Evaluating the *egg* Equality Saturation Superoptimizer

Luca Hagemans (l.c.hagemans@student.tudelft.nl)

Supervisors: Dennis Sprokholt & Soham Chakraborty

1. Background

Superoptimization is the idea of optimizing a given input program into the most optimal program possible that has the same functionality [1].

Equality Saturation is a superoptimization technique that applies rewrite rules to an e-graph until saturation or timeout, then extracts the optimal program [2].

Rewrite Rule: Dictates a left hand side and a right hand side that achieve the same behaviour such as $a \cdot 2 \implies a << 1$.

Implied Rewrite Rule: A rewrite rule that can be applied indirectly via multiple other rewrite rules.

E-graph: An e-graph is a graph structure existing of e-classes and e-nodes, where e-classes contain one or multiple e-nodes. E-nodes are pieces of code with e-classes as children. Every e-node in an e-class achieves the same result. Figure 1 shows the saturation of an e-graph for the input program of $(a \cdot 2)/2$.

Egg is an open source[1] e-graph implementation in Rust, capable of equality saturation. Introduced in [3], it iterates on equality saturation (eq-sat) proposed in [2]. *Egg* is the basis for extensions such as Herbie, Diospyros and Tensat.

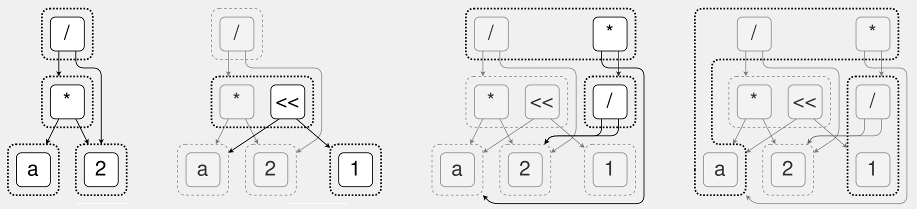


Figure 1. **Verbatim taken from** [3] An e-graph of the program $(a \cdot 2)/2$

[1] https://egraphs-good.github.io/

2. Research question

Is the *egg* superoptimizer quicker when provided with more rewrite rules?

We answered the research question using the following sub-questions:

1. Can the improvement claims from [3] be reproduced?
2. Does *egg* speed up or slow down when provided with extra unused rewrite rules?
3. Does *egg* speed up or slow down when provided with inferable rewrite rules?

3. Method

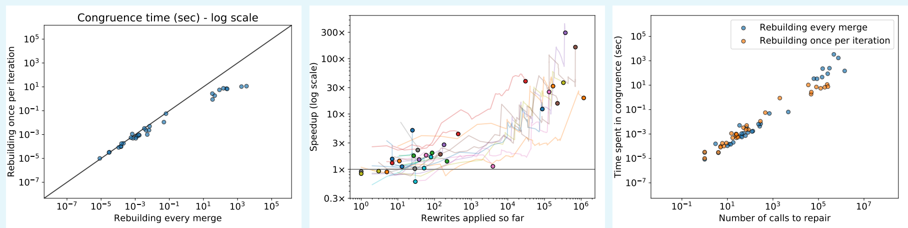
- First sub-question was answered by the research artefact of [3]. It produces plots and values to compare to the original.
- Second sub-question was answered by writing small tests and providing either all or selected rewrite rules.
- Third sub-question was answered by writing small tests and providing either selected and implied or only selected rewrite rules.

The research artefact was run 10 times. Tests for sub-questions 2 and 3 were run 5 times for both rewrite rule sets, each run created 200 data-points.

4. Reproduced results

Reproducing the results gave values and graphs to compare with those given in [3]. The reproduced graphs are given in Figure 2, the reproduced values are:

1. The correlation coefficient (r_s) of amount of times the e-graph is restored with the amount of time taken, the claimed value for r_s is .98. Our 95% confidence interval for r_s is **(.976, .982)**.
2. The speedup restoring the e-graph of *egg* over original eq-sat, the claimed value for this speedup is 88×. Our 95% confidence interval for this speedup is **(87.7, 96.7)**.
3. The overall speedup of *egg* over original eq-sat, the claimed overall speedup is 21×. Our 95% confidence interval for the overall speedup is **(16.4, 17.4)**.



(a) Time spent repairing e-graph, points below $x = y$ mean *egg* is faster than original eq-sat

(b) The speedup of *egg* over original eq-sat for different amount of rewrite rules applied with trace

(c) Correlation between the amount of times the e-graph is restored and time spent in restoring the e-graph

Figure 2. Reproduced graphs from the first run, showing similar results to those in [3]

5. Results of testing *egg*

A 95% confidence interval was calculated for each test with 200 data-points. A minimum speedup/maximum slowdown was calculated using the high and low bounds of the confidence interval, which was averaged over 5 runs.

- When removing unused rewrite rules, two different tests resulted in a **44%** and **3.84%** minimum average speedup.
- When adding an implied rewrite rule, two test cases resulted in a **14.4%** and **10.0%** minimum average speedup, another test case resulted in a **11.0%** maximum average slowdown.

6. Limitations

- Reproduced results were created on different hardware to original results.
- Only 10 runs of reproducing, accuracy likely to improve with more runs.
- Only 2 tests supplying all or selected rewrite rules, conclusion might not hold generally.
- Only 3 tests supplying selected or selected and implied rewrite rules, more tests could lead to different conclusion.

7. Conclusion

The claims in [3] that egg improves upon the eq-sat concept are correct, although the exact amount of speedup seems to be an overestimate. *Egg* also seems to be faster when unused rewrite rules are removed, and might benefit from including implied rewrite rules, but the results were inconclusive

References

[1] H. Massalin, "Superoptimizer: A look at the smallest program," *ACM SIGARCH Computer Architecture News*, vol. 15, pp. 122–126, 5 Nov. 1987, ISSN: 0163-5964. DOI: `10.1145/36177.36194`.

[2] R. Tate, M. Stepp, Z. Tatlock, and S. Lerner, "Equality saturation: A new approach to optimization," *Logical Methods in Computer Science*, vol. 7, 1 2011, ISSN: 18605974. DOI: `10.2168/LMCS-7(1:10)2011`.

[3] M. Willsey, C. Nandi, Y. R. Wang, O. Flatt, Z. Tatlock, and P. Panchekha, "Egg: Fast and extensible equality saturation," *Proceedings of the ACM on Programming Languages*, vol. 5, POPL 2021, ISSN: 24751421. DOI: `10.1145/3434304`.

TU Delft