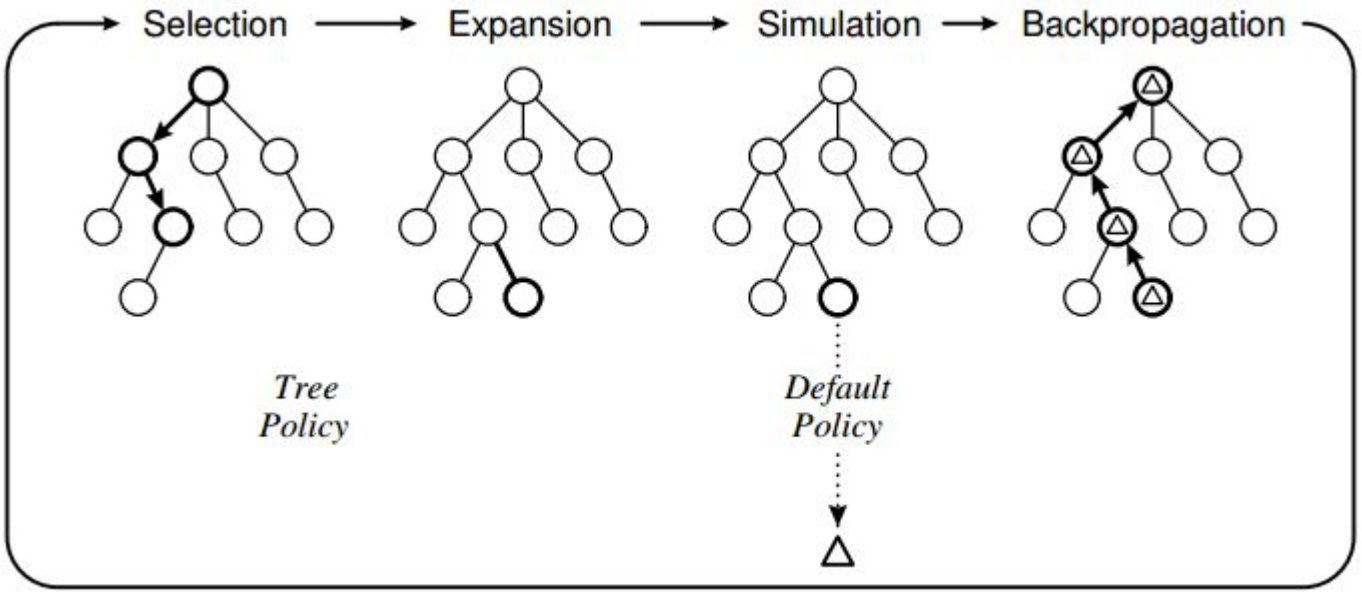
(Better??) E-graph Extraction

Jeremy Ferguson, Sora Kanosue, Jacob Yim

Background: Monte Carlo Tree Search



Background: DAG vs. Tree Cost

- DAG cost accounts for shared nodes while tree cost does not
- But minimizing DAG cost is difficult
- Use stochastic methods to search for minimum cost DAG

Background: Existing Extraction Methods

- Bottom-up: uses tree cost, may not find optimal solution in egraphs involving sharing
- Greedy DAG: uses DAG cost but not guaranteed to find optimal solution
- ILP: uses DAG cost and guaranteed to find optimal solution, but may not terminate

Algorithm Overview

Key insight: At each node, we pick an eclass, and a node from that eclass.

Improvements: Monte Carlo DAG Search

Certain nodes become redundant

Merge them based on decided eclasses and to-visit eclasses.

Tree becomes a DAG

Improvements: Warmstarting

We can pre-populate the our DAG to waste less rollouts using another Extractor

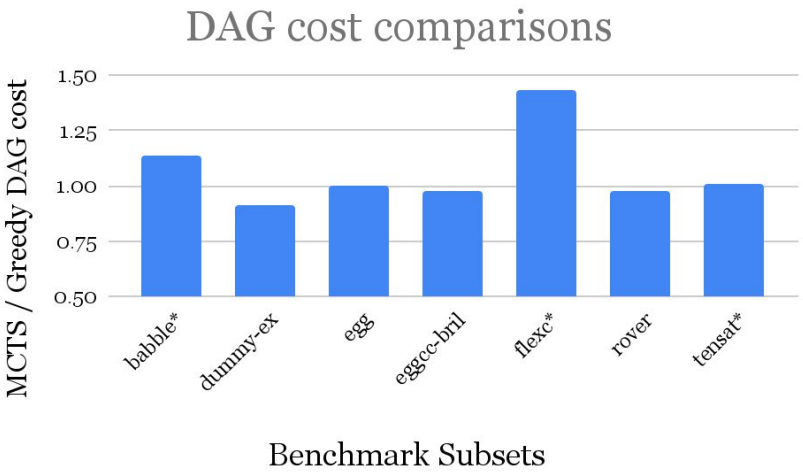
Guarantees that our worst case result is the one returned by another extractor.

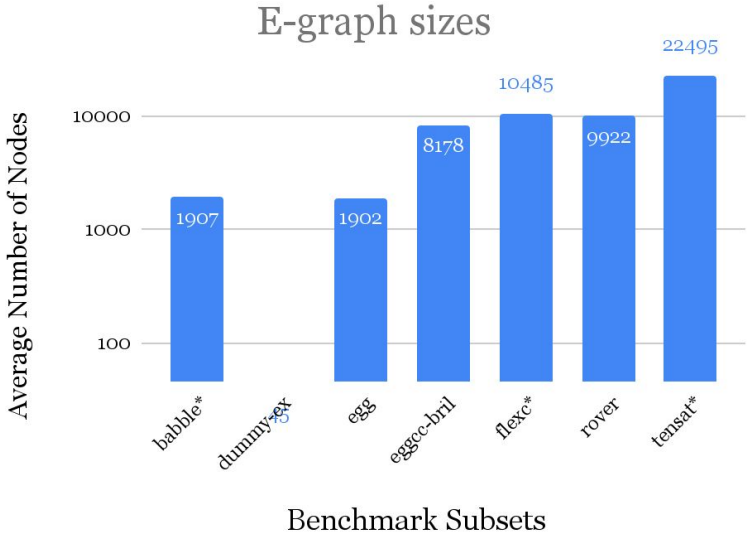
Improvements: Solution-Guided Rollouts

- Originally: choose nodes randomly during a rollout
- Key Insight: We can guide it by providing it with some prior solution
  - Can't always choose the prior solution - then we wouldn't see any improvements
- Algorithm:

```
Function randomRolloutChoice(to_visit){
    e_class = chooseRandom(to_visit)
    p = random(0,1)
    If p > α then
        e_node = chooseRandom(e_class.nodes)
    Else
        e_node = greedySolution[e_class]
    return e_node, e_class
}
```

Evaluation





Parameters: 10,000 rollouts, warm-start of 5000, Rollout choice probability: 0.2

Future Work

- Rollout selection improvements
- Leaf node selection improvements
- Future evaluations:
  - Lots of knobs to turn in our solution:
    - Number of rollouts
    - Size of warm-started graph
    - Probability of choosing random rollout node vs guided