Coursework Part 2

Michael O’Boyle

February, 2013
Second Course Work

- Based on your first course work.

- Only for MSc: 12.5% of course mark.

- Due Thursday March 21st week 10 4pm!

- Penalties for late submission.

- Plagiarism software used. Do your own work!
The Goal of the Project

• Exploit the data collected in the first course work to build an improved optimisation strategy.

• Build a model based on the data from eight benchmark programs (training set) to improve performance of the remaining three (validation set).

• Based on “Compiler Optimization-Space Exploration” paper by Triantafyllis et al. (CGO 2003)

• Write a report about your methodology and your findings.
What you should have done by now

• Collected performance results from all benchmarks on a number of different optimisation configurations.

• In other words you have:
  a) a set of benchmark programs $P$
  b) a set of configurations $C$
  c) performance results $perf(p_i, c_i)$ for each program $p_i \in P$ and configuration $c_i \in C$

• If you don’t have the performance data for each program and each configuration, you maybe have to run some more experiments
Basic Idea

- Find configurations that give good performance across all programs.

- Group programs according to their performance on these configurations.

- Gradually find more specialized configurations by only considering subsets of programs.

- Idea: Pruning the search space by only considering optimisations that worked well on “similar” programs.

- Similar to product recommendation (e.g. at Amazon.co.uk): Programs that perform well with configuration 1 also perform well with configuration 2.
Optimizing a New Program

To quickly find a good configuration for a new program:

- Start at the root node and compare the performance of the program with the two configurations found in its child nodes.
- Move to the node with the configuration that gives a better speedup.
- Repeat these steps until you’ve reached a leaf node.
- Pick the configuration on the path from the root to the leaf node that gave the best performance.
Instructions

• Randomly choose 8 of the 11 benchmarks as a training set.

• Construct a configuration tree based on the performance results of these benchmarks on different configurations.

• Use the remaining 3 programs as a validation set, i.e. find optimizations for these benchmarks using the configuration tree.

• If you don’t have the performance values for each program on the configurations used, you may have to run some more experiments.

• Compare your results and the number of evaluations needed to your results from the first course work.
Report and Results

• Maximum 5 pages + 2 pages for results

• Explain what you have done.

• Report your results and compare them to optimisation levels of GCC (-00, -01, -02, -03)

• Also compare them to your results from the first course work.

• Compare the number of evaluations you needed in both course works.
Report Structure

• Abstract (Summary of paper) and Introduction

• Methodology. CGO algorithm

• Results
  – Baseline -00, -01, -02, -03 for each program in validation set.
  – Best found flags and exec times for validation set with both approaches.
  – Number of evaluations needed to arrive at result in both approaches.

• Analysis and Discussion of Results. Followed by conclusion.

• Submit to ITO written report by 4pm Thursday 21st March.
Final Remarks

- For further questions
  - Contact TA: Chris Margiolas s1152011@sms.ed.ac.uk

- Start early!!
  You may have to run more experiments!

- Deadline: 21/03/2013 4pm