Compiler Optimizations using Deep Reinforcement Learning

Thomas Howard III
University of Rhode Island

March 26, 2018

Abstract

Treating compiler optimizations as a contextual-bandit problem will allow a deep reinforcement model to successfully map features to compiler flags, surpassing human-level ability with a *tabula rasa* approach.

1 Introduction

When compiling high-level code for release, most developers want the fastest runtime, most power efficient operation, and/or smallest memory footprint; yet, few actually know how to get this top-level performance out of modern compilers. In fact, most compilers support hundreds of possible optimization flags that can be rearranged, repeated, left out completely, or any combination of the former. It is because of this extreme complexity that no perfect compiler exists. Current solutions to this problem use various statistical and machine-learning-based methods to try and derive the best possible options, yet fall short of truly optimal performance, or lack the ability to generalize across multiple architectures and benchmarks.

Trying to eke out performance gains for production releases may seem trivial to some, however, when one considers the impact this may have on embedded devices, it is impossible to consider the impact as anything less than massive. Source code for embedded devices, such as cars, power tools, digital watches, pace makers, etc., is normally compiled a single time, before being deployed to millions of devices. Even with conservative estimates of 40 embedded devices per household, better preforming compilers would result in billions of devices performing faster, saving energy, and requiring less memory.

To truly master the compiler optimization space, one needs to consider learning *tabula rasa*, since no ground truth actually exists about the search space. At it's heart, this problem is comparable to the work Deep Mind has done with the game of Go, which similarly exhibits a cost-prohibitively-large search-space. Unlike Go, compiler optimization emits no change in state, and must therefore be treated as an extremely large contextual-bandit problem, where the programs are the bandits, and the arms represent potential optimization sequences. The agent then interacts with these bandits in order to learn the relations between program features and compiler optimizations, using deep learning and hind-sight experience replay to build underlying relations directly from the full feature set.

This paper explores how deep reinforcement learning copes with such a large search-space in a contextual-bandit problem by first implementing it on a smaller scale. Starting with just seven possible flags, arranged in sub-sets of various lengths, without allowing reordering or repetition. The goal of this work is to produce a model capable of examining the context of a program, and producing the optimal flags out of the provided search-space. For this work, the only metric being optimized is the runtime of a given program, in future works both larger search-spaces, and multi-goal approaches should be explored.

2 Conclusions

There is no longer LaTeX example which was written by [Ashouri].

References

[Ashouri] *COBAYN: Compiler autotuning with BAYsian Network*