Monte Carlo Tableaux Prover

Michael Färber, Cezary Kaliszyk, Josef Urban

11.8.2017

Introduction

Introduction

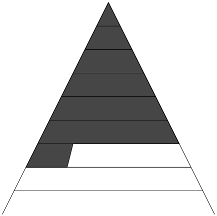
Monte Carlo Tree Search

- Combines tree search with random sampling
- Applied to many games, frequently to Go

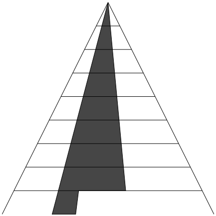
Question

If we see first-order theorem proving as a game, can we use MCTS to guide a first-order automated theorem prover?

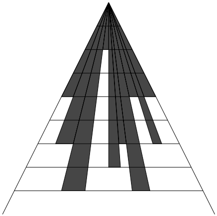
Idea



(a) Iterative deepening without restricted backtracking.



(b) Iterative deepening with restricted backtracking.



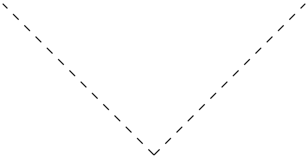
(c) Monte Carlo.

Monte Carlo Tree Search

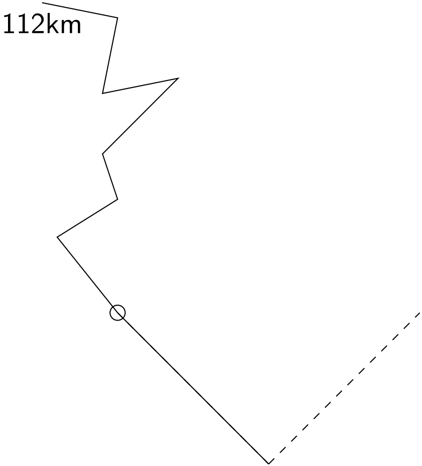
Case Study: Bicycle Routing

Figure 1: Junction near the Czech border: Which way to go?

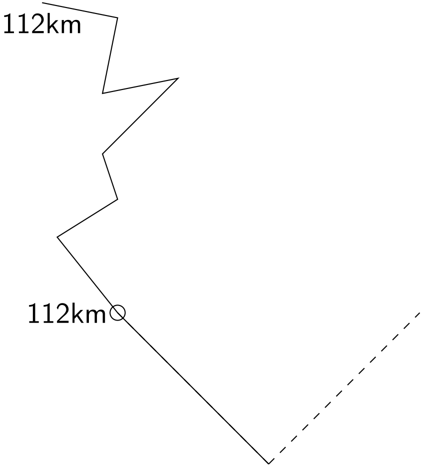
MCTS example



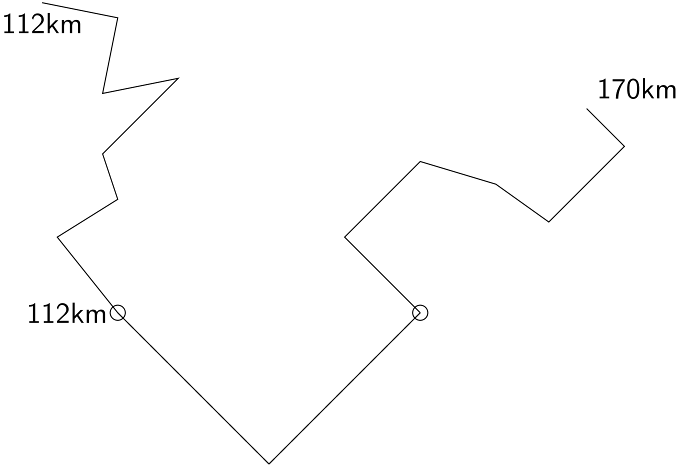
MCTS example



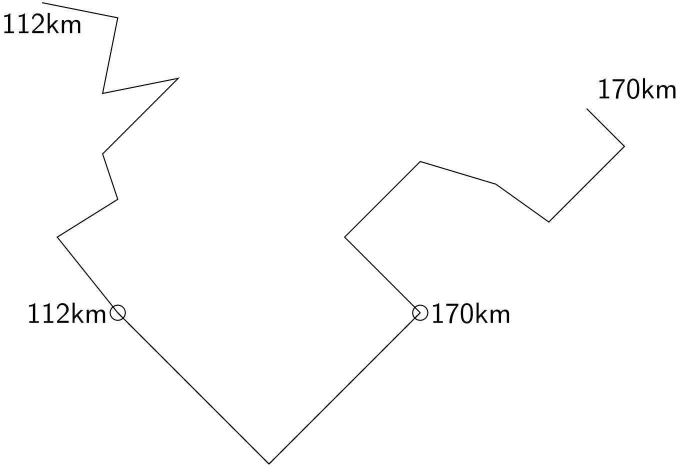
MCTS example



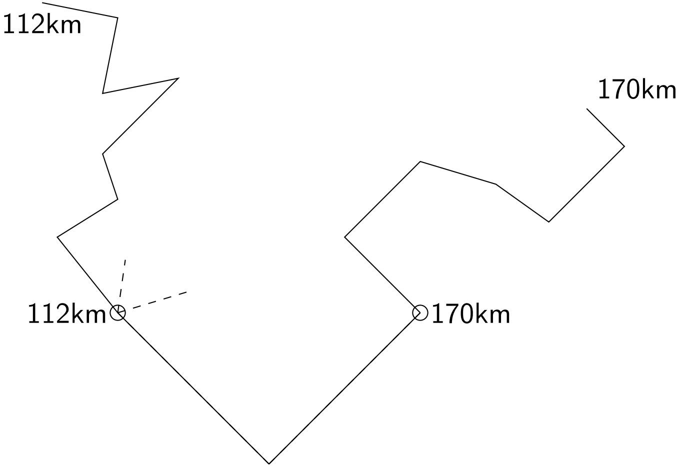
MCTS example



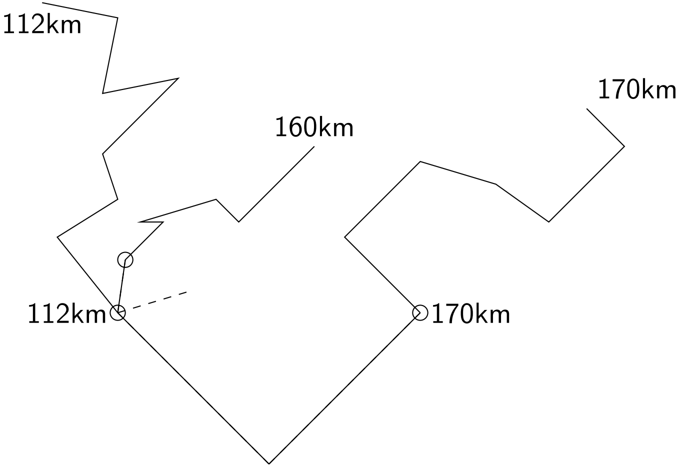
MCTS example



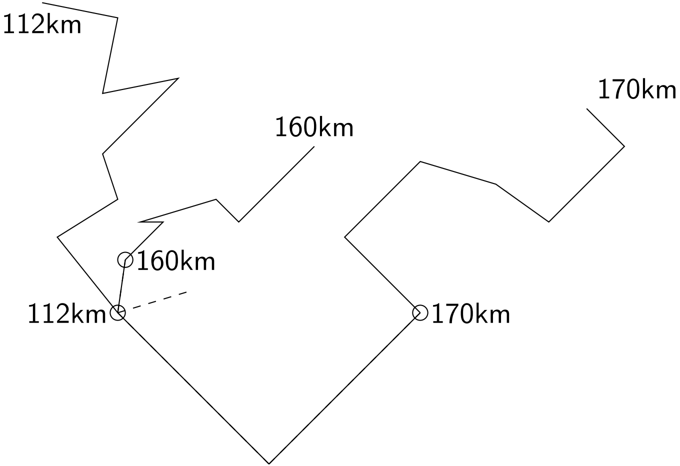
MCTS example



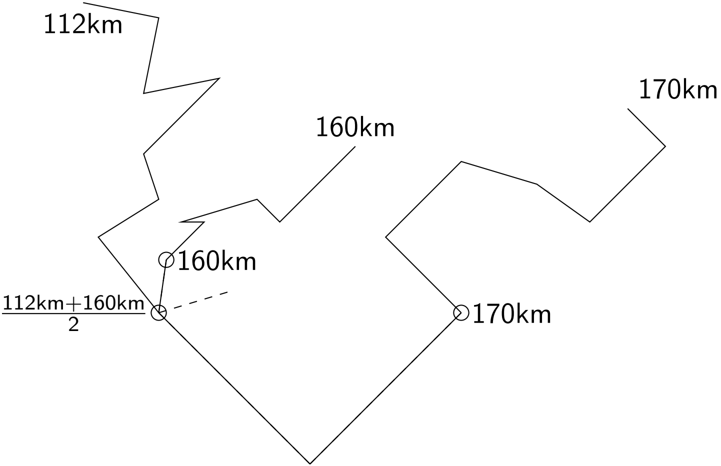
MCTS example



MCTS example



MCTS example



Monte Carlo Tree Search (MCTS)

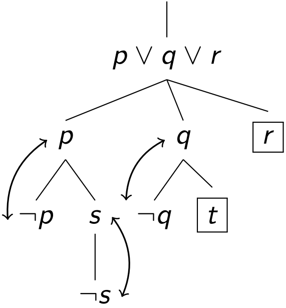
1. Pick state s based on:
   - previous reward (exploitation)
   - number of traversals (exploration)
   - exploration constant: the higher, the more exploration
2. Play random game from s to state s'.
3. Calculate reward of s'.
4. Update rewards of all ancestors of s'.

- How to represent states?
- Which states to start random games from?
- How to play random games?
- How to calculate reward of a state?

State Representation

- State: tableau tree
- Successor state: tableau tree that closes a goal

$$(p \lor q \lor r) \land (\neg p \lor s) \land (\neg p \lor t \lor u) \land \neg s \land (\neg q \lor t) \land (\neg q \lor s)$$



Heuristics

Random Playout Start States

Which states qualify to be start states of random playouts?

Default Policy

Random playout can only be started from a node if for all successor states of ancestors, at least one playout was performed.

Restricted Backtracking Policies

If a random playout started from a node s reaches a state s' that

1. closes one of the goals of s
2. closes all goals of s originating from the same clause

then one may start playouts from s'.

Transition Heuristics

Given a state s, with what probability to choose a successor state s'?

1. Equal probability
2. Inverse number of opened subgoals (clause size)
3. Bayesian probability

Bayesian Probability

Rate successor states by their usefulness in similar situations à la (FE)MaLeCoP

Order vs. Value

- (FE)MaLeCoP: only probability-induced order is used
- MCTS: use probability as visit frequency
  - problem: dimension (extremely small values)
  - solution: normalisation of probabilities

Reward Heuristics

What is the reward of a final state? (i.e. which proof attempts are promising?)

1. Random
2. Ratio of closed and opened goals
3. Size of goal formulae
4. Machine-learnt refutability estimate

Machine-learnt Refutability Estimate

How likely can we solve goals $G = \{g_1, \dots, g_n\}$?

Single goal refutability

- $p(g)$: how often goal g (and all its recursive subgoals) was closed
- $n(g)$: how often closing g failed

The more data $(p + n)$ we have about a goal, the higher its influence.

Multiple goals refutability

$$1 - \frac{1}{|G|} \sum_{g \in G} \frac{n(g)}{p(g) + n(g)} \cdot \sigma(p(g) + n(g))$$

Discrimination

How to measure success of reward function?

Discrimination

Ratio of:

- average reward on branch where proof was found and
- average reward on all explored states

Implementation

Implementation

monteCoP

leanCoP + MCTS = monteCoP

ATP advisor

Play n random games from current ATP state, then process successor states in order of reward

- Only conventional ATP: $n = 0$
- Only MCTS: $n = \infty$

Evaluation

Dataset

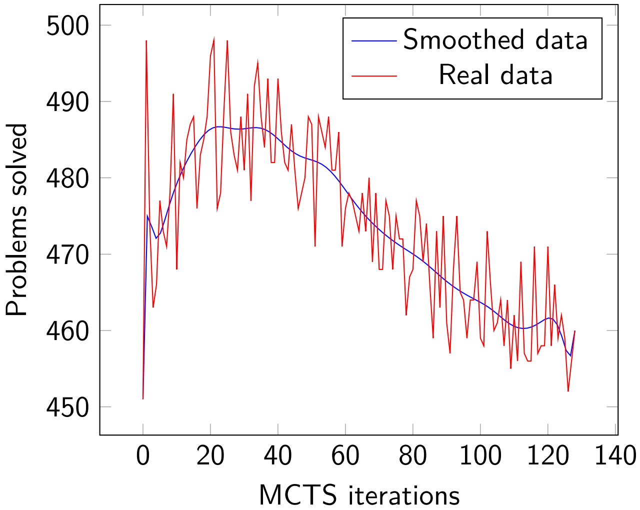
MPTP2078

- 2078 problems from Mizar Mathematical Library
- Consistent symbols/premises across problems

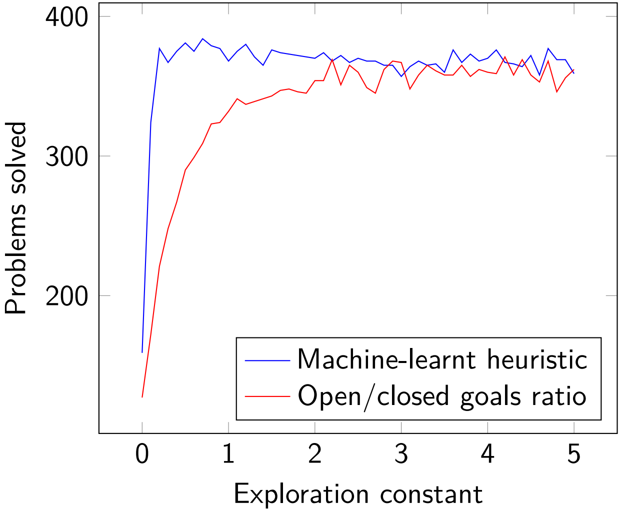
Learning setup

1. Run leanCoP on all problems, collecting training data
2. Use training data in subsequent monteCoP runs

MCTS iterations per inference



Exploration constant



Best configuration

Prover	Timeout [s]	Solved problems
leanCoP	10s	509
monteCoP	10s	538
leanCoP + monteCoP	10s+10s	598
leanCoP	20s	531

Conclusion

Summary

- MCTS used for tableaux proof search
- Reduce search space by starting simulations from deeper nodes
- Bias random simulations by number of opened subgoals
- Estimate quality of states with machine learning techniques
- Usage as advisor gives best results

Future Work

Stronger ML methods for quality estimate: neural networks