.02*10^5</td>
<td>10,000</td>
</tr>
<tr>
<td>SRAD</td>
<td>2</td>
<td>128</td>
<td>1024</td>
<td>16</td>
<td>3.136</td>
<td>2,636</td>
</tr>
</tbody>
</table>

Figure 1: On average our approach requires 3x fewer training examples to create a high quality heuristic than a random sampling technique. These speed-up values are defined as the number of training instances required for the accuracy of a heuristic to plateau using the random technique divided by that for our Qbc implementation.

**Termination Criterion.**

In all cases, the active learning iterations were halted at 200 steps. This value was selected because for all benchmarks the learning improvement had plateaued by that time.

**Run-time Measurement and Device Comparison.**

To determine if a benchmark is better suited to the CPU or GPU for a given input it was run on each processor at least 10 times, and at most 200 times. We employed a number of statistical techniques to ensure our data was as accurate as possible, including Welch’s t-test, equivalence testing, and interquartile-range outlier removal.

**Testing.**

For testing, a set of 500 inputs were excluded from any training sets. Both our active and passive learning experiments were run 10 times and the arithmetic mean of the actual values are based on the number of training examples required to produce a heuristic which is at least 90% accurate. Overall, we are able to accelerate training by 3x on average. In real terms, this translates to a saving of weeks of compute time.

4. RESULTS

Figure 1 shows the average learning speed-up of our approach over the classical random sampling technique. The actual values are based on the number of training examples required to produce a heuristic which is at least 90% accurate. Overall, we are able to accelerate training by 3x on average. In real terms, this translates to a saving of weeks of compute time.

5. REFERENCES

