

The City College of New York

CSC 59867-E: Senior Project II

AI Agents for Decision Making in the Real World

Navigating Open-Ended Problems, Establishing New Metrics, and Reliability

Saptarashmi Bandyopadhyay

By Saptarashmi Bandyopadhyay
Email: sbandyopadhyay@ccny.cuny.edu
Assistant Professor of Computer Science
City College of New York and Graduate Center at the City University of New York

October 19, 2026 CSC 59867 - E

Where we left off

- In-Depth Discussion of Physical AI Agents vis-a-vis Virtual AI Agents
  - Autonomous Drone Swarms
  - Self-Driving Vehicles
  - AR/VR/XR Hardware Robotic Agents
  - Self-Driving Labs
- Discussion of Failure Modes in Agentic Transportation settings that can generalize across applications
  - Time-To-Collision (TTC)
  - Static Right-of-Way
  - Fixed Distance Spacing
- Quantum/Non-Deterministic Control of AI Agents
- Concluded how real-world physical and edge environments of Agents break static dataset assumptions
- Impactful Student Pitches for their Class Research Project

Sabtarashmi Bandyopadhyay

Lecture Agenda

Introduction to Open-Ended Problems	Bounding Open-Ended Problems with no oracle rewards	Establishing new metrics: Decision Goodness in Systems	Quantifying Agentic Reliability
01	02	03	04
• Formalization • Curricula	• Counterfactual Baseline Design • Value Approximation via k-step rollout	• Decision Goodness • Tracking System Parameters like Energy and Memory • Tracking Algorithmic Parameters like Complexity	• Seed-Stability • Multi-Agent Verification

Saptarashmi Bandyopadhyay

- Goal: To derive the optimal rollout/verification horizon k* that minimizes joint decision error and tail-latency cost (combining Bertsekas 2020 & Harchol-Balter 2024)
- Reminder: 1 page abstract report due per team by Friday October 23 as declared previously

Course Material References

- Rollout, Policy Iteration, and Distributed Reinforcement Learning by Dimitri P. Bertsekas, 2020
  - https://web.mit.edu/dimitrib/www/dp_rollout_book.html
  - https://web.mit.edu/dimitrib/www/Rollout_Complete%20Book.pdf
  - Formalization of Open-Ended Problems and Look-ahead Rollouts
- Multi-Agent Reinforcement Learning Book by Stefano Albrecht, 2024
  - https://www.marl-book.com/#download/#download
  - Multi-Agent Evaluation Parameters under non-stationarity, Pareto frontiers for intractability, and seed variance.
- Introduction to Probability for Computing by Mor Harchol Balter, 2024
  - https://www.cs.cmu.edu/~harchol/Probability/chapters/HarcholBalterWholeBook.pdf
  - Model stochastic response times
- JAXMARL (NeurIPS 2024)
  - Throughput: https://neurips.cc/virtual/2024/poster/97649
- Multi-Agent LLM Debate (ICML 2024)
  - Reliability: https://arxiv.org/abs/2305.14325

Soham Bhadyopadhyay

Introduction

Saptarashmi Bandyopadhyay

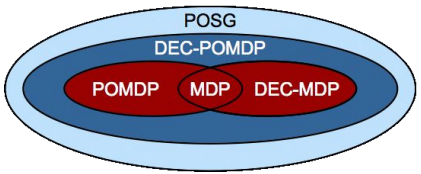
Closed-Loop Benchmarks vs. Open-Ended Problems

- Closed-Loop Benchmarks have a problem description with a clear answer
  - Could be as simple as “4+5 = 9”
- Why Senior Project II Is Harder Than Standard Closed-Loop Coursework
  - Closed-Loop Coursework (Standard Benchmark Environments)
    - Fixed state space S
    - Small discrete action space A
    - Stationary transitions P(s'|s,a)
    - Clean scalar oracle reward R(s,a) handed to you by a simulator
  - Open-Ended Senior Project Coursework (Real-World Environments)
    - Partial Observability (DEC-POMDP): Noisy multimodal sensors (audio, vision, telemetry)
    - Ill-Defined Objectives: No single ground-truth label for 'safe drone flight' or 'efficient grid'
    - System Constraints: Actions must execute within strict latency, memory, and Watt budgets
- The focus of your class-research projects will be focusing on a closed-loop subsection of an open-ended problem, bounding the state-action space of an end-to-end agent with comparative analysis

Sriparishmi Bandyopadhyay

Formalization of Open-Ended Problems

- Novelty and innovation in the real world is open-ended shaping how humans interact and evolve
- Our human-intelligence originates from this open-ended co-evolution among other organisms and the dynamic environment focusing on solving problems (Recall solution concepts taught in Game Theory lectures)
- Why not improve the capabilities of virtual and physical AI Agents with similar open-ended co-evolving steps?
- Such open-ended steps may result in agents solving an unboundedly large set of challenges
  - Leading to surprising emergent scenarios that might not have been considered when designing the learning system
    - E.g. the simulation-to-real (sim2real) gap taught in Physical AI Agents lectures
- The dynamism of the environment is considered as a separate parameter in this underspecified POMDP
- Sketch of open-endedness
  - Solve a meta-game with current policy population
  - Learn a new policy with Reinforcement Learning as a best response to the meta-strategy of populations
  - Expand the population reflecting the dynamism of the environment with the new policy



Source: http://rbr.cs.umass.edu/camato/decpomdp/overview.html

Sandeep Chandra Bhuyan Upadhyay

Curriculum

- A curriculum has a goal to train agents at the optimal frontier of its capabilities by dynamically adjusting
  - task distributions (problems specified with an open-ended reward)
  - environment configurations
  - opponent difficulty
- Recall the autonomous drug discovery problem taught earlier in class
  - We don't know the best drug design/formulations when we start discovering
  - We have a well-formulated regiment (a curriculum) to test possible drug formulations at scale
- A group of agents (currently you as students) discover new solutions following a curriculum by
  - Self-Play (Playing with past iterations of yourself e.g. learning from homeworks and classworks)
  - Exploration Bonuses (Extra Credits due to a major in-class discovery)
  - Unsupervised Environment Design (Exploring an unknown problem and mapping it to your class knowledge)
- A curriculum is dynamic to stay with the times in helping agents advance their goals

Sanmay Ganguly Bandyopadhyay

Bounding Open-Ended Problems

Saptarashmi Bandyopadhyay

Decomposing Open-Ended Problems with Rollout

- A roll-out is the attempted best strategy (policy) for agents from the current step to the next step
- How to Decompose Open-Ended Problems into Solvable Scientific/Engineering Pipelines
  - Step 1: Counterfactual Baseline Design:
    - Define a fast, deterministic domain heuristic policy π_{base} (e.g., greedy routing, PID controller, or zero-shot prompt) (no AI training is involved, just rules)
  - Step 2: Value Approximation via k-Step Rollout (Bertsekas, 2020):
    - Instead of learning a cumulative optimal reward Q*(s,a) from scratch across an astronomical state space, evaluate candidate actions using k-step lookahead with your base heuristic
    - Find your best policies k-steps ahead

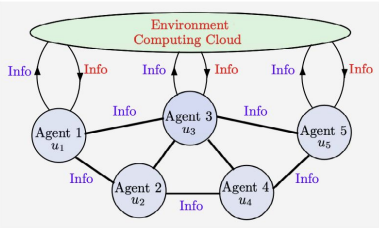


Figure 1.4.5 Schematic illustration of a multiagent problem. There are multiple "agents," and each agent $\ell = 1, \ldots, m$, controls its own decision variable u_ℓ. At each stage, agents exchange new information and also exchange information with the "environment," and then select their decision variables for the stage.

In particular, we will view each component u_ℓ, $\ell = 1, \ldots, m$, as being chosen from within $U_\ell(x)$ by a separate "agent" (a decision making entity). For the sake of the following discussion, we assume that each set $U_\ell(x)$ is finite. Then the one-step lookahead minimization of the standard rollout scheme with base policy μ is given by

$$\tilde{u} \in \arg\min_{u \in U(x)} E_w\Big\{g(x,u,w) + \alpha J_\mu\big(f(x,u,w)\big)\Big\}, \qquad (1.58)$$

Cost Improvement Property For Rollouts

- A k-step rollout (k being an integer >= 1) over a domain heuristic guarantees monotonically better decisions
- Computation stays bounded till the final trajectory
- Please refer to the detailed proof in page 247 of (Bertsekas, 2020)

Proposition 3.3.1: (Cost Improvement for Constrained Rollout) Assume that the base heuristic is sequentially improving and generates a feasible complete trajectory starting from the initial state $\tilde{y}_0 = \tilde{x}_0$, i.e., $R(\tilde{y}_0) \in C$. Then for each k, the set $U_k(\tilde{y}_k)$ is nonempty, and we have

$$\begin{aligned} G\big(R(\tilde{y}_0)\big) &\geq G\big(T_0(\tilde{y}_0, \tilde{u}_0)\big) \\ &\geq G\big(T_1(\tilde{y}_1, \tilde{u}_1)\big) \\ &\geq \cdots \\ &\geq G\big(T_{N-1}(\tilde{y}_{N-1}, \tilde{u}_{N-1})\big) \\ &= G(\tilde{y}_N), \end{aligned}$$

where

$$T_k(\tilde{y}_k, \tilde{u}_k) = \big(\tilde{y}_k, \tilde{u}_k, R(\tilde{y}_{k+1})\big);$$

cf. Eq. (3.32). In particular, the final trajectory $\tilde{y}_N$ generated by the constrained rollout algorithm is feasible and has no larger cost than the trajectory $R(\tilde{y}_0)$ generated by the base heuristic starting from the initial state.

Saptarashmi Bandyopadhyay

Establishing New Metrics

Decision Goodness



System Parameters

Balance

Evaluating Decision Goodness vs System Parameters

- Why is the Task Reward insufficient for Real World Agents
  - Recall AR Agents where an Agent can have very high accuracy but is unusable on an edge robot/AR headsets if each decisions have long latency which exhausts battery and memory (VRAM)
- Your projects should evaluate both the quality of the decisions vis-a-vis system parameters)
  - Decision Goodness
    - Task reward, convergence speed, sample efficiency (reward over periodic steps)
    - Qualitative behavior: Trajectory interpretability and edge-case sanity checks
  - System Parameters (further details can be found in Harchol-Balter, 2024)
    - Inference and Rollout Latency
    - Resource Footprint: Peak VRAM, model parameter count (quantization vs FP16 encoding), energy consumption

Saptarshi Bhattacharyay

Formulating a Compositional Real-World Agent Metric

- Moving from Unconstrained Accuracy to Constrained Optimization
  - Often Standard AI Objectives ignore Hardware Realities where we just calculate the cumulative reward
  - $\max_{\pi} \mathbb{E}[R_{task}(\pi)]$
- Real-World Systems Objective
  - $\min_{(\pi, k)} J(k) = L_{error}(k) + \lambda * T_{system}(k)$
  - Where
    - k = number of lookahead rollout steps (Bertsekas, 2020)
    - Could be the number of steps in few-shot learning or multi-agent debates
    - or multi-agent debate rounds (ICML 2024).
    - $L_{error}(k)$ = expected decision suboptimality / error after k verification steps.
    - $T_{system}(k)$ = end-to-end execution latency (compute + queueing + network).
    - $\lambda > 0$ = mission urgency weight (cost per millisecond of delay)
      - high in drone collision avoidance
      - lower in offline protein design

Sanmitra Bhattacharyya

Computing Optimal Rollout Horizon

- Read the following problem carefully
- You are deploying a drone controller (an autonomous AI Agent) that performs k steps of lookahead rollout for multi-agent verification before acting
- Decision Error Model: Each rollout step reduces expected decision loss exponentially:
  - $L_{error}(k) = L_0 * e^{\{-\mu k\}}$ (where $L_0 = 100$ is base heuristic loss, $\mu = 0.5$ is decay rate)
- System Latency Model: Each rollout step requires $\tau = 10$ ms of GPU inference which is communicated with a weighted urgency penalty $\lambda = 0.25$ per ms
  - $T_{system}(k) = \lambda * \tau * k$
- **Our Tasks**
  - Write the total composite cost function $J(k) = L_{error}(k) + T_{system}(k)$
  - Use calculus ($dJ/dk = 0$ and $d^2J/dk^2 > 0$) to derive the analytical formula for the optimal rollout horizon k^*
  - Plug in the numbers ($L_0 = 100$, $\mu = 0.5$, $\lambda = 0.25$, $\tau = 10$, $\ln(20) \approx 3.0$) to find the number of rollout steps k^* and total latency (reminder that k is an integer)

Sudarshini Pandyopadhyay

Solution (Part 1 - Analytical Calculus Derivation)

- Step 1: Total Cost Function J(k):
  - $J(k) = L_0 * e^{\{-\mu k\}} + \lambda * \tau * k$
- Step 2: Calculate the Critical Point (First Derivative dJ/dk = 0):
  - Differentiate J(k) with respect to rollout depth k (an integer)
    - $dJ / dk = -\mu * L_0 * e^{\{-\mu k\}} + \lambda * \tau = 0$
  - Compute the exponential term from the above formulation
    - $e^{\{-\mu k\}} = (\mu * L_0) / (\lambda * \tau)$
  - Take the natural logarithm (ln) of both sides to solve for the optimal k*
    - $k^* = (1 / \mu) * \ln((\mu * L_0) / (\lambda * \tau))$
- Step 3: Ensure that global minima has been reached via Second Derivative:
  - $d^2J / dk^2 = \mu^2 * L_0 * e^{\{-\mu k\}} > 0$ (Strictly positive for all $k > 0$ since $L_0, \mu > 0$ as provided earlier)

Sanjaya Krishna Bandyopadhyay

Solution (Part 2 - Numerical Analysis)

- Step 4: Plug in the provided System Parameters
  - Given: $L_0 = 100$, $\mu = 0.5$, $\lambda = 0.25$, $\tau = 10$ ms, $\ln(20) \approx 3.0$
  - Compute the ratio inside the logarithm:
    - $(\mu * L_0) / (\lambda * \tau) = (0.5 * 100) / (0.25 * 10) = 50 / 2.5 = 20$
  - Calculate optimal continuous rollout depth k^*
    - $k^* = (1 / 0.5) * \ln(20) = 2 \times 3 = 6$ rollout steps!
- Step 5: Plug in other values of k for numerical analysis, (k = 6 vs. k = 1 vs. k = 16)
  - At k = 1 step: $J(1) = 100 * e^{(-0.5*1)} + 0.25 * 10 * (1) \approx 60.65 + 2.50 = 63.15$ (High decision error!)
  - At $k^* = 6$ steps: $J(6) = 100 * e^{(-0.5*6)} + 0.25 * 10 * (6) \approx 4.98 + 15.0 = 19.98$ (GLOBAL MINIMUM — 60 ms latency)
  - At k = 16 steps: $J(16) = 100 * e^{(-0.5*16)} + 0.25 * 10 * (16) \approx 0.03 + 40.0 = 40.03$ (Latency penalty dominates!)
- Engineering Lesson
  - Thinking forever ruins real-time performance
  - k^* balances accuracy and latency

Soptarshi Bandyopadhyay

Agentic Reliability, Statistical Rigor and Verification

Saptarashmi Bandyopadhyay

Quantifying AI Agent Reliability

- Why average return across 1 training run is scientifically invalid?
  - AI Agents in the virtual (even more so in the physical world) exhibit high variance across random seeds and prompt perturbations
- Three Reliability Checks to guide evaluation in your class research projects
  - Seed & Run Stability (Reproducibility)
    - Evaluate across N ≥ 3 independent random seeds
    - Report Mean ± Standard Deviation (μ ± σ) reward or Interquartile Mean (IQM).
  - Out-of-Distribution (OOD) & Noise Robustness
    - Inject Gaussian sensor noise, dropped packets, or unseen test states
    - Measure performance drop: Reliability Ratio = R_{noisy} / R_{clean}
  - Multi-Agent Verification (from the Multi-Agent LLM Debate discussion during Senior Project I)
    - For generative agents, you can cross-check high-stakes actions with a verifier/critic before physical execution

Sambuddha Bandyopadhyay

Reliability Trap: Metric Gaming and Goodhart's Law

- Goodhart's Law in AI Agents
  - When a measure becomes a target, it ceases to be a good measure
  - If you design an initial underspecified/flawed proxy reward R_{proxy} for an open-ended problem, your AI agent will exploit loopholes in the formula rather than solving the real task!
  - Build good rewards
  - Build good sandboxes
  - Report your updates every 2 weeks in the biweekly progress updates
- Real-World Examples
  - Robotic Agent, rewarded for an acceptable moving forward velocity, learns to fall over tall obstacles to maximize center-of-mass speed for 1 second
  - Robotic Boat going round and round at the source as it's reward to reach the destination was underspecified to track movements from 1 step to the next step
  - Smart Grid Agent, rewarded solely for zero line overload, opens all circuit breakers, thereby causing a total blackout with zero line current!
- Keep a separate Held-Out Reliability / Safety Metric (e.g. constraint violation rate)
  - Do not have the agent optimize it directly during training
  - Track this safety metric continuously

Lakshmi Bandyopadhyay

Summary

- Open-Ended Problems:
  - Formulate and bound the state,action, observation space along with environmental dynamism
  - Compute a Bertsekas k-step rollout instead of a fast domain heuristic/rule
- New Compositional Metrics: Evaluate agents on both Decision Goodness and System Costs
  - An example usecase is to balance accuracy vs. latency
  - Workout the math done in class carefully for classwork on October 26
- Agentic Reliability: Report
  - multi-seed variance (μ ± σ)
  - guard against Goodhart's Law
  - Remember to write unit tests taught in Senior Project I

Saptarashmi Bandyopadhyay

Questions?

Saptarashmi Bandyopadhyay