1. Evaluation

This file simply typesets the benchmark results analyzed through R.

Measurement Setup. We measured how much time was required to construct indexes. For comparison, we also present the time needed for loading the bytecode processed:

Name	Elapsed time
Reading class files	3773.13±5.51
Method name	69.13±0.68
Exception handlers	179.81±2.44
Instruction type	7529.42±28.50

Below we measure in all cases the pure query time. We discuss the runtime overhead of the optimizer itself separately below.

Base vs. reference implementation. We set out to compare SQuOpt to FindBugs. Therefore, we measured *startup performance* [Georges et al. 2007], that is the performance of running the queries only once, to minimize the effect of compiler optimizations.

Name	Elapsed time	CPU time
FindBugs	43.00±0.57	90.18±1.90
SQUOPT	22.88±0.23	58.01±1.46

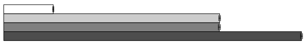
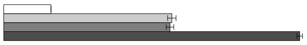
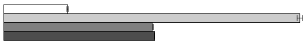
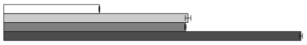
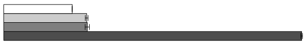
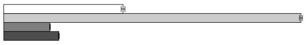
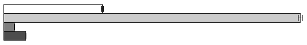
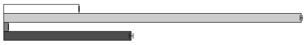
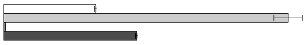
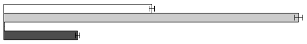
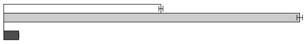
Interpretative overhead and optimization potential. We present the results of our benchmarks in Table 2. We see that, in its current implementation, SQUOPT causes an interpretation overhead between 1.2x and 5x.

References

A. Georges, D. Buytaert, and L. Eeckhout. Statistically rigorous Java performance evaluation. In *Proc. Int'l Conf. Object-Oriented Programming, Systems, Languages and Applications*, OOPSLA '07, pages 57–76, New York, NY, USA, 2007. ACM.

Identifier	Description
PROTECTED_FIELD	Class is final but declares protected field
NO_CLONE	Class implements Cloneable but does not define or use clone method
SUPER_CLONE_MISSING	The clone method does not call super.clone()
NOT_CLONEABLE	Class defines clone() but doesn't implement Cloneable
COVARIANT_COMPARETO	Covariant compareTo() method defined
GC_CALL	Explicit garbage collection; extremely dubious except in benchmarking code
RUN_FINALIZERS_ON_EXIT	Method invokes dangerous method runFinalizersOnExit
COVARIANT_EQUALS	Abstract class defines covariant equals() method
FINALIZER_NOT_PROTECTED	Finalizer should be protected, not public
UNUSED_PRIVATE_FIELD	The value of a private field is not read
DONT_CATCH_IMSE	Dubious catching of IllegalMonitorStateException

Table 1. Implemented Analyses

Name	Base impl. (in ms)	Optimiz. time	IS	OS	OS-Opt	Performance (relative)
SUPER_CLONE_MISSING	13.59±0.17	22.57±0.16	0.2x	0.2±0x	0.2±0x	
PROTECTED_FIELD	2.02±0.00	5.55±0.04	0.3x	0.3±0x	0.2±0x	
UNUSED_PRIVATE_FIELD	422.89±4.59	10.46±0.14	0.2x	0.4±0x	0.4±0x	
NO_CLONE	3.32±0.02	4.01±0.03	0.5x	0.5±0x	0.3±0x	
COVARIANT_COMPARETO	4.30±0.01	13.45±0.12	0.8x	0.8±0x	0.2±0x	
GC_CALL	229.46±3.62	17.11±0.10	0.4x	2.6±0x	2.2±0x	
RUN_FINALIZERS_ON_EXIT	167.07±1.75	19.34±0.14	0.3x	9.2±0.2x	4.5±0x	
NOT_CLONEABLE	12.40±0.05	20.22±0.32	0.3x	16±0.2x	0.6±0x	
COVARIANT_EQUALS	12.26±0.16	17.45±0.16	0.3x	47.6±0.8x	0.7±0x	
FINALIZER_NOT_PROTECTED	19.49±0.36	9.61±0.27	0.5x	198.3±6.2x	2±0.1x	
DONT_CATCH_IMSE	100.05±1.33	9.56±0.04	0.5x	2935.8±146.6x	10.4±0.1x	

Performance given as average±standard deviation in milliseconds; plot whiskers denote standard deviation
IS: interpretation slowdown for SQ (bigger is better) OS: optimization speedup for SQuOpt (bigger is better)
OS-Opt: optimization speedup, considering the optimization time (SQuOpt + Opt) (bigger is better)
The plot shows performance relative to the slowest performance: □ Base ░ SQ ▒ SQuOpt ▓ SQuOpt + Opt

Table 2. Performance measurements (in ms)