IMPROVEMENTS IN MONTE CARLO TREE SEARCH IN PROGRAM SYNTHESIS

AUTHOR: Nathalie van de Werken

SUPERVISOR: Sebastijan Dumančić

1 RESEARCH QUESTION

How can we improve the different components of Monte Carlo Tree Search to make it perform better in program synthesis?

2 BACKGROUND

- Program synthesis:
  - Define input and output together with the language
  - Outputs a program, a sequence that solves the examples
- Monte Carlo Tree Search:

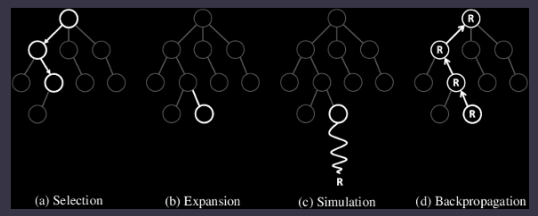


Fig. 1: Monte Carlo Tree Search

- MUTE:
  - Uses UTC for selection and simulation is testing the current program's loss
  - Has shown great potential, but still gets stuck in local optima

3 COMPONENTS CHANGED

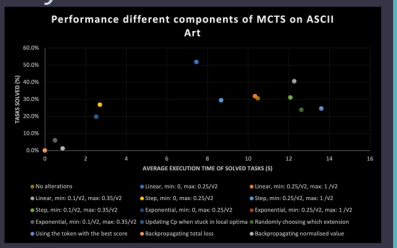
- Balancing exploration and exploitation
  - Changing C_p linearly, step and exponential
  - Increasing C_p once stuck in local optima
- Expansion & Simulation
  - Randomly choosing a token to add
  - Using token's past performance, selecting best token for expansion
- Backpropagation
  - Backpropagating complete loss
  - Backpropagating either +1, 0 or -1

4 EXPERIMENTAL SETUP

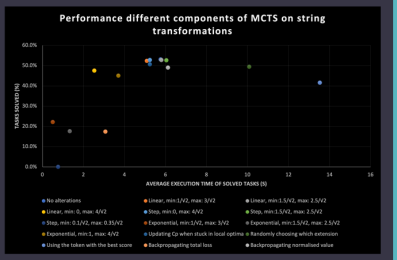
- Compared on two metrics: Accuracy and time
- Ran on three domains:
  - ASCII art
    - Binary distance
  - Robot planning
    - Manhattan distance
  - String transformations
    - Levenshtein's distance

5 RESULTS

- Robot planning : 100% Accuracy
- ASCII Art
  - Greatly improved by changing C_p linearly



- String transformations
  - No big improvements
  - Better tuning parameters could improve results



6 CONCLUSION

- Main improvements by changing C_p linearly, step also showed potential
- Expansion & simulation alternatives did not produce significant results, but still potential
- Changes in backpropagation lowered results
- MCTS can still be improved with many other small and big changes