Lecture 6
Model Predictive Control, Multiagent Rollout
1. Model Predictive Control (MPC) and Variations
2. Multiagent Problems in General
3. Multiagent Rollout/Policy Improvement
4. Autonomous Multiagent Rollout
5. Multirobot Repair - A Large-Scale Multiagent POMDP Problem
REGULATION PROBLEM
Keep the state near some given point
Traditionally 0 (the origin)
\( \theta = 0, \dot{\theta} = 0 \)

FOLLOW A GIVEN TRAJECTORY

Fixed Obstacles
Moving Obstacle
Acceleration Constraints
Velocity Constraints

Must Deal with State and Control Constraints
Linear-Quadratic Formulation is Often Inadequate
The Original Form of MPC for Regulation to the Origin Problems

- System: $x_{k+1} = f(x_k, u_k)$; 0 is an absorbing (goal) state, $f(0, u) \equiv 0$.
- Cost per stage: $g(x_k, u_k) > 0$, except that 0 is cost-free, $g(0, u) \equiv 0$.
- Control constraints: $u_k \in U(x_k)$ for all $k$. Perfect state information.
- MPC: At $x_k$ solve an $\ell$-step lookahead version of the problem, requiring $x_{k+\ell} = 0$ ($\ell$: fixed and sufficiently large to allow the transfer to 0).
- If $\{\tilde{u}_k, \ldots, \tilde{u}_{k+\ell-1}\}$ is the control sequence so obtained, apply $\tilde{u}_k$, discard $\tilde{u}_{k+1}, \ldots$
MPC is rollout w/ base heuristic the \((\ell - 1)\)-step min to 0 (and stay at 0). Let \(H(x)\) denote the optimal cost of the \((\ell - 1)\)-step min, starting from \(x\).

This heuristic is **sequentially improving** (not sequentially consistent), i.e.,

\[
\min_{u \in U(x)} [g(x, u) + H(f(x, u))] \leq H(x)
\]

because \((\text{opt. cost to reach 0 in } \ell \text{ steps}) \leq (\text{opt. cost to reach 0 in } \ell - 1 \text{ steps})\)

**Sequential improvement** \(\rightarrow\) “stability”, i.e., that the MPC controller has a finite cost from every initial state \(x_0\).

Reason: By the cost improvement property, the cost of the MPC controller starting from \(x_0\) is no greater than \(H(x_0) < \infty\).
At state $x_0$, instead of requiring that $x_\ell = 0$, we solve

$$\min_{u_i, i=0,\ldots,\ell-1} \left[ G(x_\ell) + \sum_{i=0}^{\ell-1} g(x_i, u_i) \right],$$

subject to $u_i \in U(x_i)$ and $x_{i+1} = f(x_i, u_i)$, where $G(x) > 0$ for $x \neq 0$, and $G(0) = 0$.

This is $\ell$-step lookahead minimization with terminal cost function $G$.

Let us assume that $TG \leq G$, where $T$ is the min-Bellman operator, i.e., for all $x$,

$$(TG)(x) = \min_{u \in U(x)} \left[ g(x, u) + G(f(x, u)) \right] \leq G(x).$$

We can show that this condition implies stability of the MPC controller. An analytical proof is possible (see the “Lessons ...” book, Section 3.2), but we give a graphical argument in this lecture.

The argument is based on the concept of the region of stability: this is the set of all $\tilde{J}$ such that the policy $\tilde{\mu}$ obtained by one-step lookahead minimization,

$$T_{\tilde{\mu}} \tilde{J} = T\tilde{J},$$

is stable.
Region of Stability - A Terminal Cost Function $G$ Satisfying $TG \leq G$

$TG \leq G$ implies that $T^\ell G$ lies within the region of stability for all $\ell$
Abstract Visualization of MPC with $\ell$-Step Lookahead Minimization and Terminal Cost Function $G$ Satisfying $TG \leq G$

$TG \leq G$ implies that $G$ lies in the region of stability, and so does $T^\ell G$ for any $\ell \geq 1$
MPC with $\ell$-Step Lookahead Minimization, $m$-Step Truncated Rollout, and Terminal Cost Function $G$: The AlphaZero Architecture!

Larger values of $m$ and $\ell$ help make the MPC policy stable.
Other Variants of MPC

Unstable System

\[ x_{k+1} = 2x_k + u_k \]

Control constraint: \[ |u_k| \leq 1 \]

Tube \[ X = \{ -a \leq x \leq a \} \]

Tube Constraint Cannot be Satisfied for all \( x_0 \in X \)
if \( a > 1 \)

**MPC with state/safety/tube constraints: \( x_k \in X \) for all \( k \)**

- Special difficulty: The tube constraint may be impossible to satisfy for some \( x_0 \in X \)
- Need to construct (off-line) an inner tube from within which the state constraints can be met
- Leads to the methods of reachability of target tubes (my 1971 PhD thesis, on-line)

**Combinations with off-line training methods**

Training of terminal cost function approximation, a base policy for truncated rollout, etc

**MPC for stochastic problems:** Must solve an \( \ell \)-step stochastic DP problem on-line. Can be dealt with certainty equivalence, except for the first stage
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 structures

- The classical information pattern: Agents are fully cooperative, fully sharing and never forgetting information. Can be treated by Dynamic Programming (DP)
- The nonclassical information pattern: Agents are partially sharing information, and may be antagonistic. HARD because it cannot be treated by DP
Our Starting Point: A Classical Information Pattern ... but we will Generalize

The agents have exact state info, and choose their controls as functions of the state

Model: Stochastic DP (finite or infinite horizon) 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 $u = (u_1, \ldots, u_m)$
- We apply approximate DP/rollout ideas, aiming at faster computation in order to:
  - Deal with the exponential size of the search/control space
  - Be able to compute the agent controls in parallel (in the process we will deal in part with nonclassical info pattern issues)
To simplify notation, consider infinite horizon setting. The standard rollout operation is

\[
(\tilde{\mu}_1(x), \ldots, \tilde{\mu}_m(x)) \in \arg \min_{(u_1, \ldots, u_m)} \mathbb{E}_w \left\{ g(x, u_1, \ldots, u_m, w) + \alpha J_{\mu} \left( f(x, u_1, \ldots, u_m, w) \right) \right\};
\]

the search space is exponential in \( m \) (\( \mu \) is the base policy, seq. consistency holds)

**Multiagent rollout (a form of simplified rollout; implies cost improvement)**

Perform a sequence of \( m \) successive minimizations, one-agent-at-a-time

\[
\tilde{\mu}_1(x) \in \arg \min_{u_1} \mathbb{E}_w \left\{ g(x, u_1, \mu_2(x), \ldots, \mu_m(x), w) + \alpha J_{\mu} \left( f(x, u_1, \mu_2(x), \ldots, \mu_m(x), w) \right) \right\}
\]

\[
\tilde{\mu}_2(x) \in \arg \min_{u_2} \mathbb{E}_w \left\{ g(x, \tilde{\mu}_1(x), u_2, \mu_3(x), \ldots, \mu_m(x), \mu_2(x), w) + \alpha J_{\mu} \left( f(x, \tilde{\mu}_1(x), u_2, \mu_3(x), \ldots, \mu_m(x), \mu_2(x), w) \right) \right\}
\]

\[
\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\
\tilde{\mu}_m(x) \in \arg \min_{u_m} \mathbb{E}_w \left\{ g(x, \tilde{\mu}_1(x), \tilde{\mu}_2(x), \ldots, \tilde{\mu}_{m-1}(x), u_m, w) + \alpha J_{\mu} \left( f(x, \tilde{\mu}_1(x), \tilde{\mu}_2(x), \ldots, \tilde{\mu}_{m-1}(x), u_m, w) \right) \right\}
\]

- Has a search space with size that is linear in \( m \); ENORMOUS SPEEDUP!

Objective is to catch the flies in minimum time

- At each time we must select one out of \( \approx 5^{15} \) joint move choices
- Multiagent rollout reduces this to \( 5 \cdot 15 = 75 \) choices (while maintaining cost improvement); applies a sequence of one-spider-at-a-time moves
- Later, we will introduce “precomputed signaling/coordination” between the spiders, so the 15 spiders will choose moves in parallel (extra speedup factor of up to 15)
Base policy: Move along the shortest path to the closest surviving fly (in the Manhattan distance metric). **No coordination.**

**Time to catch the flies**
- Base policy (each spider follows the shortest path): Capture time = 85
- Standard rollout (all spiders move at once, $5^4 = 625$ move choices): Capture time = 34
- Agent-by-agent rollout (spiders move one at a time, $4 \cdot 5 = 20$ move choices): Capture time = 34
Think about an equivalent problem reformulation for multiagent rollout

- "Unfold" the control action
- Consider standard (not multiagent) rollout for the reformulated problem
- What about cost improvement?

Let's Take a Working Break
Justification of Cost Improvement through Reformulation: Trading off Control and State Complexity (NDP Book, 1996)

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

$$J^1(x, u_1), J^2(x, u_1, u_2), \ldots, J^{m-1}(x, u_1, \ldots, u_{m-1})$$

- Multiagent rollout is just standard rollout for the reformulated problem
- The increase in size of the state space does not adversely affect rollout (only one state per stage is looked at 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 $u_i$
Consider MPC where $u_k$ consists of both discrete and continuous components

$$u_k = (y_k^1, \ldots, y_k^m, v_k),$$

where $y_k^1, \ldots, y_k^m$ are discrete, and $v_k$ is continuous.

- For example $y_k^1, \ldots, y_k^m$ may be system configuration variables, and $v_k$ may be a multidimensional vector with real components (e.g., as in linear quadratic control).

- The base policy may consist of a “nominal configuration" $\bar{y}_k^1, \ldots, \bar{y}_k^m$ (that depends on the state $x_k$), and a continuous control policy that drives the state to 0 in $(\ell - 1)$ steps with minimum cost.

- In a component-by-component version of MPC, at state $x_k$:
  - $y_k^1, \ldots, y_k^m$ are first chosen one-at-a-time, and with all future components fixed at the values determined by the nominal configuration/base policy.
  - Then the continuous component $v_k$ is chosen to drive the state to 0 in $\ell$ steps at minimum cost with the discrete components fixed.

- This simplifies lookahead minimization by:
  - Separating the “difficult" minimization over $y_k^1, \ldots, y_k^m$ from the continuous minimization over $v_k$
  - Optimizing over $y_k^1, \ldots, y_k^m$ one-at-a-time (simpler integer programming problem).

- Maintains the cost improvement/stability property of MPC.
Multiagent rollout/policy improvement is an inherently serial computation. How can we parallelize it, to get extra speedup, and also deal with agent autonomy?

### Precomputed signaling

- **Obstacle to parallelization:** To compute the agent $\ell$ rollout control we need the rollout controls of the preceding agents $i < \ell$
- **Signaling remedy:** Use precomputed substitute “guesses” $\hat{\mu}_i(x)$ in place of the preceding rollout controls $\tilde{\mu}_i(x)$

### Signaling possibilities

- **Use the base policy controls for signaling** $\hat{\mu}_i(x) = \mu_i(x), i = 1, \ldots, \ell - 1$ (this may work poorly)
- **Use a neural net representation of the rollout policy controls for signaling** $\hat{\mu}_i(x) \approx \tilde{\mu}_i(x), i = 1, \ldots, \ell - 1$ (this requires training/off-line computation)
- **Other, problem-specific possibilities**
The Pitfall of Using the Base Policy for Signaling

Two spiders trying to catch two stationary flies in minimum time

- The spiders have perfect vision/perfect information. The flies do not move.
- **Base policy for each spider**: Move one step towards the closest surviving fly

Performance of various algorithms

- **Optimal policy**: Split the spiders towards their closest flies
- Standard rollout is optimal for all initial states (it can be verified)
- Agent-by-agent rollout is also optimal for all initial states (it can be verified)
- Agent-by-agent rollout with base policy signaling is optimal for “most” initial states, with **A SIGNIFICANT EXCEPTION**
- When the spiders start at the same location, the spiders oscillate and never catch the flies
Damage level of each site is unknown, except when inspected. It deteriorates according to a known Markov chain unless the site is repaired (this is a POMDP).

Control choice of each robot: Inspect and repair (which takes one unit time), or inspect and move to a neighboring site.

State of the system: The set of robot locations, plus the belief state of the site damages.

Stage cost at each unrepaired site: Depends on the level of its damage.
Videos: Multirobot Repair in a Network of Damaged Sites
(Agents Start from the Same Location)

Cost comparisons

- Base policy cost: 5294 (30 steps)
- Multiagent rollout: 1124 (9 steps)
- Multiagent Rollout with base policy signaling: 31109 (Never stops)
- Multiagent Rollout with neural network policy signaling: 2763 (15 steps)

We will return to this problem in the future (in the context of infinite horizon policy iteration)
About the Next Lecture

We will cover:

- Rollout algorithms for constrained deterministic problems
- Applications in combinatorial and discrete optimization

Note on today’s and next lectures:

- The material on rollout and MPC are minimally covered in the class notes. The book "Lessons from AlphaZero ..." has more material.

Homework: Exercise 1.3 of latest version of class notes; due Sunday, Feb. 27

About questions on your project

- Send me email (dbertsek@asu.edu)
- Make appointment to talk by zoom (there are no fixed office hours in this course)