Phase Order Search with Feature Selection
João Martins & David Henriques

Group Info

João Martins - jmartins@cs.cmu.edu
David Henriques - dhenriqu@cs.cmu.edu

Project Web Page

`http://www.cs.cmu.edu/~jmartins/15745/`

Project Description

Running a compiler optimisation pass usually leads to more code being optimizable in subsequent passes. For example, constant propagation (CP) frequently makes several variable assignments become dead code. If we run CP after running dead code elimination (DCE), we either have to run DCE again (increasing compilation time) or leave useless code in the binary (increasing runtime). This makes choosing a suitable ordering for running passes an interesting and relevant problem, called *Phase Order Search*.

We propose to implement an analysis pass that gathers statistics (features) about code. Using this pass on the SPEC 2006 benchmarks, we will then train a classifier to return a suitable solution for the phase order search problem. Finally, we will evaluate our classifier.

In this work, we will focus especially on the following points.

- An exhaustive but naïve approach would use each ordering of passes as a class. Unfortunately, the number of possible orderings is factorial. To adequately cover all these classes, we would need a very large training dataset, i.e. a number of benchmarks we do not have access to. We propose to identify manageable subsets of all orderings to use as classes. We will extract these subsets from the training data set using heuristics based on partial orderings, non-exhaustive pairwise comparisons, etc.

- Many code statistics appear to be meaningful for classification. However, it is very common for some of them to be almost irrelevant in practice. In fact, these extraneous features introduce noise in the classification process and make it harder to produce useful results. In the machine learning community there is a set of techniques, collectively called *feature selection*, that focuses on identifying and extracting only the most important and relevant features. We propose to compare some of these techniques, such as principal component analysis or ridge and lasso regression, and how they interact with the selected classes.

Since we are putting a strong focus on selecting relevant features, for this project we will try to use classification algorithms that expose the selected features. This should give us some useful insights in our tasks.

For training and validation, we plan on using the standard SPEC 2006 benchmarks.

Our goals are as follows:

75% Our reduced objectives are to find a reasonable subset of possible orderings and learn a classification function using an appropriate feature selection method. Perform a comparison with unoptimised code, LLVM optimised code, and random-ordering optimised code.

100% In addition, we propose to study and compare several methods for limiting the phase-order space.

125% Time permitting, we will implement and evaluate different feature selection methods, and extend the training set to more benchmarks.

Logistics

Plan of attack and schedule

Week	David	João
1	Literature review	SPEC 2006 setup
2	Find appropriate set of orderings, learning technique	
3	Find methods for 100% goal	Develop basic framework
4	Implement methods for 100% goal	75% goal done
5	Tests for 100% goal	
6	Finalise and writeup	

The critical path will be setting up the benchmarks, implementing our method and running a robust and informative set of tests.

Milestone

By the milestone, we would like to have met our 75% goal in order to validate the approach before extending it.

Literature Search

[1, 2] address the problem of finding an optimal ordering of optimisation passes, but do not discuss learning techniques. [3] gives an overview of feature selection techniques in general settings. [4] tackles the long runtimes of iterative compilation by using limited heuristics and static performance impact analysis.

Resources Needed

For this project, we will need access to a language that supports ML libraries (MATLAB, Java, etc) and access to the SPEC benchmarks.

Getting Started

So far, we have identified many of the options for the choices we will have to make, such as using PCA or lasso regression for feature selection, or pairwise comparisons between passes and partial orderings for selecting subsets of orderings, etc. We also have several papers from which to draw further ideas. With these options on the table, we are ready to start thinking about which of them to implement.

References

[1] Agakov, F. and Bonilla, E. and Cavazos, J. and Franke, B. and Fursin, G. and O'Boyle, M. F. P. and Thomson, J. and Toussaint, M. and Williams, C. K. I. Using Machine Learning to Focus Iterative Optimization. In *Proceedings of the International Symposium on Code Generation and Optimization (CGO, pages 295-305)*, 2006.

[2] Prasad A. Kulkarni and David B. Whalley and Gary S. Tyson. Evaluating heuristic optimization phase order search algorithms. In *Proceedings of the International Symposium on Code Generation and Optimization (CGO, pages 157-169)*, 2007.

[3] Andrew Y. Ng. On Feature selection: Learning with Exponentially many Irrelevant Features as Training Examples. In *Proceedings of the Fifteenth International Conference on Machine Learning (pages 402-414)*. 1998.

[4] Triantafyllis, Spyridon and Vachharajani, Manish and Vachharajani, Neil and August, David I. Compiler optimization-space exploration. In *Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization (CGO, pages 204-215)*. 2005.