Optimizing Intopt Report

Marcus Begic
Fredrik Horn Dannert

December 2021

1 Introduction

This report will look into how we can optimize the ILP given in laboration 4 by analyzing the program using various tools and strategies. We can check how our program performs by running it against the test suite found on Forsete and note how our score changes. What we have noted in previous labs, using given tools, is the Pivot function is taking the most time out of all operations in the program. This is where most of the time will be spent optimizing.

2 Optimizations

2.1 Vectorizing

Vectorization allows SIMD instructions to be used to effectively introduce parallelism into the code, which will substantially speed up certain iterative calculations. GCC optimization levels O3 and above automatically enable this optimization. What we have to ensure, in the code, is that the compiler can detect patterns in loops that would allow for vectorization.

2.2 Branchless Pivot

Since we don't want to reschedule instructions when bumping into an incorrectly predicted branch we can simply remove them by rewriting the loops to not contain any if statements. By removing all branches in the pivot function we were able to improve the score by a factor of 1.8 on Forsete.

2.3 Floating point division

Studying the execution of intopt with sollpv we find that one of the mostly costly instructions is floating point division. This instruction also cannot be pipelined and instead has to be executed one after another. It's particularly expensive when we have division in loops just because it's a repeated operation.

So we solved this in pivot by calculating the quotient outside of for-loops and using multiplication instead.

Although one has to be careful when doing this, as one can lose accuracy unknowingly, however, this seems to have no unwanted side effect with the needed accuracy for this assignment.