Notes on the scheduling and extraction problems of EqSat

Yihong Zhang

This is a companion to my PLSE blog post on the termination problem of equality saturation. It is intended to give an overview of two other interesting problems related to equality saturation besides termination, namely extraction and scheduling.

Extraction

E-graph extraction tries to extract an optimal program out of an E-graph. This is both an easy and a hard problem: If we only consider the tree size/cost of programs (e.g., `(x + 1) * (x + 1)` has size 7), this is an easy problem, as very efficient algorithms for extraction exist. However, in practice we usually want to consider the DAG size/cost of programs (e.g., `(x + 1) * (x + 1)` has size 4). In a blog post, I showed this problem is NP-hard via a reduction from the minimal set cover problem. This problem is equivalent to the shortest path problem over hypergraphs, if we consider an E-graph as a directed hypergraph, where e-nodes are hyperedges.

Despite the NP-hardness, many people have been working on E-graph extraction. Many algorithms are proposed. For example, Peggy, SPORES, and TenSat use ILP solvers. babble and an early prototype of eggcc use a dynamic programming algorithm. Strikingly, three talks at EGRAPHS 2023 are on this topic: KestRel proposed a simulated annealing–based algorithm, Eli Rosenthal proposed a Zero-Suppressed Binary Decision Diagram (ZDD) based algorithm, and He et al. proposed a MaxSAT-based extraction algorithm. Yet it is still open to me which extraction algorithm I should use. There is an extraction gym that surveys recent extraction algorithms. A surprising finding is that, despite being much more expensive, these algorithms are not significantly better than the greedy extraction algorithm that is optimal for the tree cost on the benchmark tested. It is not clear if this is because the benchmark is too weak, or the greedy algorithm is just good enough.

One may wonder if they can develop approximation algorithms or fixed-parameter tractable algorithms for E-graph extraction. Unfortunately, the inapproximability results of the set cover show that no good polynomial-time approximation algorithm exists unless P=NP. On the other hand, we have some

early thoughts on a fixed-parameter tractable algorithm for E-graph extraction. We do not know how good it is in practice yet.

Scheduling

Another problem I am very excited about is scheduling. The naive equality saturation algorithm always applies all the rules applicable. The issue with this approach is that it may spend most of its run time in exploring exponential rules (e.g., associativity) without making progress, which is highly inefficient. A slightly better approach is taken by the `BackOff` scheduler, which bans rules from firing too often. Yet it is still very heuristic.

Equality saturation can be thought of as a search problem, and the naive scheduling strategy corresponds to the breath-first search (BFS) algorithm. A natural question then is what other classic search algorithms correspond to in equality saturation. For example, best-first search uses the current cost of a search node to guide the search. Can we do something similar in equality saturation by prioritizing E-classes or E-nodes that are more "optimal" than others? For example, if an E-class c is part of the optimal term from extraction at iteration i, prioritizing firing rules whose root is c is an obviously good strategy. On the other hand, if the path from the root to some E-class is very expensive, then spending lots of effort making such E-class super-duper-optimized is not a wise move, since it is unlikely to be included in the extracted program anyway. There seems to be some low-hanging fruit along this line of reasoning. Sketch-guided equality saturation achieves something similar by extracting intermediate forms and starting again, an idea RisingLight independently came up with. This approach biases equality saturation towards more profitable explorations and is similar to beam search by way of analogy. Other recent approaches include MCTS-GEB, which uses reinforcement learning for equality saturation scheduling, but it requires many calls to equality saturation, so it is unclear how useful it is in practice.

As a related note, the scheduling problem is also related to proofs in equality saturation, since the shortest proof essentially gives the optimal rewrite sequence to the target term. I think this is very promising, because, unlike the Monte Carlo Tree Search approach used by MCTS-GEB, constructing a fairly good proof requires only one call to equality saturation and is relatively efficient.

Another direction I explored in my EGRPAHS 2023 talk is how to design schedulers with *guarantees*. The naive scheduler indeed provides the guarantee that if two terms can be shown equivalent by rewriting each term for n steps, they can be shown equivalent within n iterations of naive equality saturation. The talk introduced two more schedulers, merge-only scheduler and depth-bounded scheduler, both of which always terminate with the following guarantee: if two terms can be shown equivalent using only terms in the initial E-graph (resp. terms with a bounded depth), the equivalence can be shown with the given scheduler. These schedulers are arguably less powerful than the naive scheduler, but they can be useful in ensuring both the termination and completeness

properties of systems and, when composed with other scheduling strategies (e.g., search-inspired ones mentioned above), can be very efficient.