Monte Carlo Tree Search for Deterministic Finite Automaton Inference

A New Search Strategy in the Red-Blue Framework

Author: B. Zandvliet (B.zandvliet@student.tudelft.nl) Supervisor: Dr.ir. S.E. Verwer

TU Delft

Introduction

- A behavioural model (DFA) constructed from system observations aids verification, testing, and anomaly detection
- Inferring a minimal DFA from observations is NP-complete
- The red-blue framework constructs a DFA by sequentially applying merge and extend refinements
- The ordering of these refinements determines the final DFA size
- Monte Carlo Tree Search (MCTS) is proposed as a heuristic to find smaller DFAs

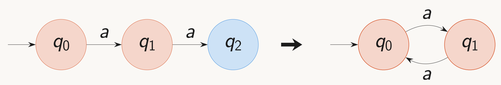
Research Question

How do different action-selection policies and reward metrics in the Monte Carlo Tree Search affect the compactness of Deterministic Finite Automata produced by the red-blue framework?

Red-Blue Framework

Constructs a DFA by applying refinements to **red** (final) and **blue** (candidate) states, minimising the number of states.

Extend: *promotes a* **blue** *state to* **red**.



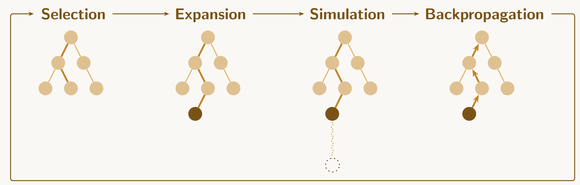
Merge: *combines q_2 into q_0, reducing the automaton size.*

The ordering of these refinements determines the final DFA size. Merge refinements are scored by the EDSM evaluation function; extends receive a score of 0.

MDP Formalisation

The red-blue framework is modelled as an MDP $M = \langle S, A, T, R \rangle$, where states are DFAs with red-blue partitions, actions are consistent refinements, transitions apply a refinement to a state, and the reward is evaluated at the terminal state. Thus, MCTS can be applied to the red-blue framework.

Monte Carlo Tree Search



Selection

Traverse explored nodes based on rewards to find the most promising node which has an unsimulated refinement.

Expansion

A child node is created by applying an untried refinement, selected greedily by highest merge score.

Simulation

A rollout is a sequence of refinements. At each step, a refinement is sampled by a policy:

Uniform

Selects uniformly at random over all available refinements.

Uniform-Merge-First

Selects uniformly over merges if available, otherwise uniformly over extends.

Weighted

Selects refinements proportional to EDSM score, where extends are weighted by the *average* or *minimal* merge score.

Weighted-Merge-First

Proportional to EDSM score over merges if available, otherwise uniformly over extends.

Backpropagation

The reward is propagated on nodes from the expanded node towards the root. The reward metrics analysed:

DFA Size

The size of the simulated DFA.

Rollout Steps

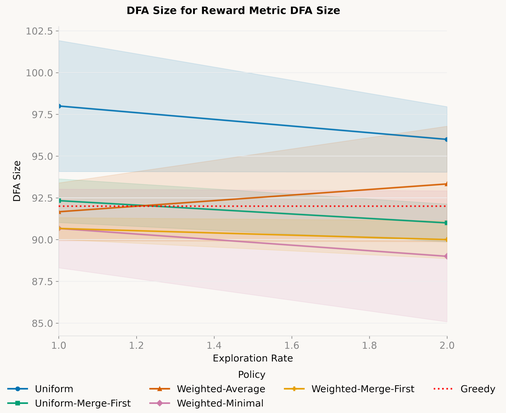
The number of steps taken in the rollout sequence

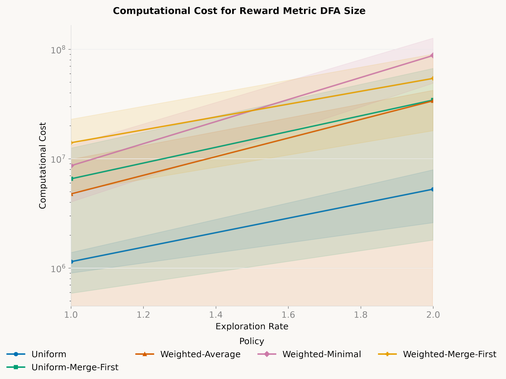
Method

The performance is analysed on 4 datasets with metrics:

- **DFA size**: number of states in the final DFA produced by MCTS
- **Computational cost**: total number of rollout steps across all simulations

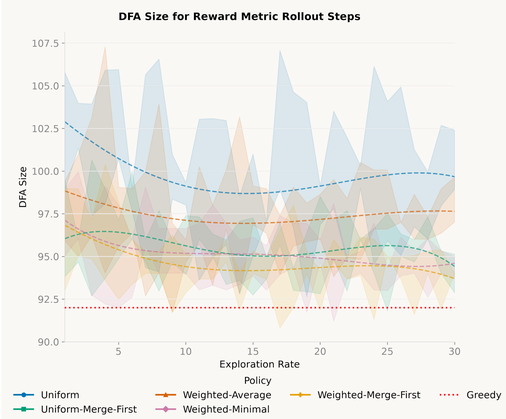
DFA Size Reward Results

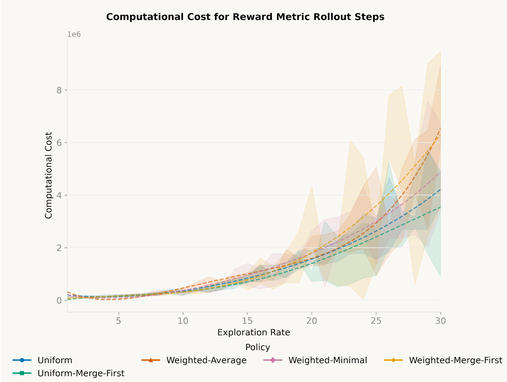




Merge-favouring policies (uniform-merge-first, weighted-minimal, weighted-merge-first) produce smaller DFAs than the greedy baseline, while the computational cost grows exponentially with the exploration rate.

Rollout Steps Reward Results





Merge-favouring policies produce smaller DFAs than the weighted-average and uniform policies, with a lower computational cost than the DFA size reward metric.

Conclusion

- Uniform-merge-first, weighted-minimal and weighted-merge-first produce similarly sized DFAs.
- The weighted-minimal produces more compact DFAs than the weighted-average policy.
- The uniform-merge-first policy produces more compact DFAs than the uniform policy.
- The DFA size reward produces more compact DFAs than rollout steps, at the cost of higher computational expense.

https://github.com/Dev-Bjorn/FlexFringe/tree/mcts_bsc_thesis