The City College of New York

CSC 59866-E: Senior Project I

AI Agents for Decision Making in the Real World

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

Saptarashmi Bandyopadhyay

February 11, 2026 CSC 59866

Distributed Processing for AI Agents: Modalities (tabular, graphical, multi-modal)

Saptarashmi Bandyopadhyay

Today's Agenda

Hierarchy of Decision-Making Models and Socially Intelligent Agents

The Scale Problem: Why single-process Python isn't enough.

Distributed Paradigms: Data Parallelism vs. Model Parallelism, Synchronous vs. Asynchronous.

Modalities:

- Tabular (The classic way).
- Graphical (The relational way).
- Multimodal (The future).

MAJOR ANNOUNCEMENT: Project Group & Topic Assignments.

Santarashmi Bandyopadhyay

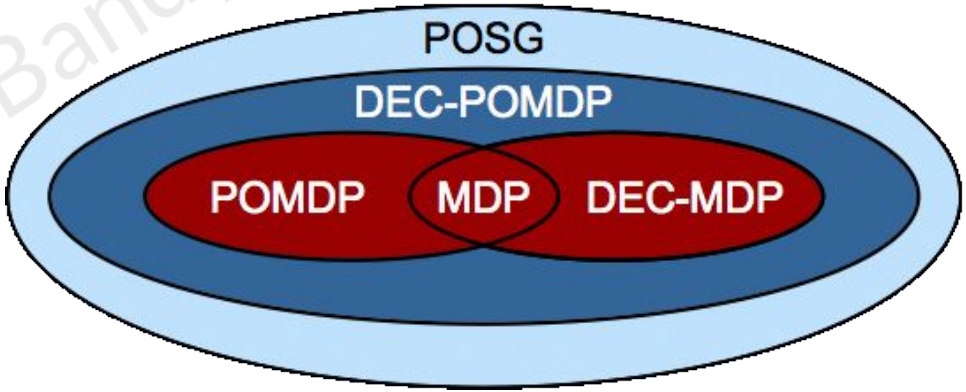
Hierarchy of Decision-Making Models and Socially Intelligent Agents

Saptarashmi Bandyopadhyay

Decision-Making Models

This Venn Diagram visualizes the mathematical relationship between single-agent, multi-agent, fully observable, and partially observable environments.

As we move outward in the diagram, we relax constraints (e.g., assuming full observability) and increase complexity (e.g., adding multiple agents).

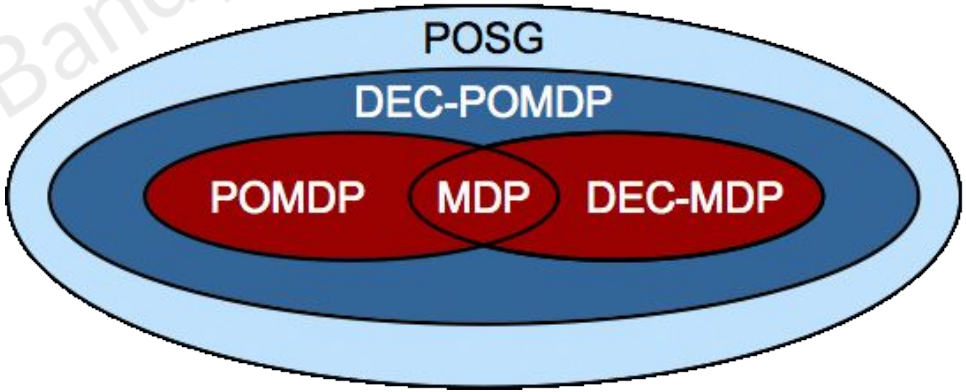


Sobtarashmi Bandyopadhyay

Markov Decision Process

Agents: Single ($n = 1$).
Observability: Full (The agent knows the exact state s_t).
Objective: Maximize cumulative reward.
Tuple: $\langle S, A, T, R, \gamma \rangle$

- S : State space
- A : Action space
- T : Transition function
- R : Reward function



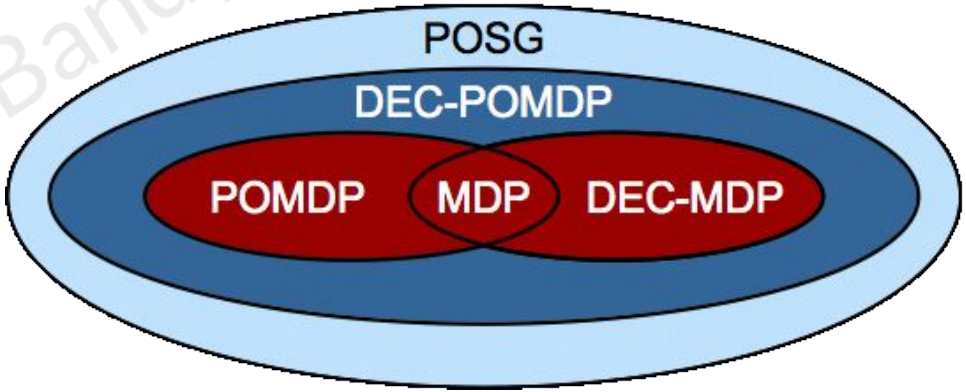
Saptarashmi Bandyopadhyay

Partially Observable MDP (POMDP)

The left inner circle.

Definition:

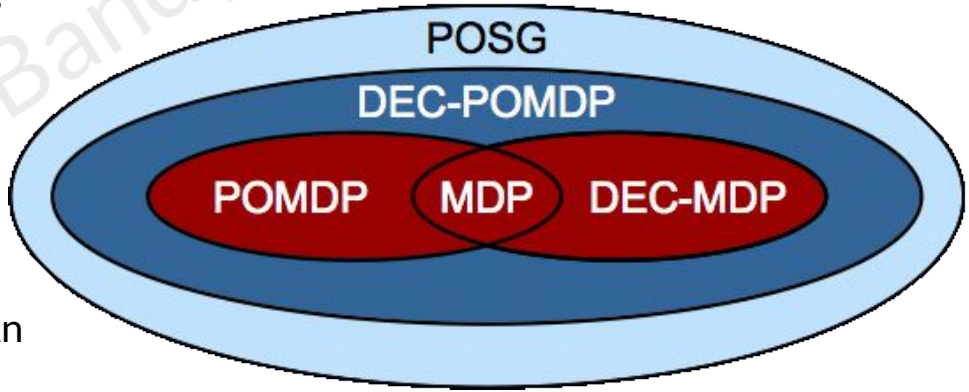
An MDP where the agent cannot directly observe the underlying state of the environment.



Saptarashmi Bandyopadhyay

Partially Observable MDP (POMDP)

- **Observability:** Partial. Agent receives **Observation** (O) correlated with state but not identical to it.
- **Belief States:** The agent maintains a probability distribution (belief) over possible states.
- **Subset Relation:** $MDP \subset POMDP$ (An MDP is just a POMDP where the observation gives you the exact state).



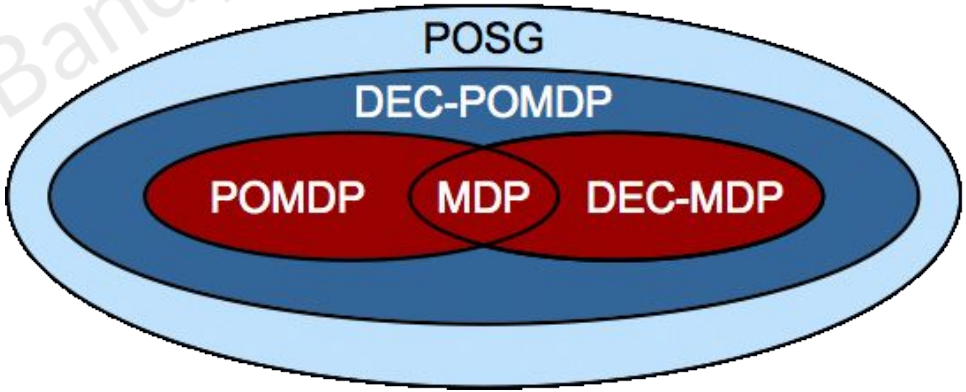
Sabtarashmi Bandyopadhyay

Decentralized MDP (DEC-MDP)

Context: The right inner circle.

Definition:

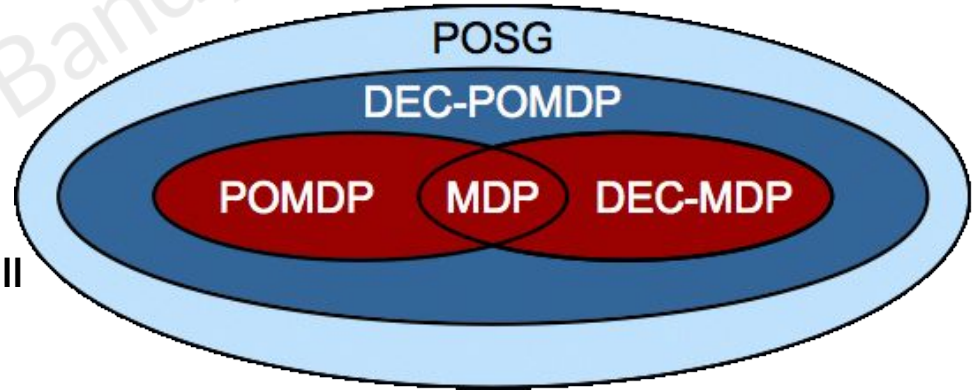
An extension of the MDP to multiple distributed agents who act based on local views but share a common goal.



Saptarashmi Bandyopadhyay

Decentralized MDP (DEC-MDP)

- **Agents:** Multiple ($n > 1$).
- **Decentralization:** Execution is decentralized (agents act independently), but planning can be centralized.
- **Observability:** Often implies **Joint Full Observability** (the combination of all agents' observations identifies the state), even if individual agents don't see everything.



- Subset Relation: $MDP \subset DEC - MDP$. (A DEC-MDP with 1 agent is an MDP).

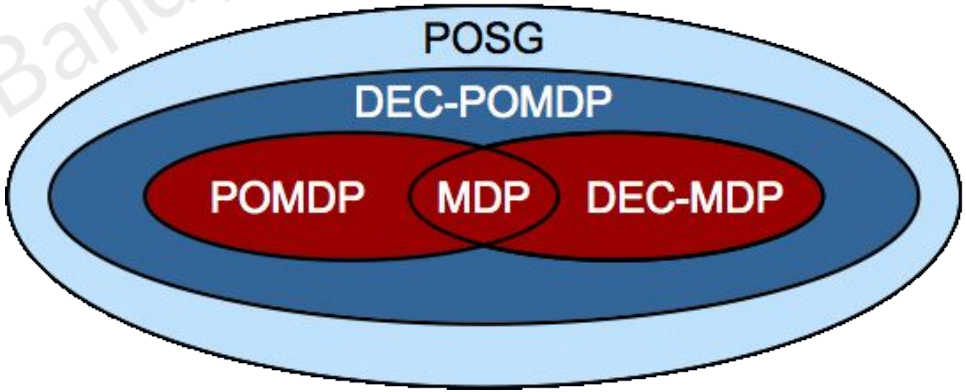
Sauptarashmi Bandyopadhyay

Decentralized POMDP (DEC-POMDP)

The dark blue circle encompassing POMDP and DEC-MDP.

Definition:

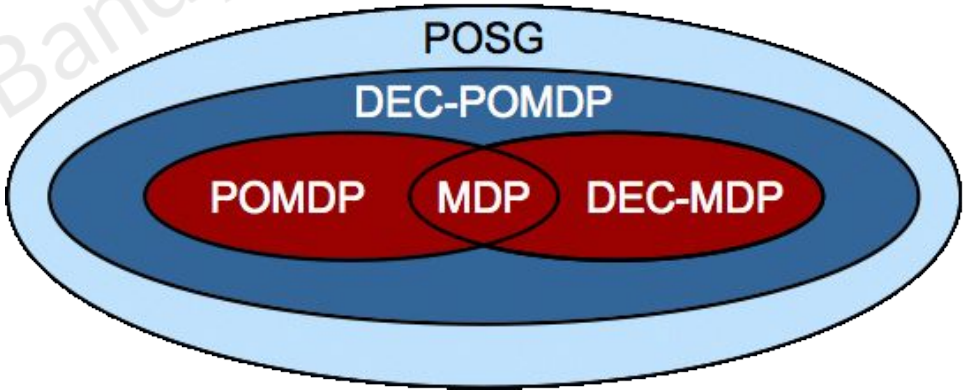
A framework for multi-agent coordination under uncertainty. Agents share a common reward but have different, partial views of the world.



Saptarashmi Bandyopadhyay

Decentralized POMDP (DEC-POMDP)

- **Goal:** Cooperative (All agents maximize the *same* global reward).
- **Challenge:** Agents must reason about the state of the world *and* the likely beliefs/actions of other agents.
- Known for being computationally intractable to solve optimally.



It contains **POMDPs** (Special case where n=1)and **DEC-MDPs** (Special case where observations jointly reveal the state).

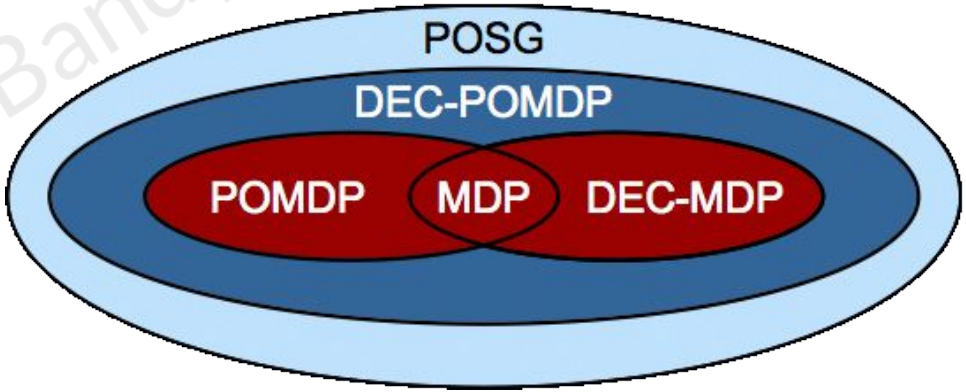
Saptarashmi Bandyopadhyay

Partially Observable Stochastic Game (POSG)

The outermost light blue circle.

Definition:

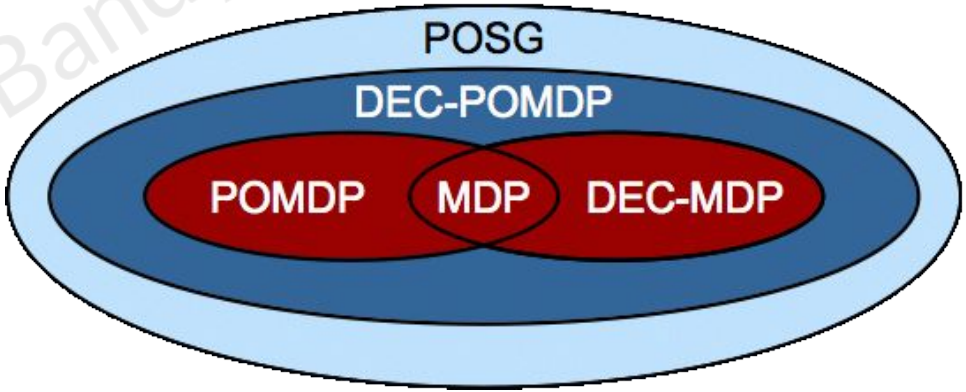
The most general framework for multi-agent decision-making.



Saptarashmi Bandyopadhyay

Partially Observable Stochastic Game (POSG)

- **Reward Structure:** Unlike DEC-POMDPs, agents in a POSG may have **conflicting or self-interested goals** (General Sum Game).
- **Examples:** Poker, Robotics Soccer (competitive), Autonomous Driving (mixed motives).



Hierarchy Summary:

$$MDP \subset POMDP \subset DEC\text{-}POMDP \subset POSG$$

Saptarashmi Bandyopadhyay

Example: Socially Intelligent Agents

Operates in the **Partially Observable Stochastic Game (POSG)** domain.

- **Why POSG?**
  1. **Partial Observability:** Agents have **private types** (θ_i) that others cannot see. The "state" includes these hidden types.
  2. **General-Sum:** Agents have distinct, often conflicting payoff functions ($G(\theta_i) \neq G(\theta_{-i})$). It is not purely cooperative like a DEC-POMDP.

The Twist: While the *structure* is a POSG (competitive/mixed-motive), the *goal* is cooperative (minimizing Altruistic Regret).

- We are trying to find a cooperative equilibrium within a potentially competitive game structure.

Santanashmi Bandyopadhyay

Example: Socially Intelligent Agents

Why is this so difficult?

- **Theoretical Difficulty:** As shown in the Venn diagram, POSGs are the most general and computationally difficult class (often intractable/undecidable).
- **The Challenge:** A "Naive" Imitation Learning approach fails here because the AI agent doesn't know the partner's private type or the specific conventions (handshakes) the population uses.
- **Impossibility Result:** The paper proves that without specific assumptions (Consistency & Compatibility), you *cannot* learn to cooperate zero-shot. You might be playing a game where "cooperation" looks completely different depending on the invisible private type.

Santanshini Bandyopadhyay

Example: Socially Intelligent Agents

Imitate-Then-Commit Algorithm

Step 1: Imitation (The "POMDP" phase):

- Agent observes the population to learn the how to signal private types.
- It mimics a member of the population for $\tilde{T}$ steps to extract this hidden information.

Step 2: Commitment (The "MDP" phase):

- Once the type is inferred (state becomes "known"), the AI switches to a deterministic strategy that forces a Pareto-Optimal equilibrium.

Result: This strategy achieves an upper bound on sample complexity that beats generic IL

Soparashmi Bandyopadhyay

Distributed Processing for Agents

Saptarashmi Bandyopadhyay

Why Distributed?

The Problem: Training a decent Agent (e.g., AlphaGo, ChatGPT, or even a robust StarCraft bot) takes billions of interactions.

The Math:

- Time per Step (t_{step}) $\times$ Required Steps (N_{steps}) $=$ Total Training Time.
- If $t_{step} = 0.1s$ and $N_{steps} = 10^9 \rightarrow 3$ years to train.

The Solution: Distributed Processing. We cannot reduce N_{steps} easily, so we must perform steps simultaneously.

Sabyasachi Bandyopadhyay

The Two Main Architectures

1. Data Parallelism (The "Rollout" Worker):

- *Concept:* Copy the Agent to 100 CPUs. Each Agent plays the game independently. They send their data to a central "Learner" to update the brain.
- *Relevance:* This is how standard RL (PPO/A3C) works.

2. Model Parallelism (The "Giant Brain"):

- *Concept:* The Agent is too big for one GPU (e.g., a 70B parameter LLM). We split the *layers* across multiple GPUs.
- *Relevance:* Essential for "LLM Agents."

Saptarashmi Bandyopadhyay

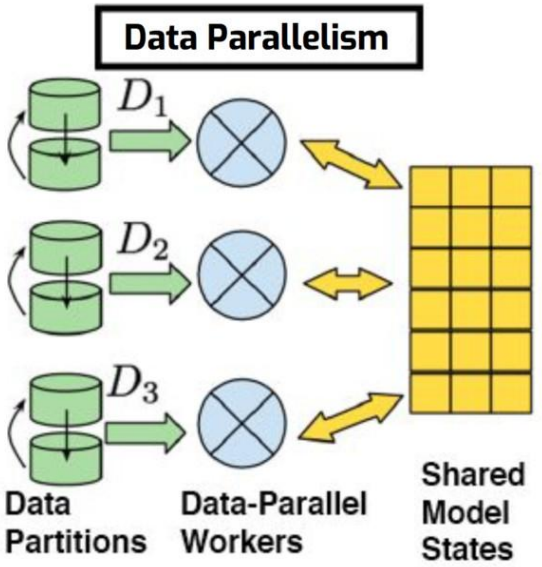
Data Parallelism

Copies of the same Agent across N devices (CPUs or GPUs).
How it Works:

- Each worker runs its own instance of the environment.
- They collect experience (data) independently and send it to a central "Learner" to update the shared brain.

Why we use it:

- **Throughput:** Increases experience collection linearly with N.
- **Relevance:** The standard approach for PPO, A3C, and most projects in this class (e.g., using JAX vmap).



Santarashmi Bandyopadhyay

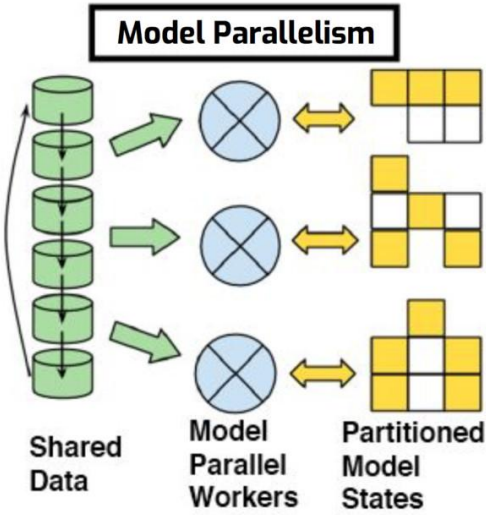
Model Parallelism

Split a **single** giant Agent across multiple GPUs.
How it Works:

- The model (e.g., 70B+ parameters) is too big for one GPU
- **Partitioning:** GPU 1 holds Layers 1-10, GPU 2 holds Layers 11-20, etc. Data flows through them like an assembly line.

Why we use it:

- **Capacity:** Essential for "LLM Agents" and Foundation Models that physically cannot fit on one device.
- **Trade-off:** High communication overhead (latency) between GPUs.



Saptarashmi Bandyopadhyay

Synchronous vs. Asynchronous (A3C vs. PPO)

Synchronous (PPO): Everyone waits for the slowest worker. "Stop, update weights, resume." (Stable, math is cleaner).

Asynchronous (A3C/Impala): Workers don't wait. They dump data and keep going. (Faster, but mathematically "messy" due to lag).

Your Projects: You will likely use **Synchronous** methods (via JAX/Gymnasium) because they are easier to debug.

Saptarashmi Bandyopadhyay

Synchronous Distributed Agents

Workers act like an orchestra. They collect data together and pause together.
Mechanism (e.g., PPO, A2C):

- **Step 1:** All workers collect N steps of experience.
- **Step 2:** They **pause** and send data to the Learner.
- **Step 3:** The Learner updates the model and sends new weights to *all* workers.
- **Step 4:** Workers resume.

Pros & Cons:

- **Stable:** Mathematically cleaner; everyone uses the same policy version.
- **Debuggable:** Easier to reproduce bugs.
- **The "Straggler" Problem:** The whole system waits for the slowest worker.

Saptarashmi Bandyopadhyay

Asynchronous Distributed Agents

Jazz musicians playing at their own tempo, updating the central brain whenever they are ready.
Mechanism (e.g., A3C, IMPALA):

- Workers run independently without waiting for others.
- When a worker finishes a batch, it pushes gradients to the **Parameter Server** immediately and pulls the latest weights.

Pros & Cons:

- **Fast:** No waiting for slow workers; maximum hardware utilization.
- **Stale Gradients:** Worker may calculate gradients using an old model, destabilizing learning.
- ❌ **Chaos:** Hard to debug; runs are non-deterministic.

Saptarashmi Bandyopadhyay

Modalities (What the Agent Sees)

Saptarashmi Bandyopadhyay

Modality 1 - Tabular/Vector Data

Definition: The state is a list of numbers.

- *Example:* GridWorld (x, y), Stock Market (Price, Volume).

Pros: Fast, low memory, interpretable.

Cons: Lose spatial/relational information. A list of pixels is not the same as an image.

TL;DR: Good for simple control tasks, bad for the "Real World."

Saptarashmi Bandyopadhyay

Modality 2 - Graphical Data (GNNs)

The Insight: The world is not a list; it's a network.

Example: Supply Chain Agents.

- Factory A $\rightarrow$ Truck B $\rightarrow$ Store C.
- If Truck B breaks, it affects A and C directly.

Graph Neural Networks (GNNs):

- Agents process "Nodes" and "Edges."
- Allows the agent to understand *relationships* and *topology* (who is connected to whom?).

Research Hotspot: Multi-Agent Communication is a graph problem!

Saptarashmi Bandyopadhyay

Modality 3 - Multi-Modal

Definition: Combining disparate data streams.

Visual-Language Models (VLM): "Red light" (Image) + "Stop sign text" (Language) $\rightarrow$ Action: Brake.

Audio-Visual: Hearing a siren + Seeing the ambulance.

The Challenge: Alignment. How do you map the word "Dog" (Text embedding) to a picture of a Dog (Pixel embedding)?

If you are building a "Robotic Agent" or "AI Agent Assistant," you *must* handle multi-modality.

Sabtarashmi Bandyopadhyay

Group Project Assignments

Saptarashmi Bandyopadhyay

Project Groups & Topics

Protocol:

1. I will display the 10 Groups.
2. Find your partner immediately after class.
3. **Task 1:** Exchange emails/Discord handles/Your preferred way to communicate.
4. **Task 2:** Set up a weekly meeting time outside of class.

Saptarashmi Bandyopadhyay

Groups 1-5

- **Adaptive Resource Allocation for LLM Agents in Edge Networks**
  - Arsh Anand & Farida Balogun
- **Benchmarking Multi-Agent Reinforcement Learning (MARL) in Augmented Reality**
  - Karthikeya Reddy Basavanagoudgari & Hamim Choudhury
- **Resilient Consensus for Connected Autonomous Vehicles (CAV)**
  - Marcley Colin & Alhassana Diallo
- **AI Agents for Distributed Chip Design**
  - Johir Hossain & Linwei Zheng
- **AI Agents for Distributed Discovery of Materials and Chemicals for Downstream Tasks like Drug Discovery**
  - Parthkumar Joshi & Alexis Juarez Gomez

Subhashis Bandyopadhyay

Groups 6-10

- **Fault-Tolerant Multi-Agent Systems for Vision-Language Models**
  - Fardin Khandaker & Xin Fan Li
- **Game-Theoretic Coordination in Distributed AR Environments**
  - Yuki Li & Minning Liu
- **Robust Multi-Agent Communication in Wireless MIMO Networks**
  - Mahdi Mahin & Amir Nabiyev
- **Interpretable Multi-Agent Coordination for Urban Mobility**
  - Sajid Patwary & Kenneth Romero Linares
- **Energy-Efficient Protocols for Distributed AI on Edge Devices**
  - Sri Suvetha Meenaa Subramanian & Atif Tausif

Sanjukta Bandyopadhyay

Reminders and Next Steps

Feb 16: No Class (College Closed).

Next Class (Feb 18): Systems: Latency and Bandwidth balancing.

Action Items:

1. Meet your group.
2. Start reading the "Pillar Papers" for your specific topic (Posted on the Website soon).
3. If your group wants to switch to an alternate topic please let me know!

Sabtarashmi Bandyopadhyay

Questions?

Saptarashmi Bandyopadhyay