Parallelizing POMCP to Solve Complex POMDPs

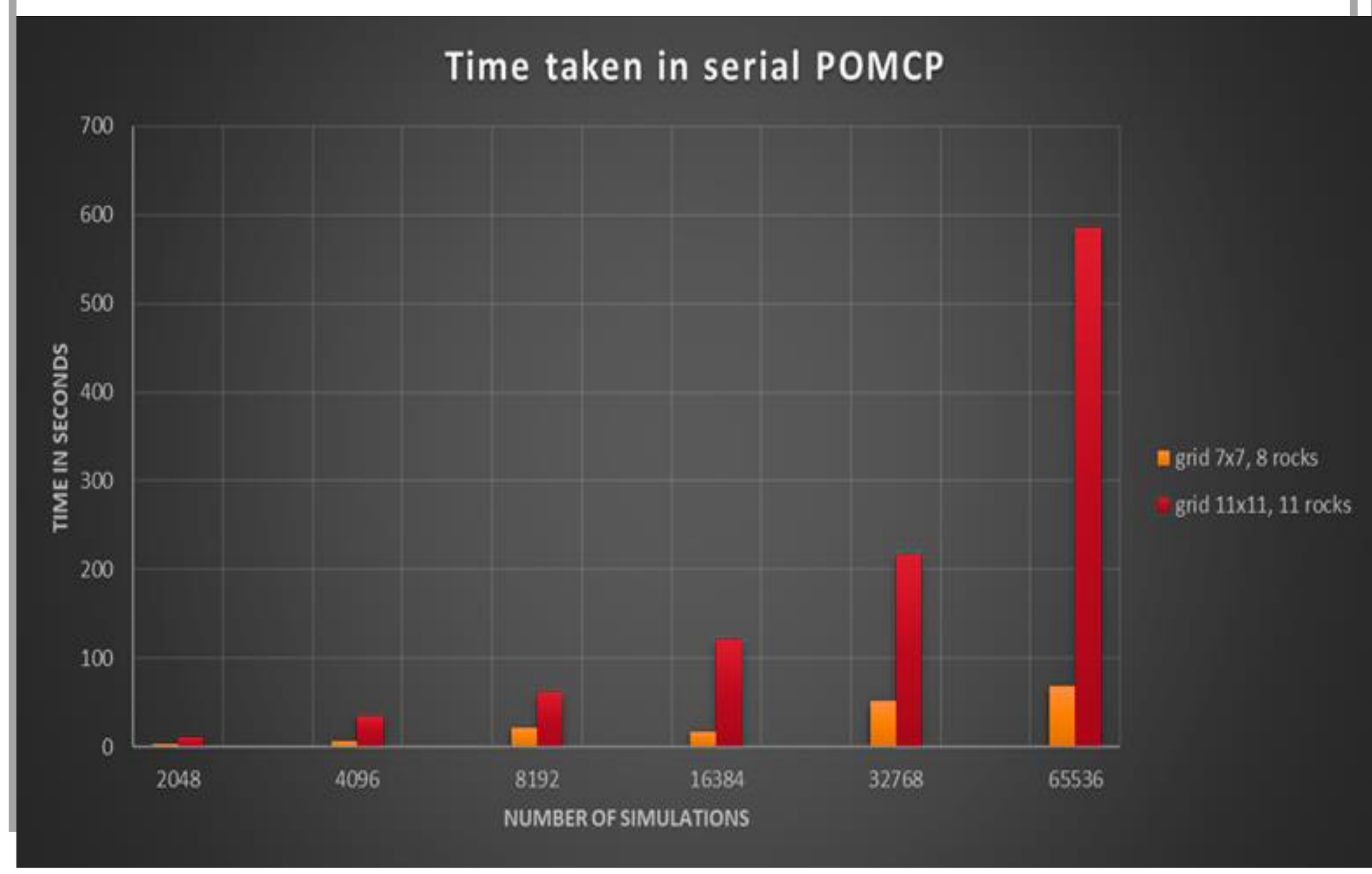
Semanti Basu Sreshtaa Rajesh Kaiyu Zheng Stefanie Tellex R. Iris Bahar

Dept. of Computer Science, Brown University, Providence, RI 02912

BROWN

Problem overview

- Robots must plan under uncertainties.
- Partially Observable Markov Decision Process (**POMDP**) : architecture used to **model planning under uncertainty**.
- Partially Observable Monte-Carlo Planning (**POMCP**) : A planning algorithm that **solves POMDP problems**.
- POMCP is based on running **large number of simulations**.
- On **large, complex domains**, running enough simulations takes too long, rendering **POMCP unusable** in many situations.
- **Goal**: **Parallelize POMCP** to aid in **faster decision making** across large, complex problems.



Challenges

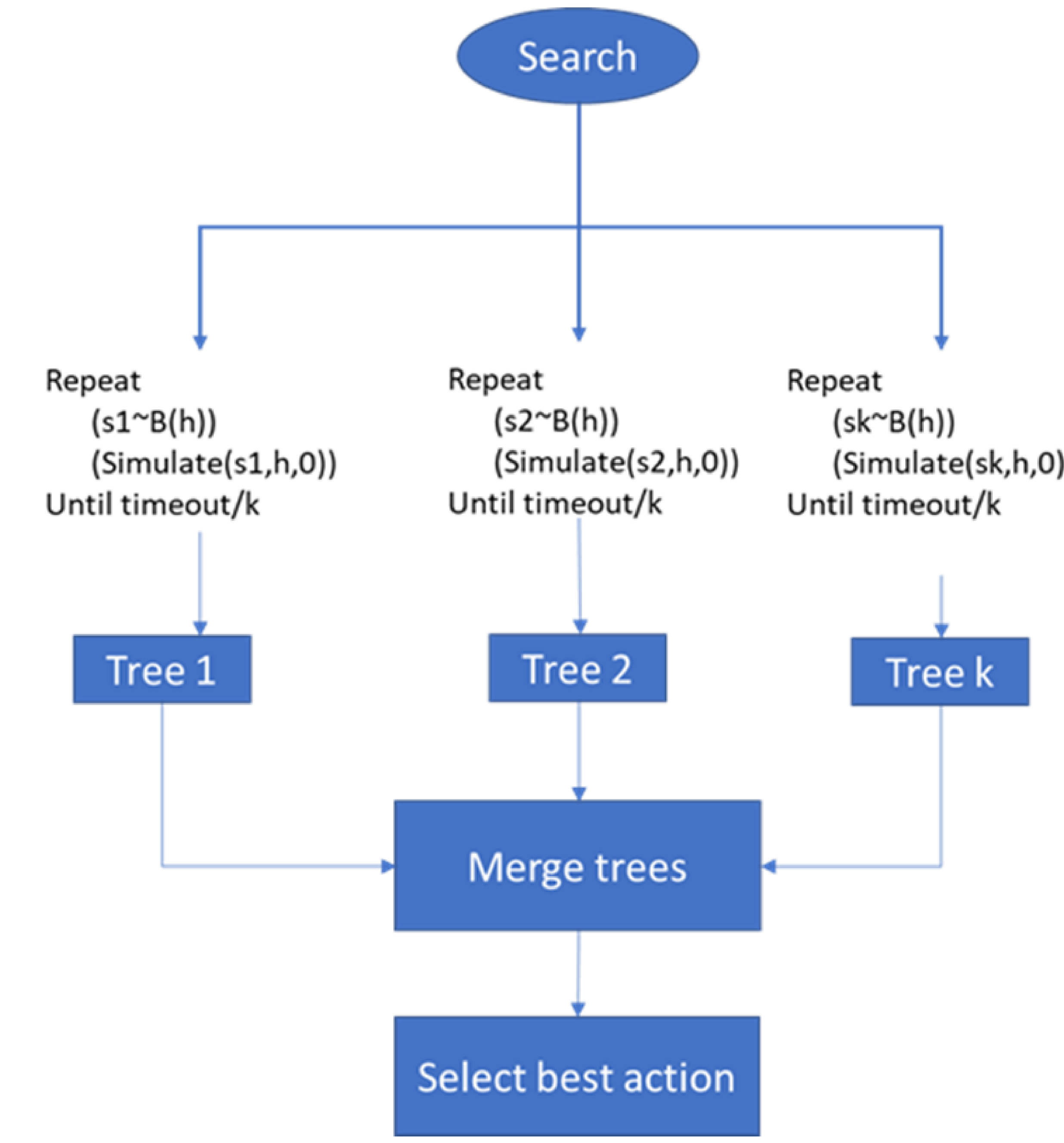
POMCP IS **INHERENTLY SEQUENTIAL**

- Sampling a state from current belief.
- Building search tree
- Selecting best action and executing it.
- Receiving an observation and pruning tree

- How to **parallelize sequential decision** making ?
- How do we **speed it up without compromising on performance/accuracy**?

Our Solution

- Isolating the **most computationally expensive** portion of POMCP
  - POMCP builds a **look-ahead tree of histories** (action-observation pairs up to that point) by running several simulations.
  - A modified Monte Carlo Tree Search (MCTS) is used to then select the best action.
- Speed up building the search tree for action selection
  - Extend techniques for **parallelizing MCTS** to POMCP.
  - **Root and Tree parallelization** are two common schemes for parallelizing MCTS.
  - We extend these to POMCP.
- A root parallel version of POMCP was built by modifying the original C++ code.



Root Parallel POMCP

- In serial POMCP, the search tree is built by running a certain number of simulations.
- **Equivalent accuracy in lesser time** : extending the concept of root parallelization in MCTS to POMCP.
- This involves **building multiple search trees** simultaneously.
- The results are **merged** to obtain the final action.
- After action execution, an observation and reward is received which are used to prune all the search trees.
- Action selection is repeatedly performed using the steps above, until termination.

Results

The rock sample problem consists of a grid with rocks (can be good/bad). The agent must move to the exit area and sample as many good rocks as it can along the way. The agent gets a positive reward on sampling a good crock, negative reward on sampling a bad one. The agent must sense if the rocks are good/bad.

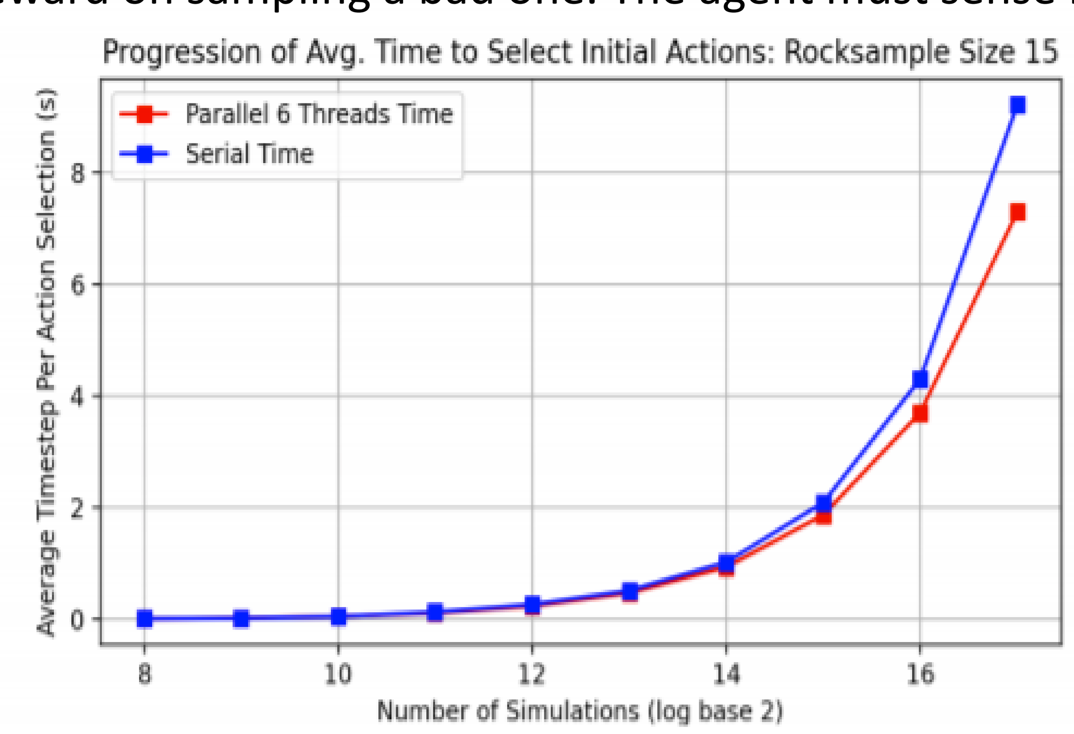


Fig. 1: Average time to select initial actions: Rocksample 15,15

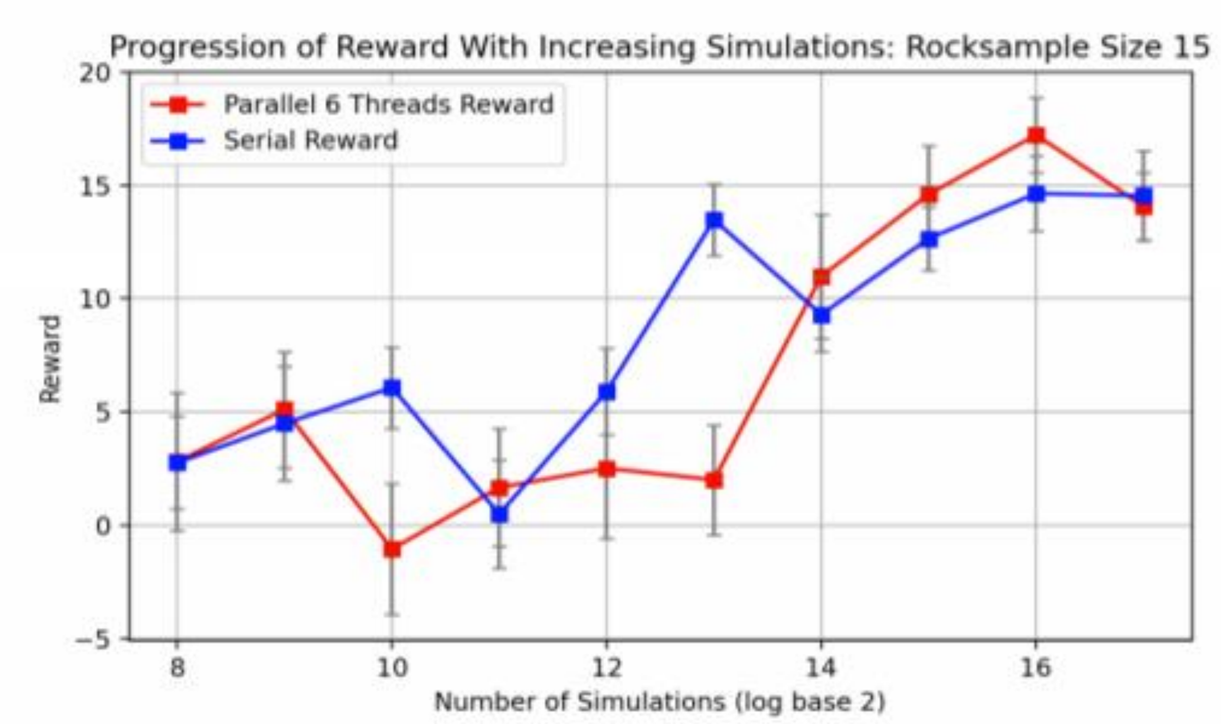


Fig. 2: Cumulative reward comparison: Rocksample 15,15

Conclusion and Future work

- ✓ Encouraging results from the root parallel POMCP algorithm on the Rocksample domain.
  - **Parallel POMCP runs faster** than the serial version but **performs no worse** in terms of cumulative reward achieved.
- ✓ Future work:
  - Implement Tree Parallel POMCP
  - Evaluate parallel POMCP on a hard POMDP problem on an **actual robot, such as a grasping/robotic arm manipulation problem**.