Topics in Reinforcement Learning
Review of Multiagent Rollout

Dimitri P. Bertsekas\textsuperscript{1} Yuchao Li\textsuperscript{2}
Video Credit: Alejandro Penacho Riveiros\textsuperscript{3} William Emanuelsson\textsuperscript{4}

\textsuperscript{1}dbertsek@asu.edu
\textsuperscript{2}yuchao@kth.se
\textsuperscript{3}alejpr@kth.se \textsuperscript{4}wem@kth.se

March 1, 2023
Multiagent Problems (1960s →)

- Multiple agents collecting and sharing information selectively with each other and with an environment/computing cloud
- Agent $i$ applies decision $u^i$ sequentially in discrete time based on info received

The major mathematical distinction between problem structures
- The classical information pattern: Agents are fully cooperative, fully sharing and never forgetting information. Can be treated by DP
- The nonclassical information pattern: Agents are partially sharing information and may be antagonistic. HARD because it is hard to treat by DP
Starting Point: A Classical Information Pattern

At each time: Agents have exact state info; choose their controls as function of state

Model: A discrete-time (possibly stochastic) system with state $x$ and control $u$

- Decision/control has $m$ components $u = (u^1, \ldots, u^m)$ corresponding to $m$ “agents”
- “Agents” is just a metaphor - the important math structure is $u = (u^1, \ldots, u^m)$
- The theoretical framework is DP. We will reformulate for faster computation
  - We first aim to deal with the exponential size of the search/control space
  - Later we will discuss how to compute the agent controls in distributed fashion (in the process we will deal in part with nonclassical info pattern issues)
3 agents move in 4 directions with perfect vision. They have been assigned to some targets. Objective is to reach their respective targets in minimum time while avoiding collision with each other.

- At each time we must select one out of \( \approx 5^m \) joint move choices
- We will reduce to \( 5 \cdot m \) (while maintaining good properties)
- Idea: Break down the control into a sequence of one-agent-at-a-time moves
- Scales well, up to \( m = 200 \) agents, with average computational time around 50 ms. Can also adapt to a changing environment through recomputation. See paper https://arxiv.org/abs/2211.08201 as well as implementation in C++ https://github.com/will-em/multi-agent-rollout
Reformulation Ideas: Trading off Control and State Complexity

An equivalent reformulation - “Unfolding” the control action

- The control space is simplified at the expense of $m - 1$ additional layers of states, and corresponding $m - 1$ cost functions
  \[ J^1(x, u^1), J^2(x, u^1, u^2), \ldots, J^m(x, u^1, \ldots, u^m), \]

- Allows far more efficient rollout (one-agent-at-a-time). This is just standard rollout for the reformulated problem (so it involves a Newton step)

- The increase in size of the state space does not adversely affect rollout (only one state and its successors are looked at each stage during on-line play)

- Complexity reduction: The one-step lookahead branching factor is reduced from $n^m$ to $n \cdot m$, where $n$ is the number of possible choices for each component $u^i$
Reshuffling the order of agents results in a different, yet still equivalent problem.

Multiagent rollout admits parallel implementation.

Multiple base heuristics can be applied to enhance the performance further.

All those ideas are independent of each other, and can be combined.