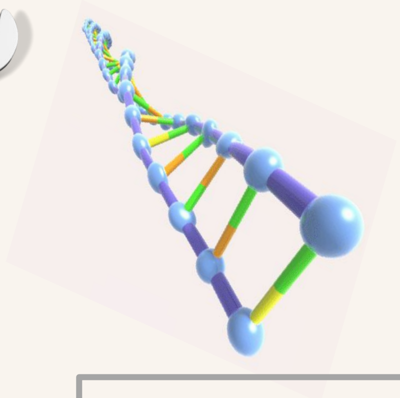
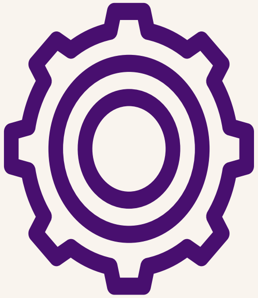
Enhancing Software Runtime with Reinforcement Learning-Driven Mutation Operator Selection in Genetic Improvement

Damien Bose and **Carol Hanna** and **Justyna Petke**

GI Workshop @ ICSE 2025

Search-based runtime improvement of software



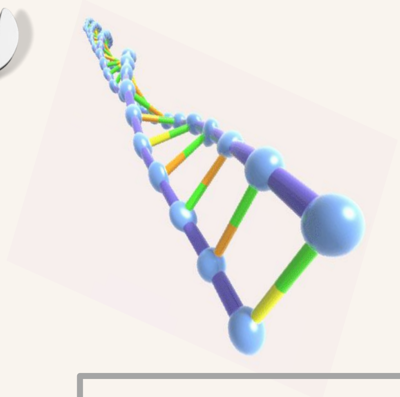


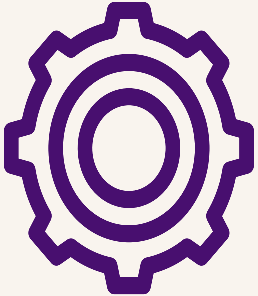
Select Operator

Create Variant

Check Fitness

Reinforcement learning aided mutation operator selection





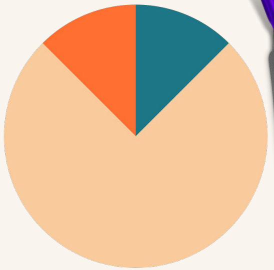
Select Operator

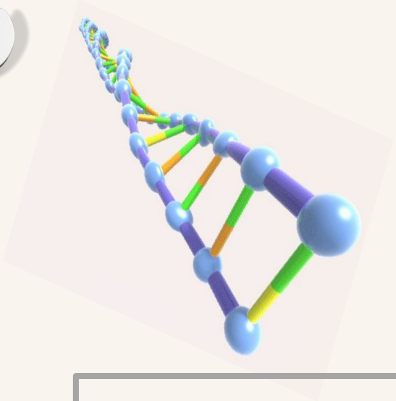
Create Variant

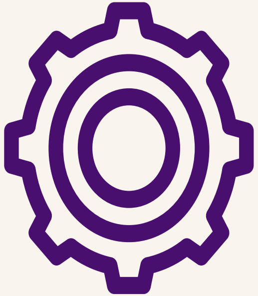
Check Fitness

Assign Credit

Reinforcement learning aided mutation operator selection

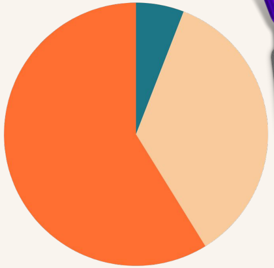


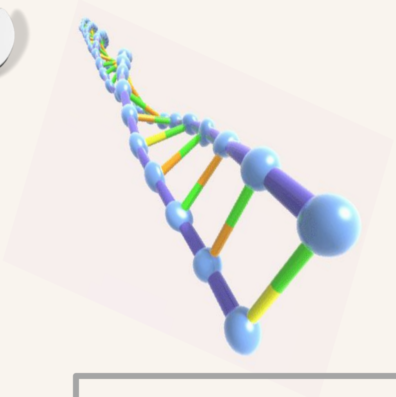


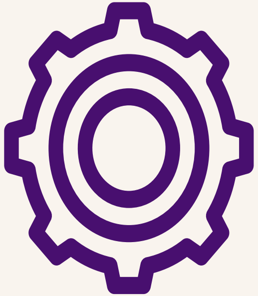


Select Operator → Create Variant → Check Fitness → Assign Credit

Reinforcement learning aided mutation operator selection







Select Operator

Create Variant

Check Fitness

Assign Credit

Operator Selection

We experiment with 4 operator selection algorithms:

Probability Matching	*Upper Confidence Bound*
Epsilon-Greedy	*Policy Gradient*

Speaker: Carol Hanna

Operator Selection

We experiment with 4 operator selection algorithms:

Probability Matching	***Upper Confidence Bound***
Epsilon-Greedy	*Policy Gradient*

[1] Hanna, C., Blot, A. & Petke, J. Reinforcement learning for mutation operator selection in automated program repair. *Autom Softw Eng* 32, 31 (2025)

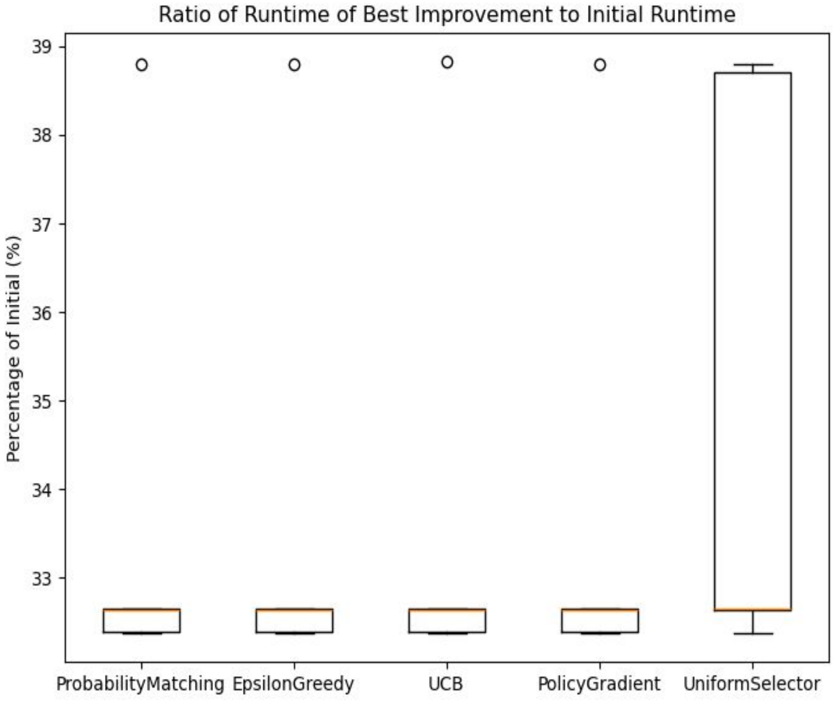
Methodology

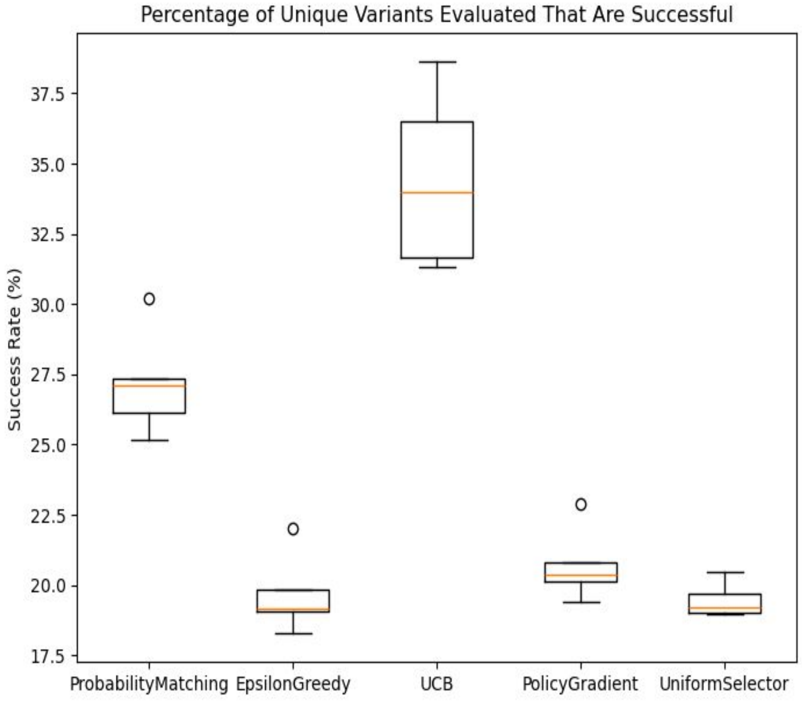
- Tool: Magpie
- Benchmark: MiniSAT (1000 test instances)
- 20 test training set (for fitness during search) and 980 validation set for checking true runtime improvement
- 2 search strategies: neighborhood search and hill climbing
- 5 repetitions
- Limited time budget
- Metrics:
  - Best runtime improvement found (ratio of new runtime to original runtime)
  - Percentage of unique variants evaluated that are successful

Speaker: Carol Hanna

Which operator selection strategy leads to the best **efficacy and efficiency** of search for **Neighbourhood Search** and **Hill Climbing**?

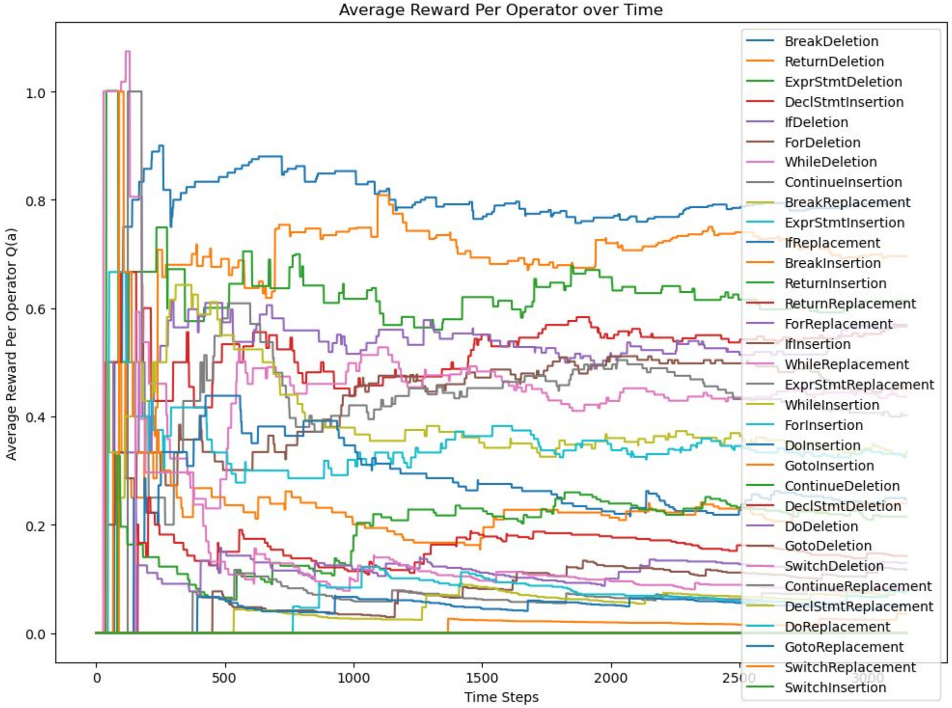
RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?

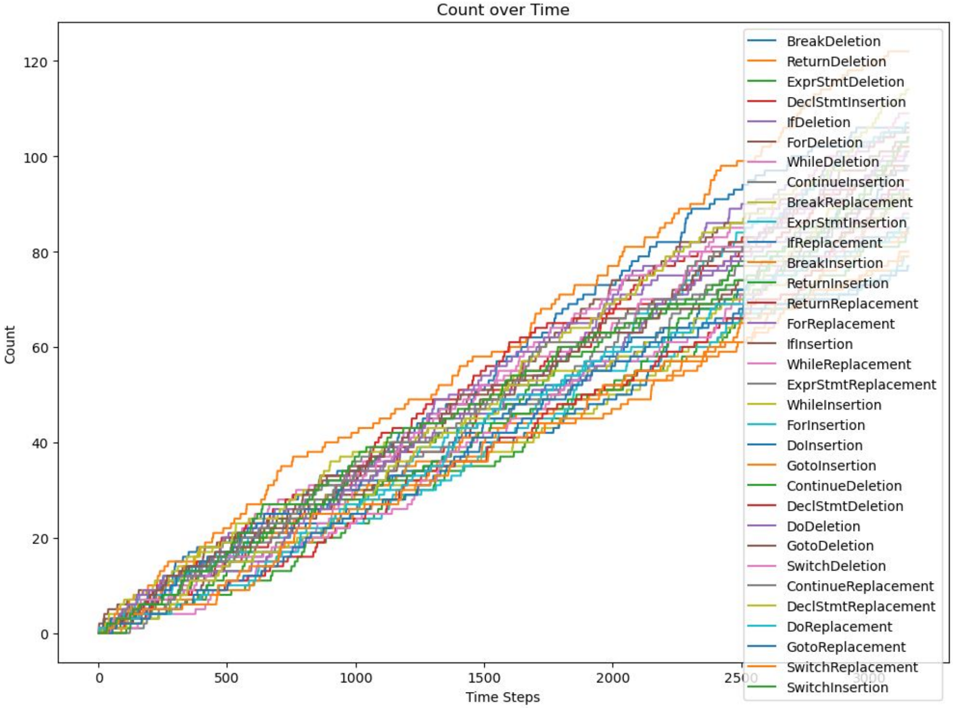




Speaker: Carol Hanna

RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?

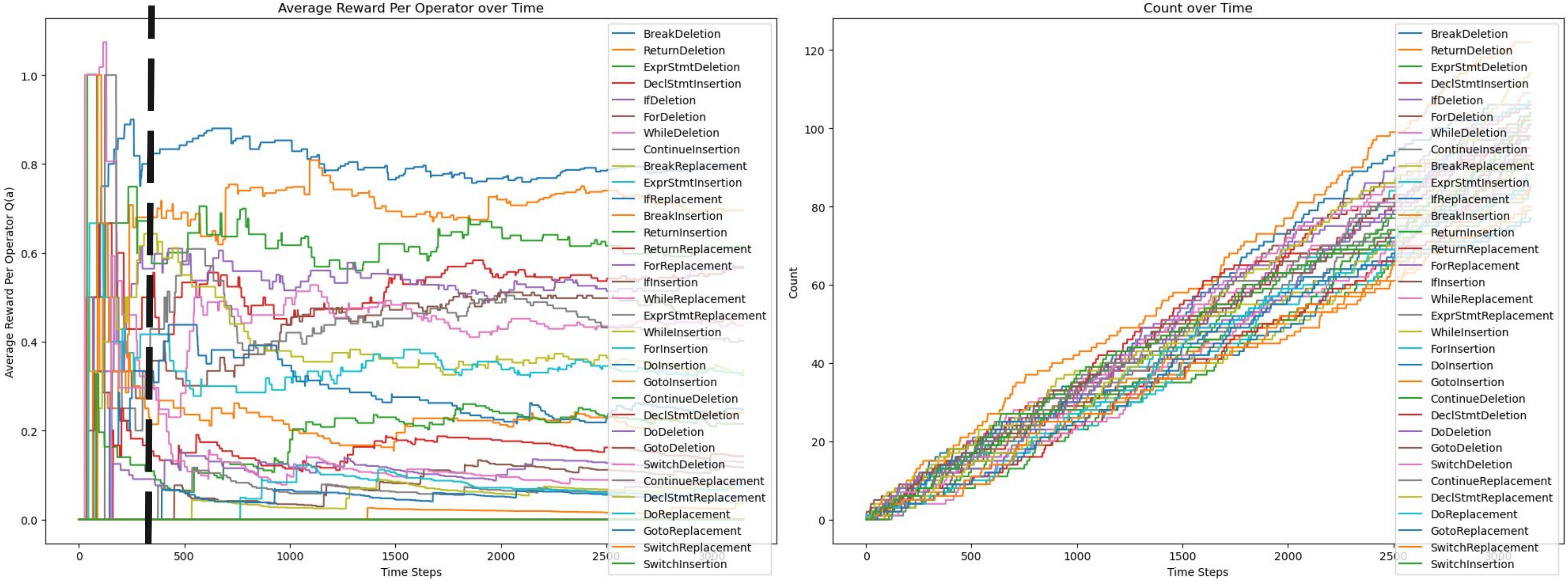




Run statistics for Uniform Selector

Speaker: Carol Hanna

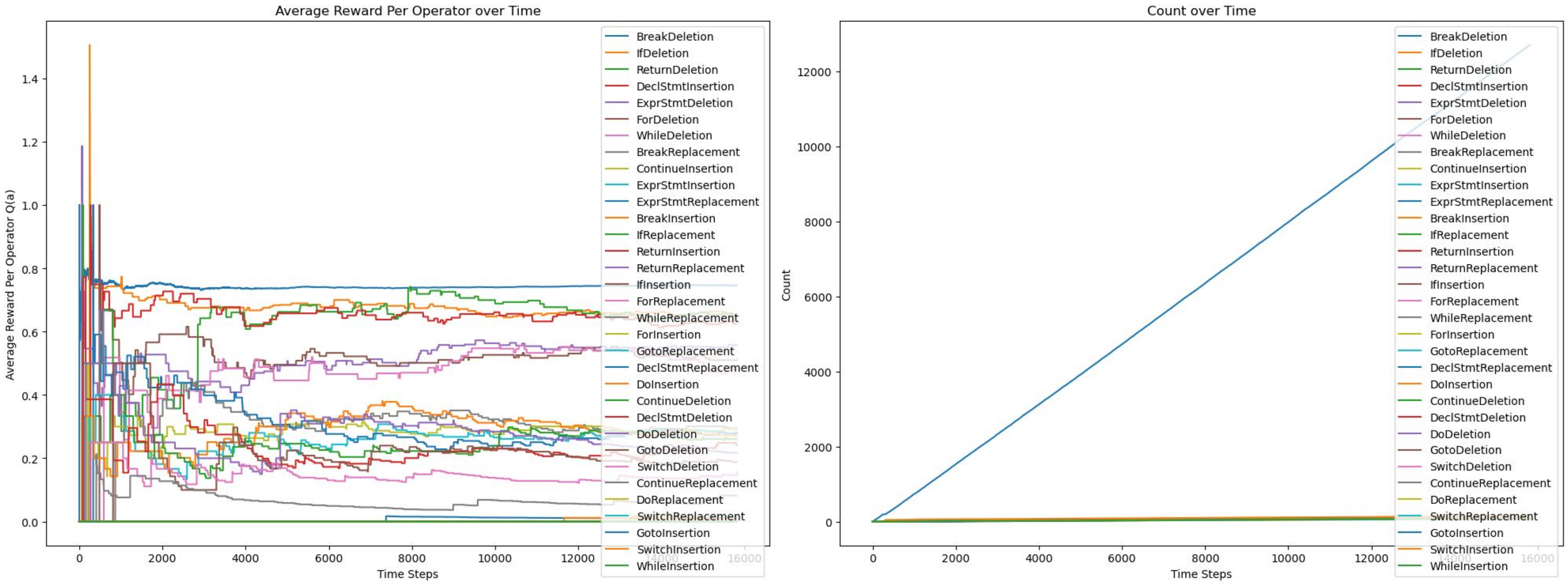
RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?



Run statistics for Uniform Selector

Speaker: Carol Hanna

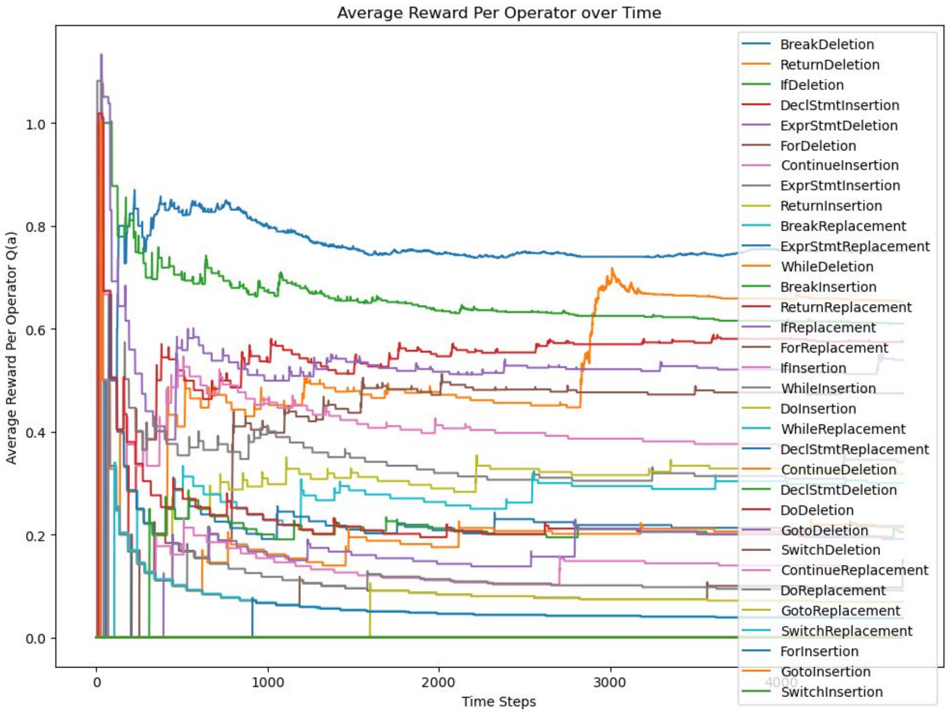
RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?

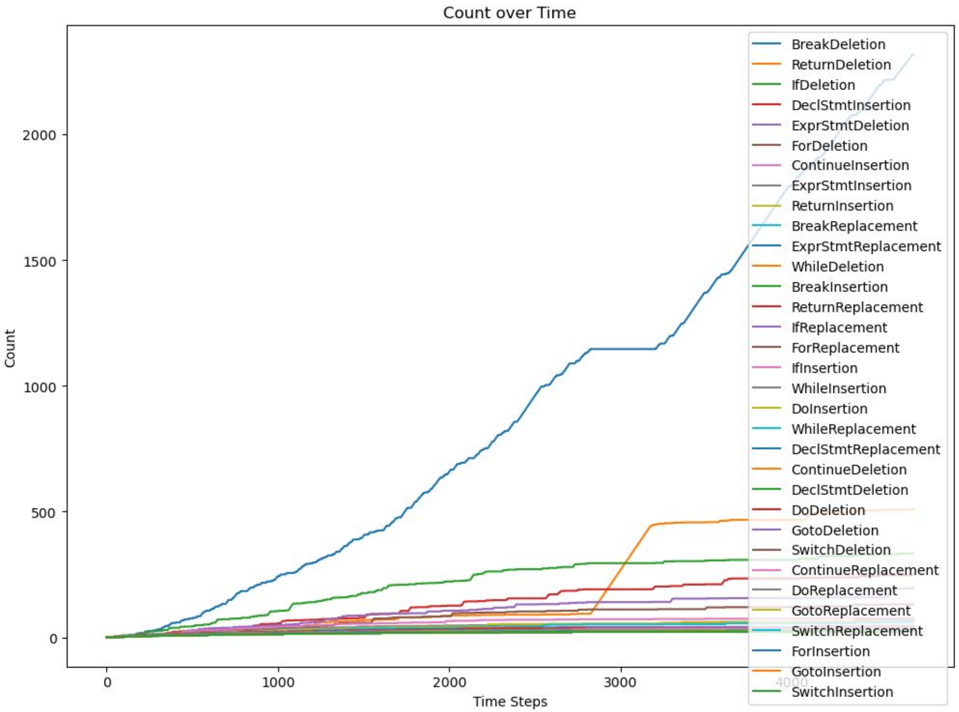


Run statistics for Epsilon Greedy

Speaker: Carol Hanna

RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?

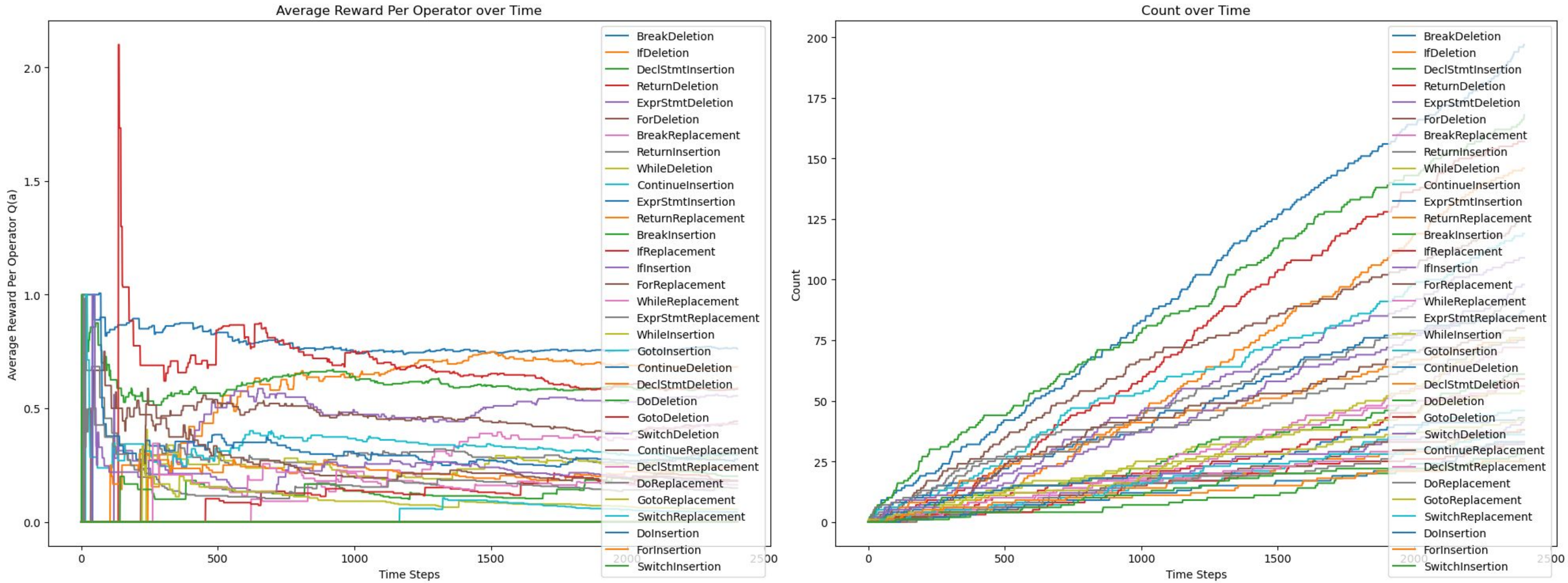




Run statistics for UCB

Speaker: Carol Hanna

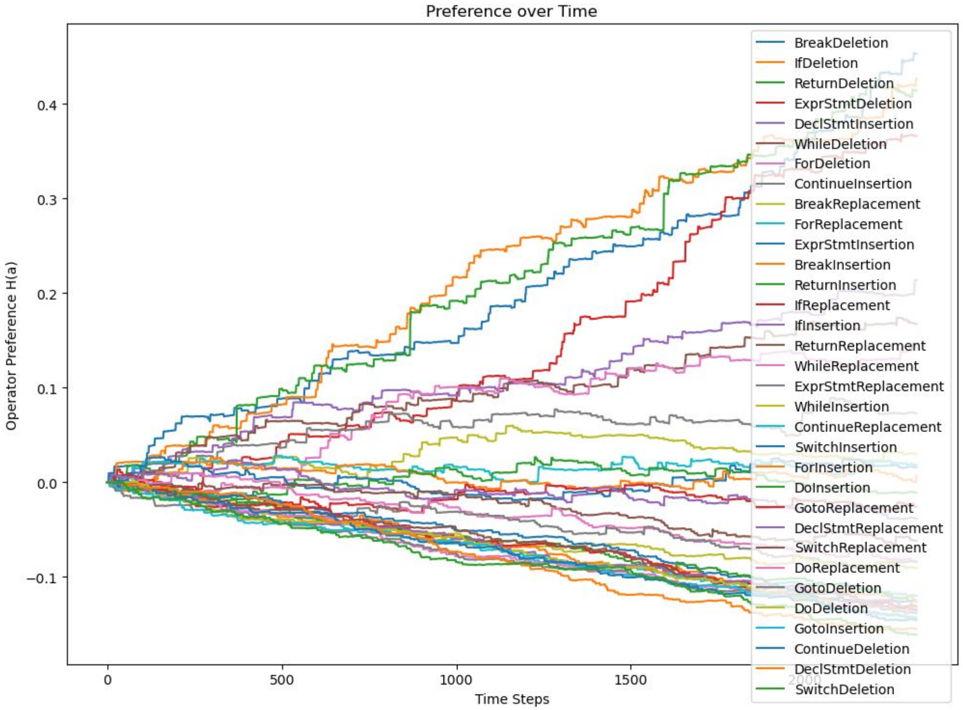
RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?

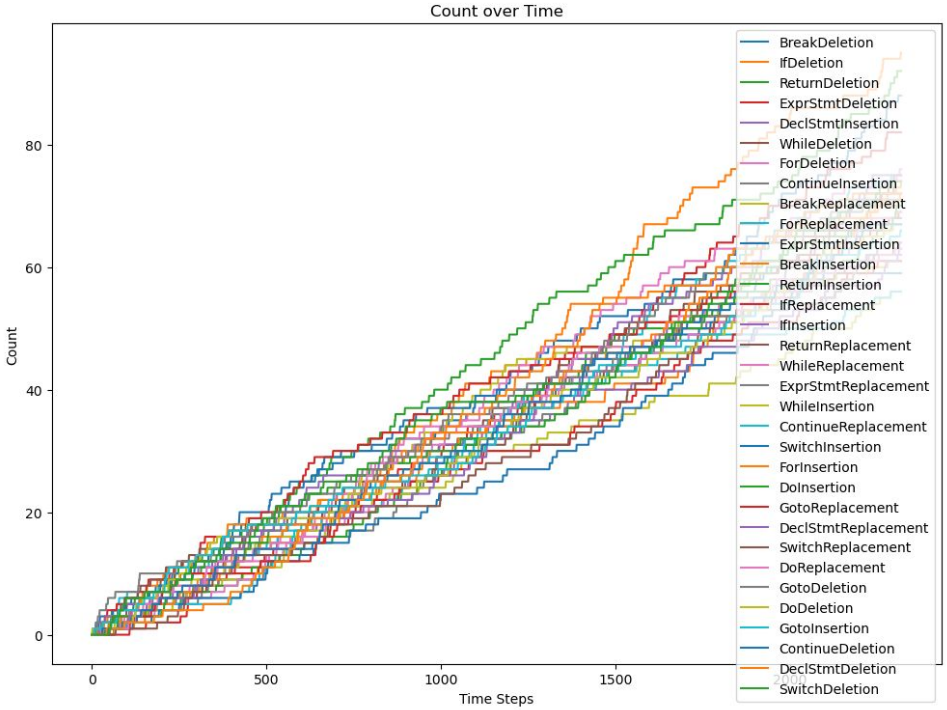


Run statistics for Probability Matching

Speaker: Carol Hanna

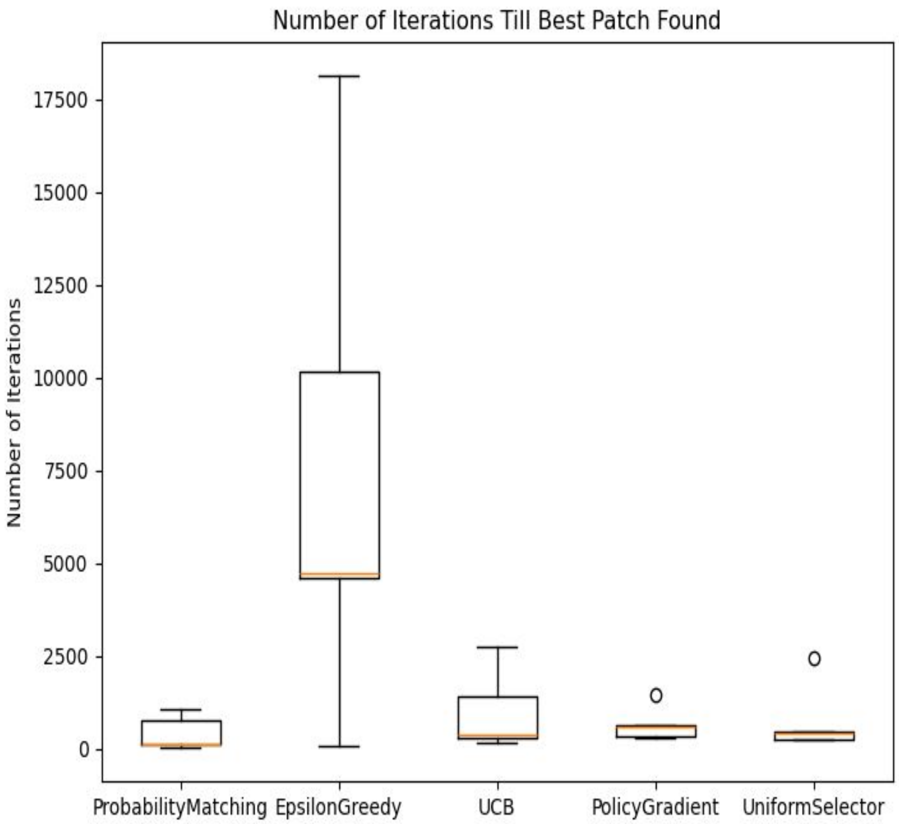
RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?





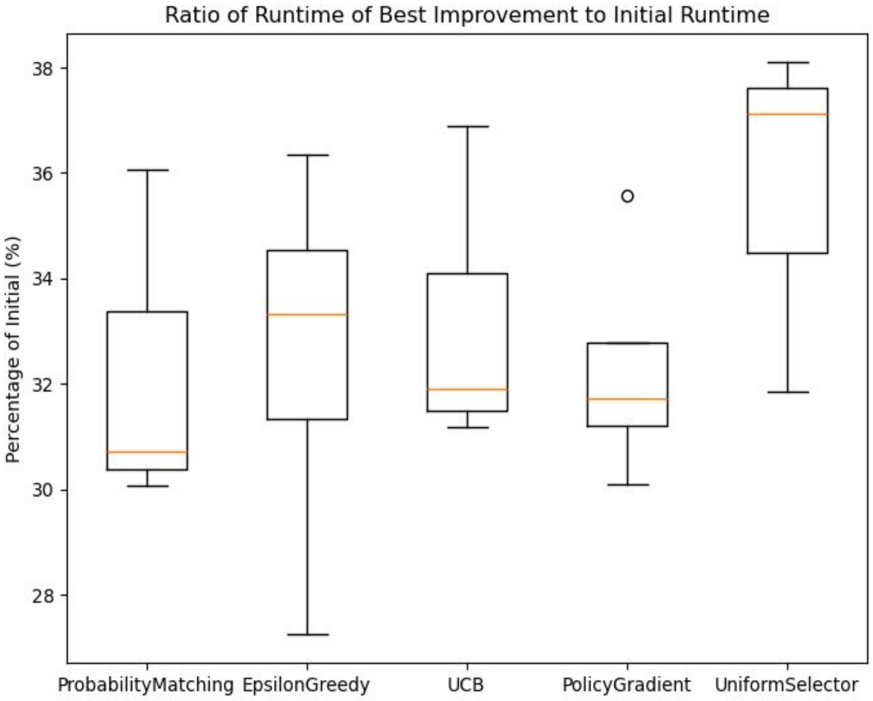
Run statistics for Policy Gradient

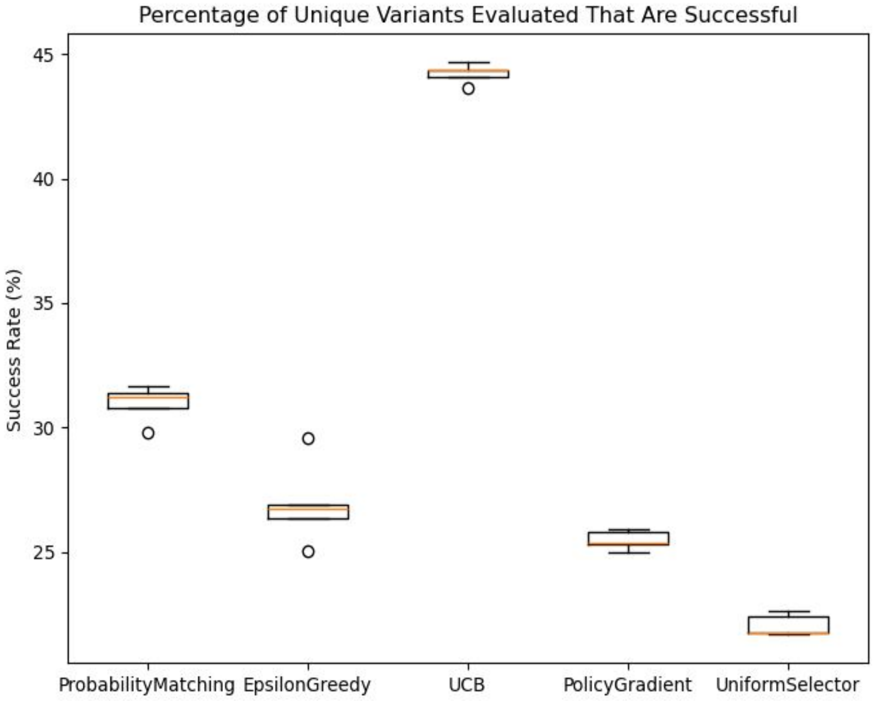
RQ1: Which operator selection strategy leads to the best efficiency of search for Neighbourhood Search and Hill Climbing?



Speaker: Carol Hanna

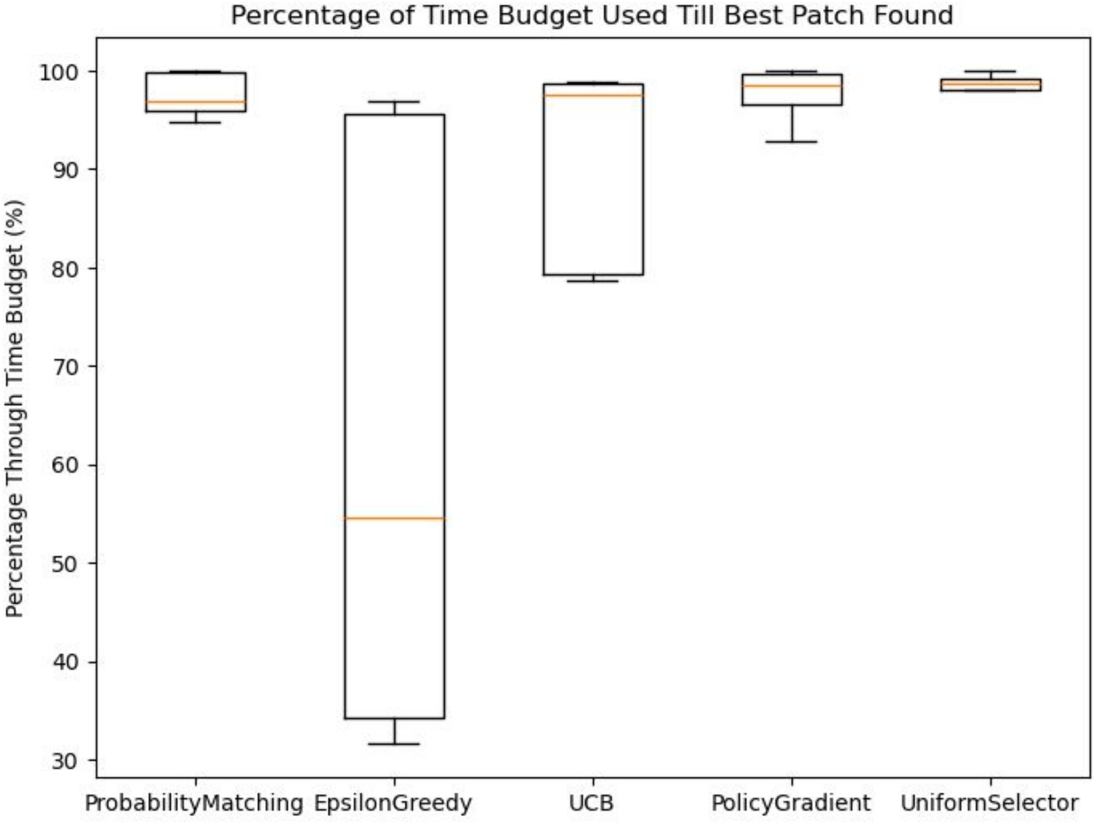
RQ2: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?





Speaker: Carol Hanna

RQ2: Which operator selection strategy leads to the best efficiency of search for Neighbourhood Search and Hill Climbing?



Speaker: Carol Hanna

Discussion

- The results for Hill Climbing are similar to those with the Neighbourhood Search experiments. E.g. BreakDeletion, ReturnDeletion still have the highest average reward

- The best edit found took only 27.24% of the original runtime to evaluate the 980 test instances in the validation split.

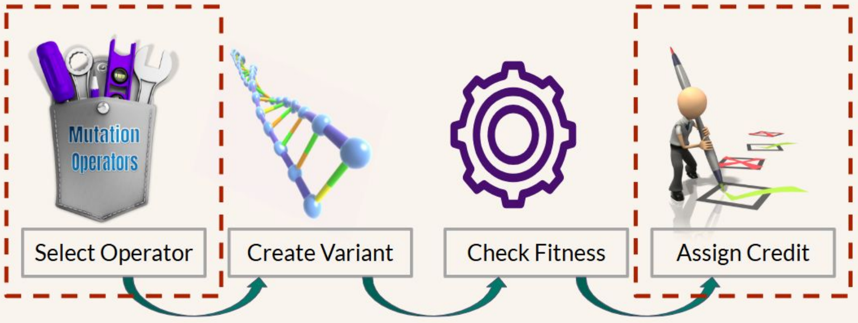
- Test-suite passing vs Manual analysis

- All operator selectors heavily value code deletion as is common with GI for runtime improvement (e.g. none of the test cases checks for exceptions, so the assert statements are redundant and thus deleted)

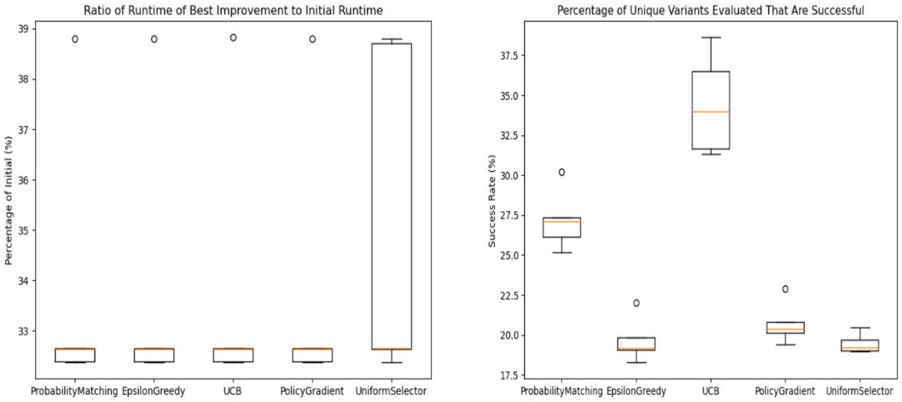
- Generalizability of MiniSAT benchmark

- Hyperparameter tuning

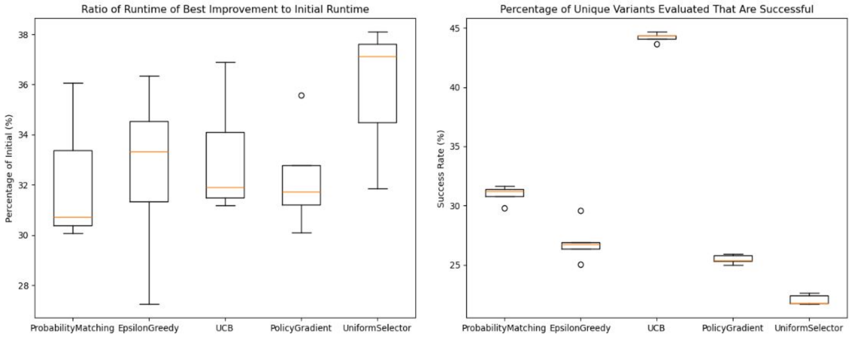
Reinforcement learning aided mutation operator selection



RQ1: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?



RQ2: Which operator selection strategy leads to the best efficacy of search for Neighbourhood Search and Hill Climbing?



- The results for Hill Climbing are similar to those with the Neighbourhood Search experiments. E.g. BreakDeletion, ReturnDeletion still have the highest average reward
- The best edit found took only 27.24% of the original runtime to evaluate the 980 test instances in the validation split.
- Test-suite passing vs Manual analysis
- All operator selectors heavily value code deletion as is common with GI for runtime improvement (e.g. none of the test cases checks for exceptions, so the assert statements are redundant and thus deleted)
- Generalizability of MiniSAT benchmark
- Hyperparameter tuning

SOLAR UCL

Carol Hanna
carol.hanna.21@ucl.ac.uk

EXTRA SLIDES

Value-based methods: focus on learning how good each action is in a given situation.

"How good is each action?"

Policy-based methods: focus on learning the policy directly; learning what action to take in each state. They don’t estimate values. Instead, they directly learn a policy function $\pi(s)$, which maps states to actions.

"What action should I take?"