Tuning the Og pipeline using automated testing

Stephen Tozer

What is Og?

“-Og should be the optimization level of choice for the standard edit-compile-debug cycle, offering a reasonable level of optimization while maintaining fast compilation and a good debugging experience.”

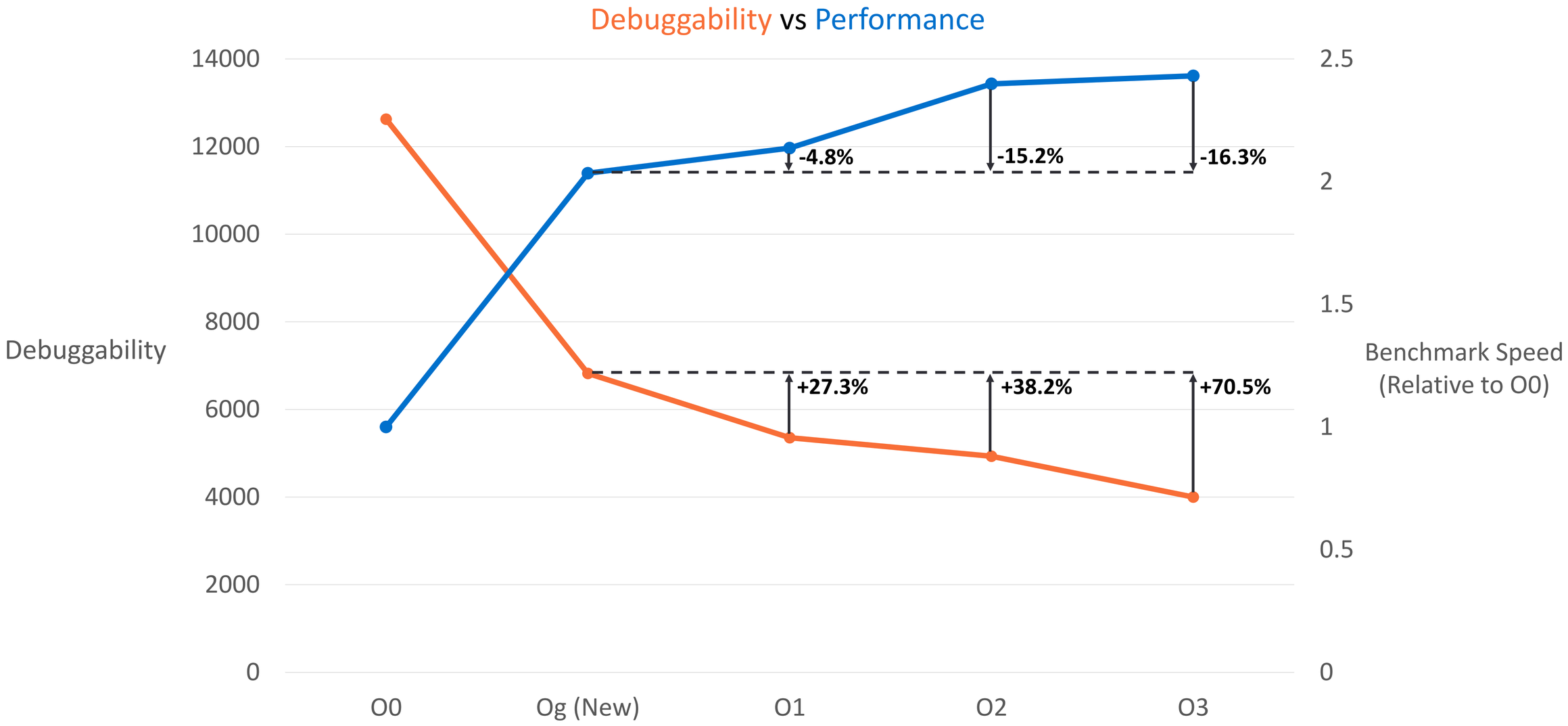
- GCC

“Like -O1. In future versions, this option might disable different optimizations in order to improve debuggability.”

- Clang

The Debuggability Metric

- A Dexter script is written declaring a set of variables and their expected values over a range of source locations.
- Dexter debugs the compiled test program and records the actual values for each variable.
- The debuggability score is then calculated as the sum, for each variable, of the number of source locations where that variable's actual value was equal to its expected value.
- Performance was recorded separately using existing benchmarks.
- We selected a set of potential Og pipelines, recorded the debuggability and performance for each, and selected the strongest candidate.
- All results obtained using a fork of clang version `5cf549e6` (post-LLVM16).


Debuggability vs Performance
Debuggability
Benchmark Speed
(Relative to O0)
14000
12000
10000
8000
6000
4000
2000
0
2.5
2
1.5
1
0.5
0
-4.8%
-15.2%
-16.3%
+27.3%
+38.2%
+70.5%
O0
Og (New)
O1
O2
O3

Optimizations at O1

IPSCCP
CalledValuePropagation
GlobalOpt
InstCombine
SimplifyCFG
Inliner
SROA
EarlyCSE
SimplifyCFG
InstCombine
LibCallsShrinkWrap
SimplifyCFG
Reassociate
LoopSimplify
LCSSA
SimplifyCFG
InstCombine

LoopSimplify
LCSSA
SROA
MemCpyOpt
SCCP
BDCE
InstCombine
ADCE
SimplifyCFG
InstCombine
DeadArgumentElimination
GlobalOpt
GlobalDCE
EliminateAvailableExternally
Float2Int
LowerConstantIntrinsics

LoopSimplify
LCSSA
LoopDistribute
InjectTLIMappings
LoopVectorize
LoopLoadElimination
InstCombine
SimplifyCFG
VectorCombine
InstCombine
LoopUnroll
SROA
InstCombine
LoopSimplify
LCSSA
AlignmentFromAssumptions
LoopSink
InstSimplify
DivRemPairs
SimplifyCFG
GlobalDCE
ConstantMerge

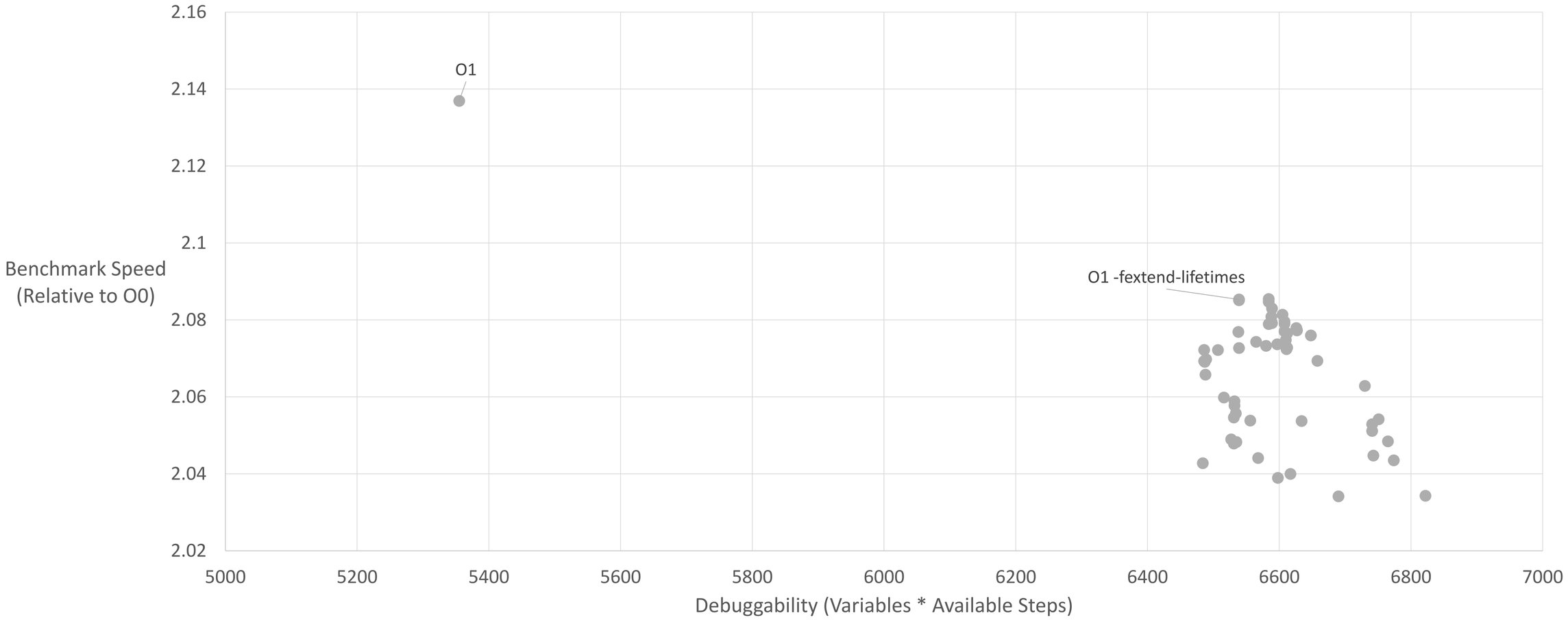
Optimizations at O1: Coarsely Grouped

IPSCCP CalledValuePropagation GlobalOpt InstCombine SimplifyCFG	Initial Simplification
Inliner	Inlining
SROA EarlyCSE SimplifyCFG	Post-Inline Restructuring
InstCombine LibCallsShrinkWrap SimplifyCFG Reassociate	Post-Inline Instruction Simplification
LoopSimplify LCSSA SimplifyCFG InstCombine	Basic Loop Simplification

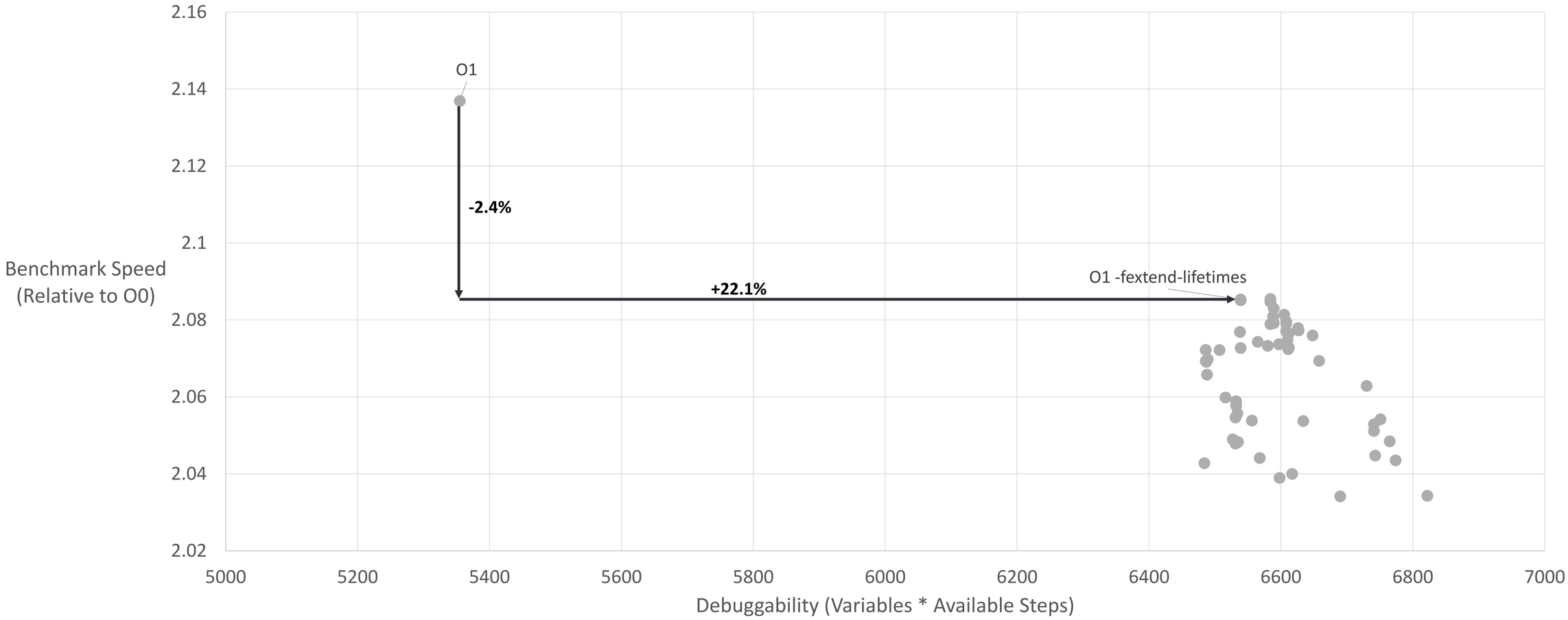
LoopSimplify LCSSA SROA MemCpyOpt SCCP BDCE InstCombine ADCE SimplifyCFG InstCombine	Loop Simplify, DCE, & Value Propagation
DeadArgumentElimination GlobalOpt GlobalDCE EliminateAvailableExternally	Global Optimizations
Float2Int LowerConstantIntrinsics	Instruction-Level Optimizations

LoopSimplify LCSSA LoopDistribute InjectTLIMappings LoopVectorize LoopLoadElimination InstCombine SimplifyCFG VectorCombine	Loop Vectorizing
InstCombine LoopUnroll SROA InstCombine LoopSimplify LCSSA AlignmentFromAssumptions LoopSink	Loop Reshaping
InstSimplify DivRemPairs	Instruction-Level Optimizations
SimplifyCFG GlobalDCE ConstantMerge	Global Optimizations

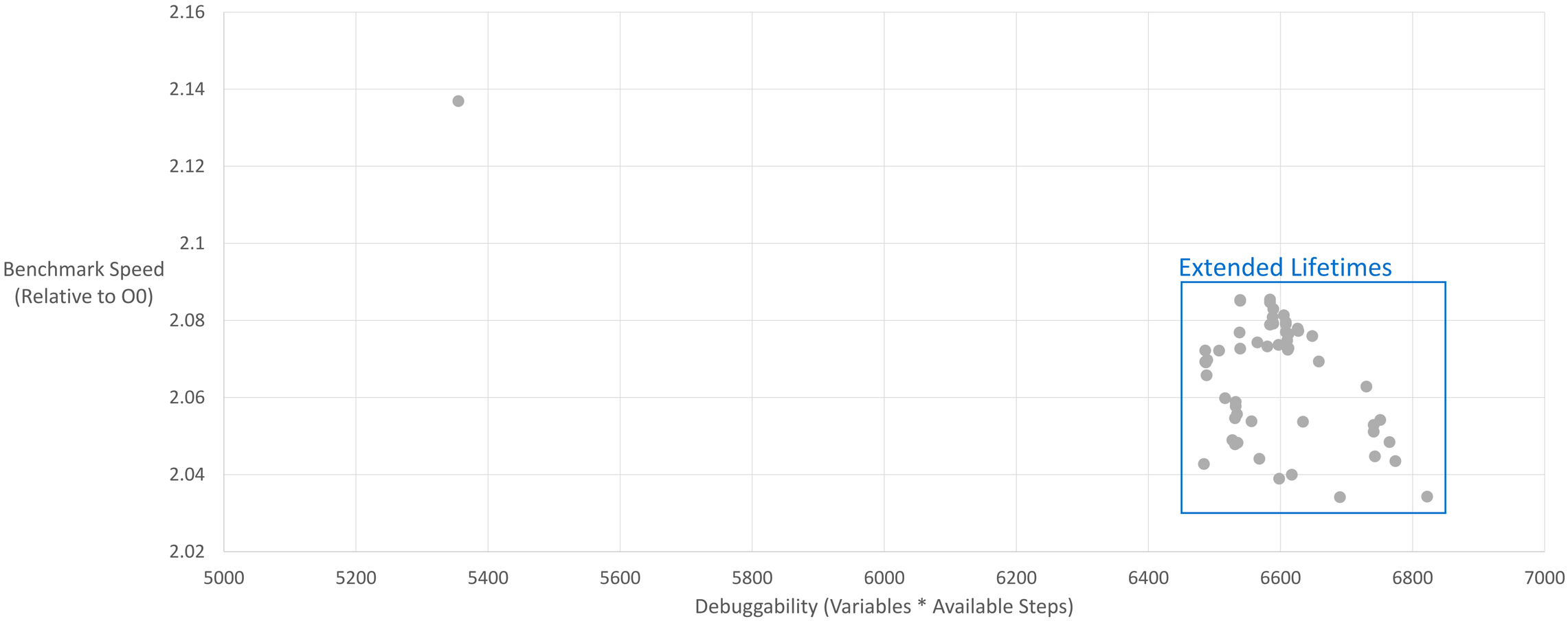
Performance vs Debug Info Quality



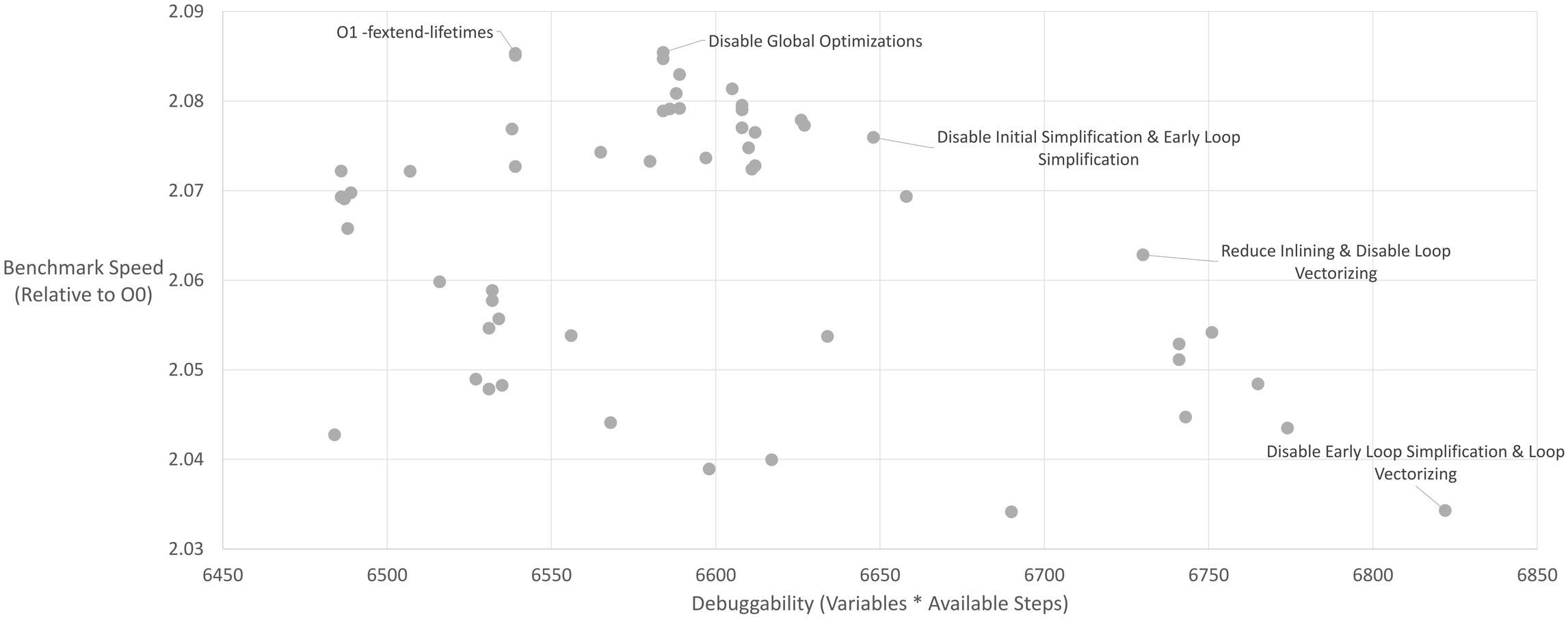
Performance vs Debug Info Quality



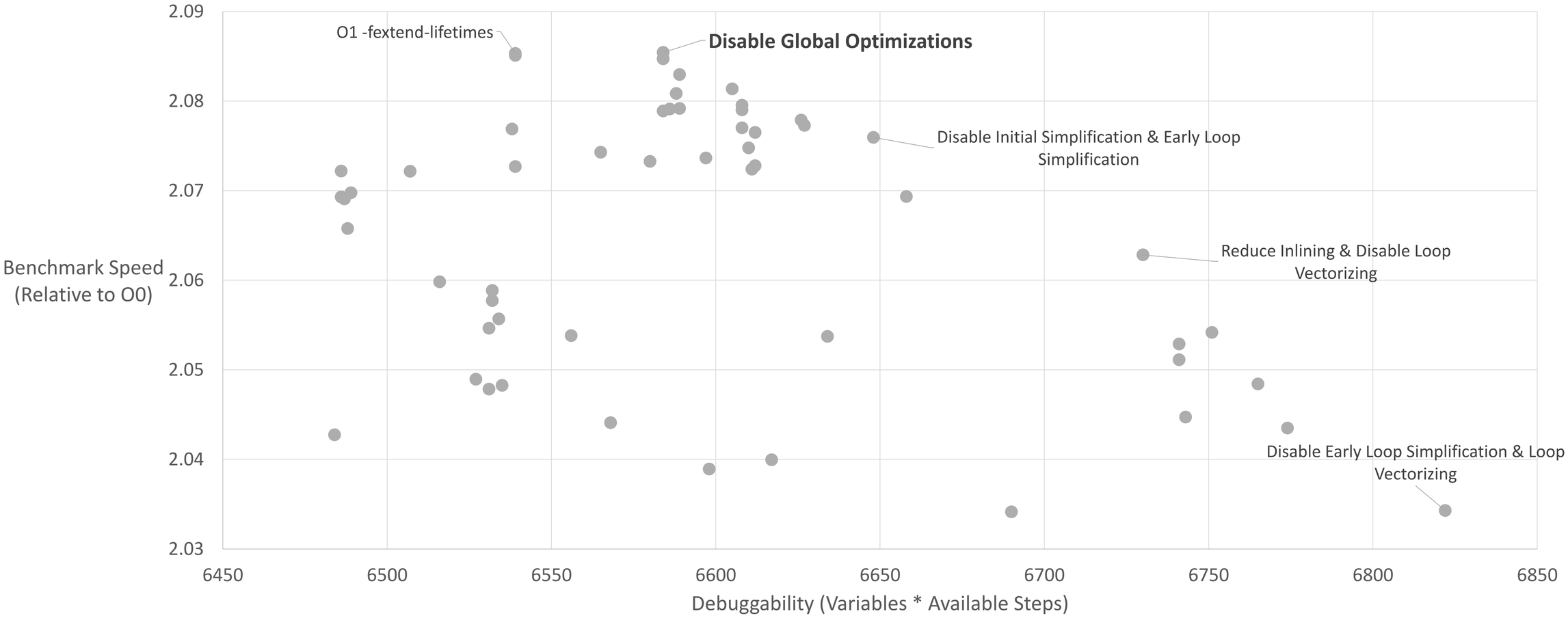
Performance vs Debug Info Quality



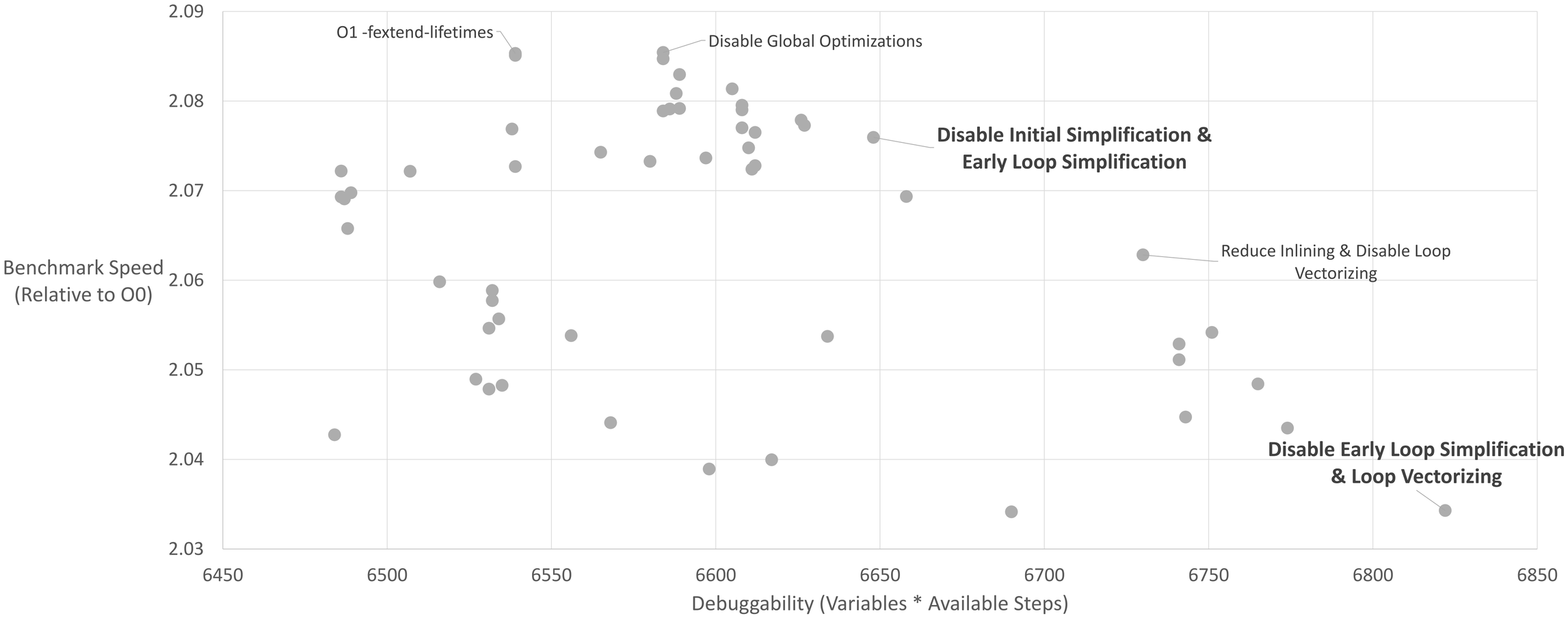
Performance vs Debug Info Quality



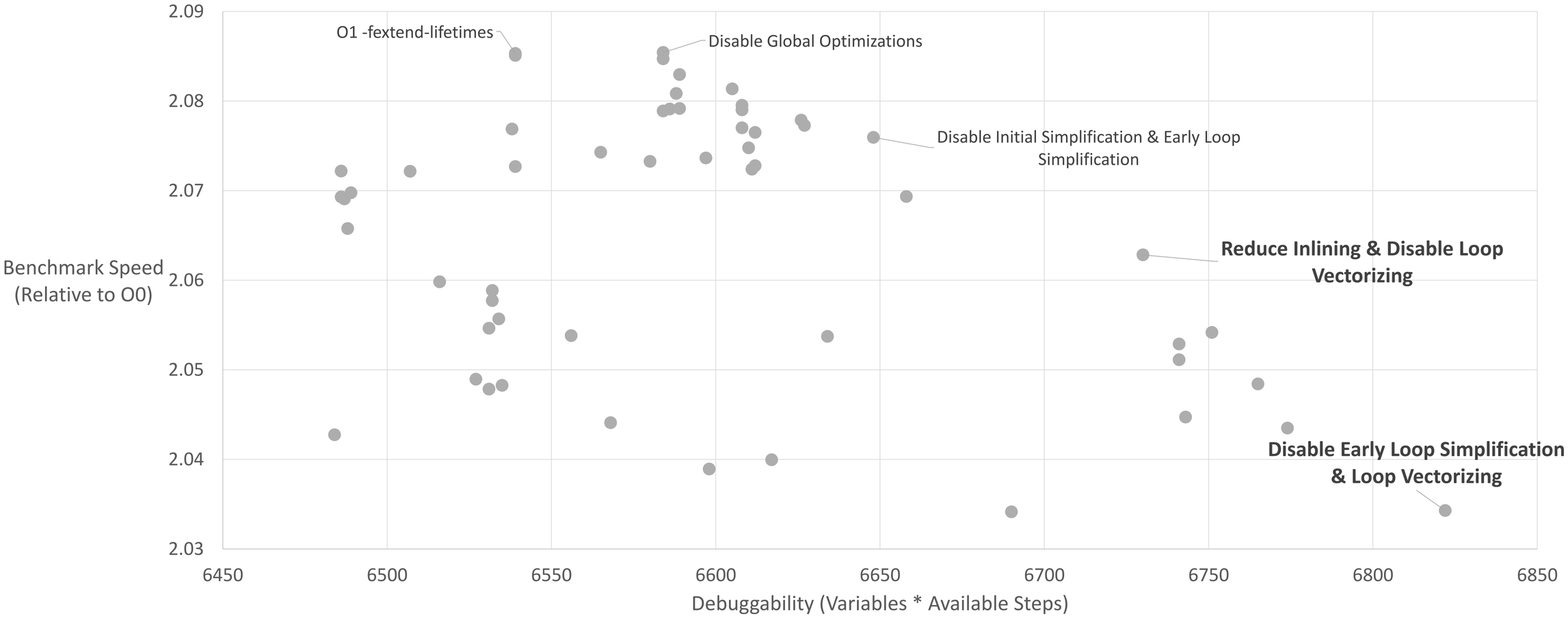
Performance vs Debug Info Quality



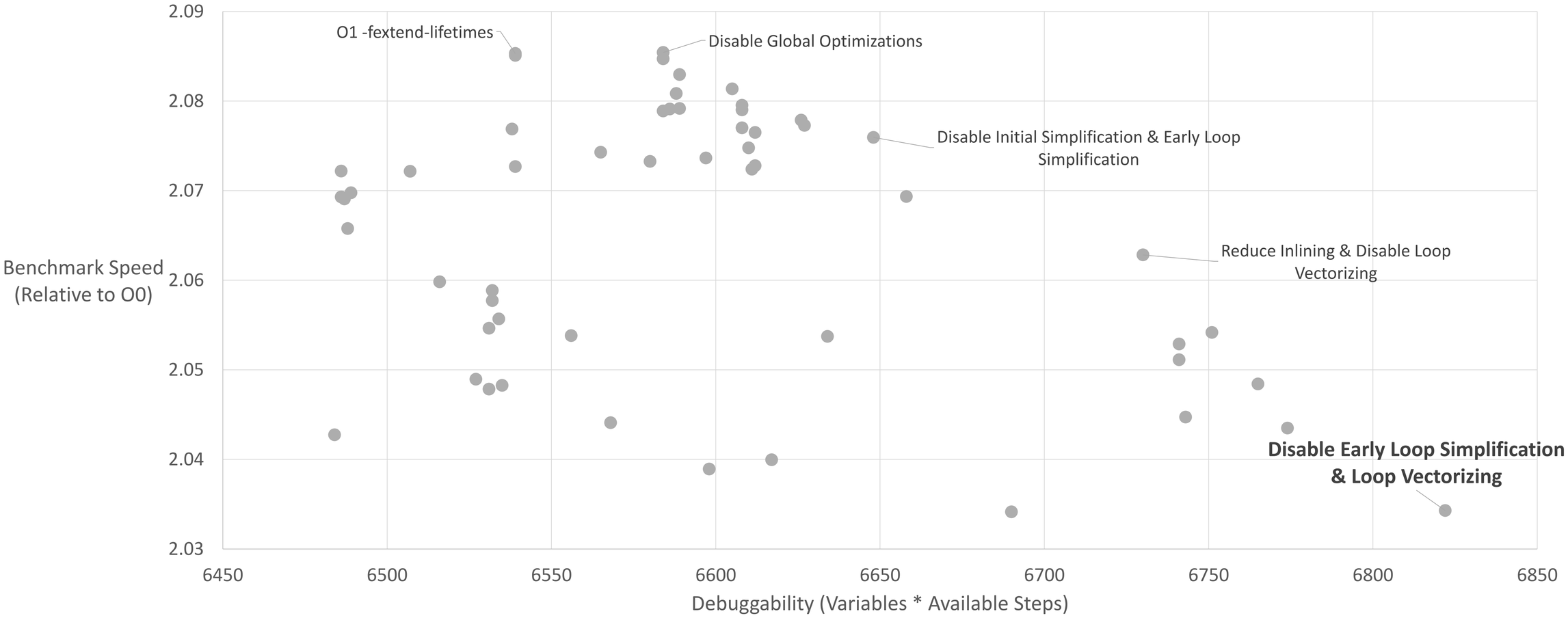
Performance vs Debug Info Quality



Performance vs Debug Info Quality



Performance vs Debug Info Quality



What comes next?

- Further experiments using a fine-grained exploration of the pipeline space.
- Demonstrated potential for an open-source debug info benchmark.
- Floor opened for future debug info flags similar to `-fextend-lifetimes`.
- Insights from the metrics can direct future debug info improvements.

Thank you!