Monte Carlo Tree Search on a Graphics Processing Unit in the Context of Online Robotic Decision Making

Felix Steinberger Eriksson

Mentors: Soon-Jo Chung and Ben Riviere

24 August 2023

Caltech

Introduction

- Sequential decision problems occur in the context of e.g. autonomous vehicles

Introduction

- Sequential decision problems occur in the context of e.g. autonomous vehicles
- In real scenarios: sequences of such problems, require fast solutions

Introduction

- Sequential decision problems occur in the context of e.g. autonomous vehicles
- In real scenarios: sequences of such problems, require fast solutions
- Critically, we want anytime algorithms that can provide reasonable solutions if terminated early.

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Definition

A (deterministic finite) *Markov Decision Process (MDP)* is a tuple $\langle \mathcal{X}, \mathcal{U}, F, R, H \rangle$ where

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Definition

A (deterministic finite) *Markov Decision Process (MDP)* is a tuple $\langle \mathcal{X}, \mathcal{U}, F, R, H \rangle$ where

- $\mathcal{X}$ is the *state space*,

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Definition

A (deterministic finite) *Markov Decision Process (MDP)* is a tuple $\langle \mathcal{X}, \mathcal{U}, F, R, H \rangle$ where

- $\mathcal{X}$ is the *state space*,
- $\mathcal{U}$ is the *action set*,

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Definition

A (deterministic finite) *Markov Decision Process (MDP)* is a tuple $\langle \mathcal{X}, \mathcal{U}, F, R, H \rangle$ where

- $\mathcal{X}$ is the *state space*,
- $\mathcal{U}$ is the *action set*,
- $F : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the *transition function*,

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Definition

A (deterministic finite) *Markov Decision Process (MDP)* is a tuple $\langle \mathcal{X}, \mathcal{U}, F, R, H \rangle$ where

- $\mathcal{X}$ is the *state space*,
- $\mathcal{U}$ is the *action set*,
- $F : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the *transition function*,
- $R : \mathcal{X} \times \mathcal{U} \to [0, 1] \subset \mathbb{R}$ is the *reward function* and

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Definition

A (deterministic finite) *Markov Decision Process (MDP)* is a tuple $\langle \mathcal{X}, \mathcal{U}, F, R, H \rangle$ where

- $\mathcal{X}$ is the *state space*,
- $\mathcal{U}$ is the *action set*,
- $F : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the *transition function*,
- $R : \mathcal{X} \times \mathcal{U} \to [0, 1] \subset \mathbb{R}$ is the *reward function* and
- $H \in \mathbb{N}$ is the *horizon*.

Markov Decision Processes (MDPs)

- Often model a sequential decision problem as an MDP.

Definition

A (deterministic finite) *Markov Decision Process (MDP)* is a tuple $\langle \mathcal{X}, \mathcal{U}, F, R, H \rangle$ where

- $\mathcal{X}$ is the *state space*,
- $\mathcal{U}$ is the *action set*,
- $F : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$ is the *transition function*,
- $R : \mathcal{X} \times \mathcal{U} \to [0, 1] \subset \mathbb{R}$ is the *reward function* and
- $H \in \mathbb{N}$ is the *horizon*.

- For simplicity, consider only finite $\mathcal{X}$ ($|\mathcal{X}| = X < \infty$) and $\mathcal{U}$ ($|\mathcal{U}| = U < \infty$).

Example (finite deterministic MDP)

At time t, in state $x_t \in \mathcal{X}$, an agent

Example (finite deterministic MDP)

At time t, in state $x_t \in \mathcal{X}$, an agent

- selects an action $a_t \in \mathcal{U}$,

Example (finite deterministic MDP)

At time t, in state $x_t \in \mathcal{X}$, an agent

- selects an action $a_t \in \mathcal{U}$,
- receives reward $R(x_t, a_t)$, and

Example (finite deterministic MDP)

At time t, in state $x_t \in \mathcal{X}$, an agent

- selects an action $a_t \in \mathcal{U}$,
- receives reward $R(x_t, a_t)$, and
- transitions to state $x_{t+1} = F(x_t, a_t)$.

Solving an MDP

- An agent's interaction is described by a *policy* π that determines the selected action given the current state.

Solving an MDP

- An agent's interaction is described by a *policy* π that determines the selected action given the current state.
- In the finite horizon case, a policy is a sequence $(a_h)_h \in \mathcal{U}^H$.

Solving an MDP

- An agent's interaction is described by a *policy* π that determines the selected action given the current state.
- In the finite horizon case, a policy is a sequence $(a_h)_h \in \mathcal{U}^H$.
- Objective: find optimal policy/action sequence $(a_h^*)_h \in \mathcal{U}^H$.

Solving an MDP

- An agent's interaction is described by a *policy* π that determines the selected action given the current state.
- In the finite horizon case, a policy is a sequence $(a_h)_h \in \mathcal{U}^H$.
- Objective: find optimal policy/action sequence $(a_h^*)_h \in \mathcal{U}^H$.
- Equivalently: determine $(a_h^*)_h \in \arg\max_{(a_h)_h} \sum_{h=0}^{H-1} R(x_h, a_h)$ where $x_{h+1} = F(x_h, a_h)\ \forall h = 0, \ldots, H-1$.

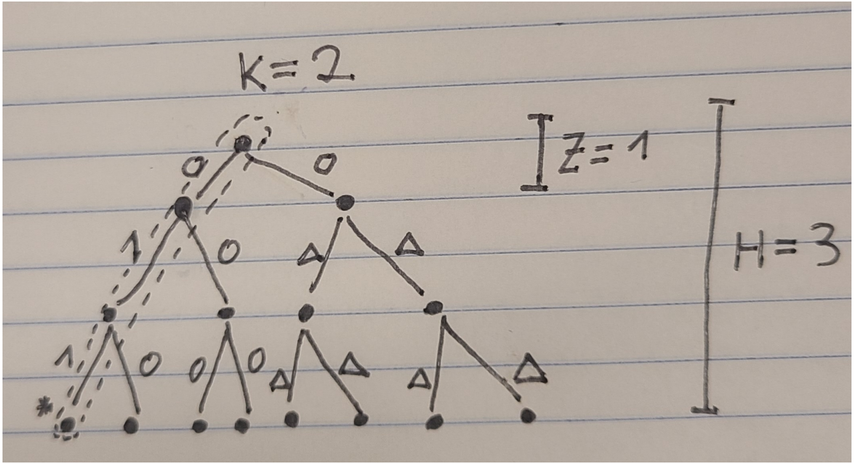
Associated search tree

- A finite MDP can be associated with a *graph* (in general), here a *tree*.

Associated search tree

- A finite MDP can be associated with a *graph* (in general), here a *tree*.
- Canonical MDP: Delayed Dense Needle with Gap

Delayed Dense Needle with Gap



Monte Carlo Tree Search (MCTS)

- General/flexible framework for randomized tree search. Iterate the following:

Monte Carlo Tree Search (MCTS)

- General/flexible framework for randomized tree search. Iterate the following:

1) Explore exhaustively to some leaf node in current tree.

Monte Carlo Tree Search (MCTS)

- General/flexible framework for randomized tree search. Iterate the following:

1) Explore exhaustively to some leaf node in current tree.
2) Add a child node.

Monte Carlo Tree Search (MCTS)

- General/flexible framework for randomized tree search. Iterate the following:

1) Explore exhaustively to some leaf node in current tree.
2) Add a child node.
3) Simulate trajectory until termination.

Monte Carlo Tree Search (MCTS)

- General/flexible framework for randomized tree search. Iterate the following:

1) Explore exhaustively to some leaf node in current tree.
2) Add a child node.
3) Simulate trajectory until termination.
4) Backpropagate collected statistics about states on trajectory.

GORP

- As a proxy for MCTS we consider Greedy over Random Policy (GORP) (Laidlaw et al., 2023).

GORP

- As a proxy for MCTS we consider Greedy over Random Policy (GORP) (Laidlaw et al., 2023).
- Similar to Model Predictive Control. To compute the next optimal action, explore exhaustively for the next κ steps, then sample m rollouts until termination.

GORP

- As a proxy for MCTS we consider Greedy over Random Policy (GORP) (Laidlaw et al., 2023).
- Similar to Model Predictive Control. To compute the next optimal action, explore exhaustively for the next κ steps, then sample m rollouts until termination.
- With exploration horizon κ and per-node sampling budget m selected appropriately, can guarantee an optimal policy with sample complexity $\mathcal{O}(H^2|U|^\kappa m)$, where H is the horizon of the MDP and $|U|$ is the number of actions in the fixed, finite action set U.

Problems with GORP

- Not adaptive: κ and m need to be large enough to find the correct action at each iteration. In many cases, this leads to over-computation for easier subproblems. Future research direction.

Problems with GORP

- Not adaptive: κ and m need to be large enough to find the correct action at each iteration. In many cases, this leads to over-computation for easier subproblems. Future research direction.
- Not anytime: Early termination gives no guarantees on degree of suboptimality of yielded solution. May not even yield a complete policy.

Problems with GORP

- Not adaptive: κ and m need to be large enough to find the correct action at each iteration. In many cases, this leads to over-computation for easier subproblems. Future research direction.
- Not anytime: Early termination gives no guarantees on degree of suboptimality of yielded solution. May not even yield a complete policy.
- Need a priori knowledge of parameters κ and m.

Problems with GORP

- Not adaptive: κ and m need to be large enough to find the correct action at each iteration. In many cases, this leads to over-computation for easier subproblems. Future research direction.
- Not anytime: Early termination gives no guarantees on degree of suboptimality of yielded solution. May not even yield a complete policy.
- Need a priori knowledge of parameters κ and m.
- Often, computing these is at least as expensive as solving the MDP in question.

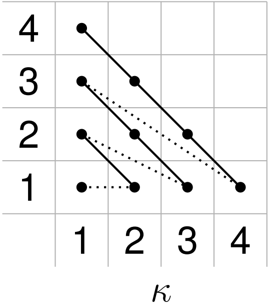
Doubling trick

- One approach to alleviate the last two problems is to employ the following doubling trick.

Doubling trick

- One approach to alleviate the last two problems is to employ the following doubling trick.

$$p = \log_{|U|} m$$



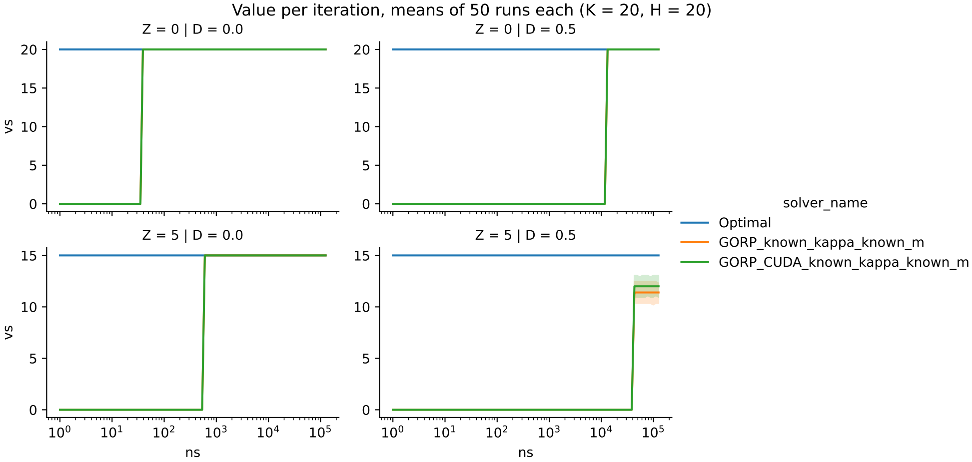
Implementation

- Employ GPGPU programming in CUDA to achieve orders of magnitude better real-time performance than possible on a CPU.

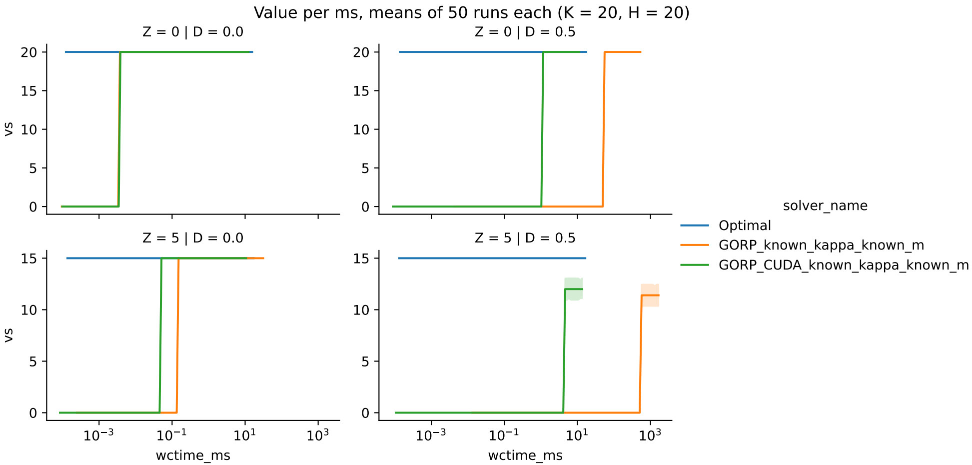
Results

- Speedups of factor 70 to 140 on canonical evaluation MDP.

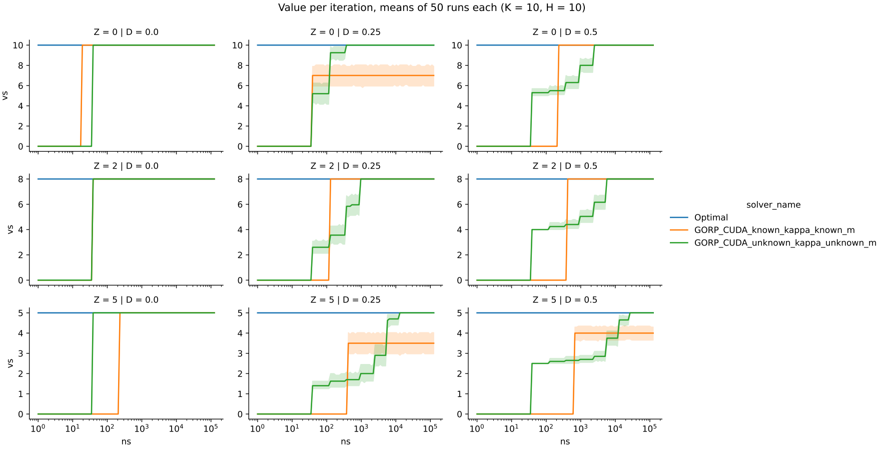
Results


Value per iteration, means of 50 runs each (K = 20, H = 20)
Z = 0 | D = 0.0
Z = 0 | D = 0.5
Z = 5 | D = 0.0
Z = 5 | D = 0.5
vs
ns
solver_name
Optimal
GORP_known_kappa_known_m
GORP_CUDA_known_kappa_known_m

Results


Value per ms, means of 50 runs each (K = 20, H = 20)
Z = 0 | D = 0.0
Z = 0 | D = 0.5
Z = 5 | D = 0.0
Z = 5 | D = 0.5
vs
wctime_ms
solver_name
Optimal
GORP_known_kappa_known_m
GORP_CUDA_known_kappa_known_m

Results



Value per iteration, means of 50 runs each (K = 10, H = 10)
Z = 0 | D = 0.0
Z = 0 | D = 0.25
Z = 0 | D = 0.5
Z = 2 | D = 0.0
Z = 2 | D = 0.25
Z = 2 | D = 0.5
Z = 5 | D = 0.0
Z = 5 | D = 0.25
Z = 5 | D = 0.5
vs
ns
solver_name
Optimal
GORP_CUDA_known_kappa_known_m
GORP_CUDA_unknown_kappa_unknown_m

Summary & Conclusion

- Orders of magnitudes faster speedup

Summary & Conclusion

- Orders of magnitudes faster speedup
- Anytime version of GORP

Summary & Conclusion

- Orders of magnitudes faster speedup
- Anytime version of GORP

$\Longrightarrow$ A strong case for feasibility of MCTS in real-time applications like robotic planning.

Acknowledgements

Thank you!
Reach out at felixse@kth.se.

Thank you to Soon-Jo Chung, Ben Riviere and all the amazing people at ARCL!