A COMPREHENSIVE SURVEY OF MARL DESIGN PARADIGMS

From Core Ideas to Concrete Algorithms

JULIA WANG

Every coordination mechanism is a reflection of human organizational wisdom - from monarchies to democracies, from corporations to communities.

OUTLINE FOR TODAY’S LECTURE

THE TWO FUNDAMENTAL CHALLENGES OF MARL

All paradigms are attempts to solve these core issues.

1. **Non-Stationarity**
   - As other agents learn, the environment dynamics change from any single agent's perspective.
   - This breaks the core Markov assumption of RL.
   - **Question:** How do you learn when the rules are always changing?
2. **Scalability**
   - The joint action space grows exponentially with the number of agents ($|\mathcal{A}|^N$).
   - A central controller that plans over this joint space is computationally intractable.
   - **Question:** How do we coordinate without exponential cost?

PARADIGM 1: CTCE - THE "GOD CONTROLLER"
THEORETICAL IDEAL, PRACTICAL BOTTLENECK

Core Idea: Treat the entire multi-agent system as a single "meta-agent." A single, monolithic policy is learned and executed by a central controller.

Major Research Directions:

1. **Joint Action Space Methods**
2. **Multi-Agent MDP (MMDP) Methods**

Key Trade-off

Maximum performance potential at the cost of minimum scalability. Primarily used as a theoretical upper bound.

1.1 JOINT ACTION SPACE METHODS - CORE IDEAS

Fundamental Approach: Directly work in the joint action space $\mathcal{A}_1 \times \mathcal{A}_2 \times \ldots \times \mathcal{A}_N$.

Key Characteristics:

- Learn a single value function $Q(s, a_1, a_2, \ldots, a_N)$ or policy $\pi(a_1, a_2, \ldots, a_N | s)$.
- Action selection requires optimization over the entire joint action space.
- Complete elimination of non-stationarity (since there's only one learner).
- Theoretical guarantee of finding optimal joint policies.

The Fatal Flaw: The joint action space size is $|\mathcal{A}_1| \times |\mathcal{A}_2| \times \ldots \times |\mathcal{A}_N|$, which grows exponentially with the number of agents.

1.1 ALGORITHM ANALYSIS: CENTRALIZED Q-LEARNING

Goal: Learn the optimal joint action-value function $Q^*(s, \mathbf{a})$ where $\mathbf{a} = (a_1, \ldots, a_N)$.

The Algorithm:

- Maintain a Q-table with entries $Q(s, a_1, a_2, \ldots, a_N)$.
- Update rule: $Q(s, \mathbf{a}) \leftarrow Q(s, \mathbf{a}) + \alpha[r + \gamma \max_{\mathbf{a}'} Q(s', \mathbf{a}') - Q(s, \mathbf{a})]$
- Action selection: $\mathbf{a}^* = \arg\max_{\mathbf{a}} Q(s, \mathbf{a})$

Why It Fails: With just 10 agents, each with 2 actions, the Q-table has $2^{10} = 1024$ columns for every state. The max operation requires evaluating all 2^{10} joint actions. This exponential complexity makes it intractable for realistic problems, directly motivating the need for decentralized approaches.

1.2 MULTI-AGENT MDP METHODS - CORE IDEAS

Fundamental Approach: Formalize the multi-agent problem as an extension of standard MDPs by expanding the state and action spaces.

Key Characteristics:

- The state space becomes the joint state: $S = S_1 \times S_2 \times \ldots \times S_N$.
- The action space is the joint action space: $A = A_1 \times A_2 \times \ldots \times A_N$.
- Standard single-agent algorithms can be applied directly.
- Provides a clean theoretical framework for analysis.

The Challenge: Both state and action spaces can grow exponentially, making practical implementation impossible for large numbers of agents.

1.2 ALGORITHM ANALYSIS: MMDP FORMALIZATION

Goal: Provide a theoretical framework for multi-agent problems in fully observable environments.

The Formalization:

- Joint state: $\mathbf{s} = (s_1, s_2, \ldots, s_N)$
- Joint action: $\mathbf{a} = (a_1, a_2, \ldots, a_N)$
- Transition function: $P(\mathbf{s}'|\mathbf{s}, \mathbf{a})$
- Joint reward: $R(\mathbf{s}, \mathbf{a})$

Significance: This work doesn't propose a specific algorithm but rather establishes the theoretical foundation for treating multi-agent problems as single-agent problems with expanded spaces. It demonstrates that while theoretically sound, the exponential growth in complexity makes centralized approaches impractical, setting the stage for the development of CTDE and DTDE paradigms.

PARADIGM 2: CTDE - "LEARN TOGETHER, ACT ALONE"
THE DOMINANT MODERN PARADIGM

Core Idea: Use centralized information during training to guide learning, but execute policies in a decentralized way.

Major Research Directions:

1. Value Function Factorization
2. Centralized Critic Methods
3. Communication Mechanisms
4. Attention-based Methods
5. Role Assignment & Hierarchical Methods

This paradigm represents the sweet spot between performance and scalability.

2.1 VALUE FUNCTION FACTORIZATION - CORE IDEAS

Fundamental Approach: Decompose the intractable global value function into a combination of learnable local functions.

Key Characteristics:

- Learn individual agent utilities: $Q_i(o_i, a_i)$ for each agent i.
- Learn a mixing function: $Q_{tot} = f(Q_1, Q_2, \ldots, Q_N; s)$.
- The Individual-Global-Max (IGM) principle:
  $\arg\max_{\mathbf{a}} Q_{tot} = (\arg\max_{a_1} Q_1, \ldots, \arg\max_{a_N} Q_N)$.
- Enables decentralized execution while learning centralized coordination.

The Challenge: Finding the right balance between expressiveness and maintaining the IGM property.

2.1 ALGORITHM ANALYSIS: QMIX

Goal: Learn a monotonic mixing function that satisfies IGM while being more expressive than simple addition.

The Architecture:

- Agent networks: Each agent i learns $Q_i(o_i, a_i)$.
- Mixing network: Takes all Q_i values and global state s as input.
- Hypernetwork: Generates mixing network weights from global state.
- Monotonicity constraint: $\frac{\partial Q_{tot}}{\partial Q_i} \geq 0$ for all i.

$$\operatorname*{argmax}_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1(\tau^1, u^1) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n(\tau^n, u^n) \end{pmatrix}$$

Key Innovation: The monotonicity constraint ensures that if an agent improves its local Q-value, the global Q-value cannot decrease. This elegantly preserves the IGM property while allowing for complex, non-linear mixing of agent utilities. The hypernetwork makes the mixing function state-dependent, enabling context-aware coordination.

2.2 CENTRALIZED CRITIC METHODS - CORE IDEAS

Fundamental Approach: Use centralized critics with global information to provide stable learning signals for decentralized actors.

Key Characteristics:

- Decentralized actors: Each agent has $\pi_i(a_i|o_i)$ for execution.
- Centralized critics: Access to global state s and/or all agents' actions during training.
- Stable policy gradients: Critics provide low-variance estimates.
- Flexible coordination: Can handle both cooperative and competitive settings.

The Advantage: Combines the stability of centralized evaluation with the scalability of decentralized execution.

2.2 ALGORITHM ANALYSIS: MADDPG

Goal: Extend DDPG to multi-agent settings using centralized critics for stable policy gradient updates.

The Architecture:

- Each agent i has actor $\pi_i(a_i|o_i)$ and critic $Q_i(s, a_1, \dots, a_N)$.
- Critic input: Global state and all agents' actions.
- Actor input: Only local observation.
- Policy gradient: $\nabla_{\theta_i} J(\theta_i) = \mathbb{E}[\nabla_{\theta_i}\pi_i(a_i|o_i)\nabla_{a_i} Q_i(s, a_1, \dots, a_N)]$

Key Innovation: By giving each critic access to everyone's actions, the environment becomes stationary from each agent's perspective during training. This eliminates the non-stationarity problem while maintaining decentralized execution. Particularly effective in mixed cooperative-competitive scenarios.

2.3 COMMUNICATION MECHANISMS - CORE IDEAS

Fundamental Approach: Learn explicit communication protocols that allow agents to share information and coordinate.

Key Characteristics:

- Message generation: Agents create messages from their local observations.
- Message integration: Agents process received messages to inform their decisions.
- Differentiable communication: End-to-end learning of communication protocols.
- Bandwidth constraints: Often limited message size or communication frequency.

The Challenge: Learning what to communicate, when to communicate, and how to use received information effectively.

2.3 ALGORITHM ANALYSIS: TARMAC

Goal: Learn targeted communication where agents selectively attend to relevant teammates' messages.

The Architecture:

- Signature-based attention: Each agent creates a signature from its observation.
- Message generation: Agents create messages based on their local state.
- Attention mechanism: Uses signatures to compute attention weights over messages.
- Integration: Weighted combination of messages integrated into policy.

Key Innovation: Instead of broadcasting to all agents or using fixed communication graphs, TarMAC learns who to listen to. The signature-based attention mechanism allows agents to identify relevant teammates dynamically, enabling more efficient and scalable communication than simple broadcast approaches.

2.4 ATTENTION-BASED METHODS - CORE IDEAS

Fundamental Approach: Use attention mechanisms to selectively focus on relevant information from other agents.

Key Characteristics:

- Selective attention: Focus on most relevant agents/information.
- Dynamic weighting: Attention weights change based on context.
- Scalable processing: Handles variable numbers of agents.
- Interpretability: Attention weights provide insights into agent interactions.

The Advantage: Enables efficient processing of information from large numbers of agents without losing important details.

2.4 ALGORITHM ANALYSIS: MAAC

Goal: Use attention mechanisms in centralized critics to focus on relevant agents when computing Q-values.

The Architecture:

- Attention critic: Uses multi-head attention over other agents' embeddings.
- Agent embedding: Each agent's observation-action pair is embedded.
- Attention weights: Computed dynamically based on query agent and context.
- Value computation: Attended information used to compute Q-values.

Key Innovation: Instead of simply concatenating all agents' information, MAAC uses attention to dynamically focus on the most relevant agents. This makes the critic more scalable and helps it learn more nuanced coordination patterns by identifying which agents' actions are most important for evaluating a particular agent's decision.

2.5 ROLE ASSIGNMENT & HIERARCHICAL METHODS - CORE IDEAS

Fundamental Approach: Learn explicit role assignments or hierarchical structures to simplify coordination.

Key Characteristics:

- Role specialization: Agents learn to adopt specific behavioral patterns.
- Hierarchical structure: High-level coordination with low-level execution.
- Interpretable behavior: Clear understanding of each agent's function.
- Structured learning: Reduces the complexity of the joint policy space.

The Advantage: Provides structured coordination that is both interpretable and efficient to learn.

2.5 ALGORITHM ANALYSIS: ROMA

Goal: Learn a small set of role policies that agents can adopt to achieve structured team coordination.

The Architecture:

- Role policies: Small set of shared policies $\{\pi^1, \pi^2, \ldots, \pi^K\}$.
- Individual policies: Each agent also has its own policy π_i.
- Role selection: Agents choose which role to adopt based on context.
- Policy mixing: Final action distribution combines individual and role policies.

Key Innovation: By learning a small set of role policies, ROMA enables agents to develop high-level team strategies (e.g., "attacker," "defender," "support"). The role selection mechanism allows for dynamic assignment based on the situation, leading to more structured and interpretable team behavior while maintaining learning efficiency.

PARADIGM 3: DTDE - "EVERY AGENT FOR ITSELF"

MAXIMUM SCALABILITY, LIMITED COORDINATION

Core Idea: Each agent learns independently with no central coordination at any stage.

Major Research Directions:

1. Independent Learning
2. Opponent Modeling
3. Game-theoretic Methods
4. Meta-Learning
5. Distributed Communication

This paradigm is crucial for massive-scale systems or when centralization is impossible.

3.1 INDEPENDENT LEARNING - CORE IDEAS

Fundamental Approach: Each agent runs its own single-agent RL algorithm, treating other agents as part of the environment.

Key Characteristics:

- Complete independence: No sharing of information between agents.
- Environmental non-stationarity: Other agents' learning changes the environment.
- Scalability: Linear complexity in the number of agents.
- Simplicity: Easy to implement and parallelize.

The Challenge: Severe non-stationarity can lead to unstable learning and poor coordination.

3.1 ALGORITHM ANALYSIS: IQL WITH PARAMETER SHARING

Goal: Enable implicit coordination through shared network parameters while maintaining independent learning.

The Architecture:

- Shared parameters: All agents use the same network weights θ.
- Independent experience: Each agent has its own replay buffer and experiences.
- Independent gradients: Each agent computes gradients from its own data.
- Shared updates: All gradients are applied to the shared parameters.

Key Innovation: While agents learn independently, the shared parameters create implicit coordination. When agent i learns something useful, it modifies the shared policy, allowing agent j to benefit from that knowledge. This simple heuristic has enabled massive-scale successes like AlphaStar and is particularly effective when agents face similar situations.

3.2 OPPONENT MODELING - CORE IDEAS

Fundamental Approach: Explicitly model other agents' policies or behaviors to predict their actions.

Key Characteristics:

- Explicit modeling: Learn representations of other agents' policies.
- Prediction: Anticipate other agents' actions to inform own decisions.
- Adaptation: Adjust behavior based on observed opponent patterns.
- Recursive reasoning: "I think that you think that I think..."

The Challenge: Computational complexity grows with the sophistication of modeling, and opponents may also be learning.

3.2 ALGORITHM ANALYSIS: DRON

Goal: Learn to predict opponent actions and use these predictions to improve own policy.

The Architecture:

- Opponent predictor: Network head that predicts opponent's next action.
- Prediction loss: Supervised learning to predict observed opponent actions.
- Policy network: Uses current state and opponent prediction as input.
- Joint training: Both prediction and policy networks trained simultaneously.

Key Innovation: DRON explicitly predicts what the opponent will do next and uses this prediction as additional input to its own policy. This allows the agent to anticipate opponent moves and plan accordingly, leading to more strategic behavior. The challenge is that the prediction quality depends on the opponent's consistency and predictability.

3.3 GAME-THEORETIC METHODS - CORE IDEAS

Fundamental Approach: Apply game theory concepts to find stable joint policies, typically Nash equilibria.

Key Characteristics:

- Equilibrium concepts: Nash equilibrium, correlated equilibrium, etc.
- Stability: No agent can improve by unilaterally changing strategy.
- Rationality assumptions: Agents are assumed to be rational optimizers.
- Convergence guarantees: Theoretical guarantees under certain conditions.

The Challenge: Computing equilibria can be expensive, and multiple equilibria may exist.

3.3 ALGORITHM ANALYSIS: NASH Q-LEARNING

Goal: Learn Q-values that correspond to Nash equilibrium policies.

The Architecture:

- Joint Q-values: Each agent learns $Q_i(s, a_1, \ldots, a_N)$ for all agents.
- Nash computation: Solve for Nash equilibrium at each state.
- Equilibrium actions: Use Nash equilibrium mixed strategies for action selection.
- Bellman update: $Q_i(s, \mathbf{a}) \leftarrow Q_i(s, \mathbf{a}) + \alpha[r_i + \gamma V_i^{Nash}(s') - Q_i(s, \mathbf{a})]$

Key Innovation: Instead of using a simple max operator, Nash-Q computes the Nash equilibrium value $V_i^{Nash}(s')$ at each state. This provides theoretical stability guarantees but requires solving a matrix game at each state, making it computationally expensive and limiting its scalability.

3.4 META-LEARNING - CORE IDEAS

Fundamental Approach: Learn to adapt quickly to new tasks, environments, or teammates.

Key Characteristics:

- Learning to learn: Optimize for fast adaptation rather than single-task performance.
- Few-shot adaptation: Quick adjustment to new scenarios with minimal data.
- Generalization: Transfer knowledge across related tasks or teammates.
- Adaptation mechanisms: Gradient-based, memory-based, or optimization-based.

The Advantage: Enables rapid adaptation to new teammates or environments without retraining from scratch.

3.4 ALGORITHM ANALYSIS: MAML FOR MARL

Goal: Learn initial parameters that can be quickly adapted to new multi-agent tasks or teammates.

The Architecture:

- Meta-training: Train on a distribution of tasks/teammate types.
- Inner loop: Few gradient steps to adapt to specific task/teammate.
- Outer loop: Update initial parameters based on post-adaptation performance.
- Fast adaptation: Use adapted parameters for execution with new teammates.

Key Innovation: MAML finds initial parameters such that a few gradient steps can quickly adapt to new teammates or tasks. In MARL, this enables an agent to rapidly adjust to previously unseen teammates' behaviors, making it valuable for ad-hoc team formation and zero-shot coordination scenarios.

3.5 DISTRIBUTED COMMUNICATION - CORE IDEAS

Fundamental Approach: Enable local communication between agents without centralized coordination.

Key Characteristics:

- Local communication: Limited to neighbors or nearby agents.
- Decentralized protocols: No central communication server.
- Scalable architecture: Communication cost grows sub-linearly with agent count.
- Robustness: Resilient to communication failures or agent dropouts.

The Challenge: Designing effective protocols that enable coordination without global knowledge.

3.5 ALGORITHM ANALYSIS: NEIGHBOR-BASED INFORMATION SHARING

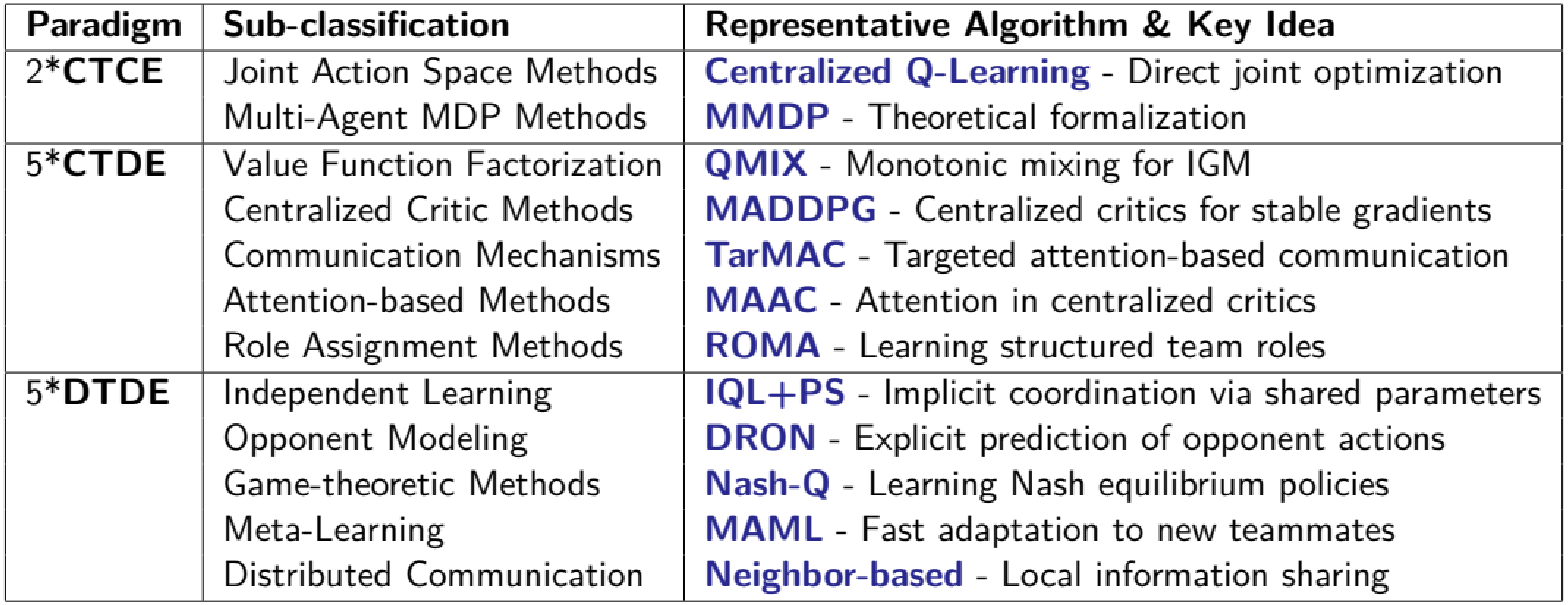
Goal: Enable local coordination through sharing information with nearby agents.

The Architecture:

- Communication graph: Defines which agents can communicate (e.g., spatial neighbors).
- Information sharing: Agents share feature representations or observations.
- Local aggregation: Each agent combines information from its neighbors.
- Decentralized updates: Each agent updates its policy based on local information.

Key Innovation: This approach enables swarm-like coordination where agents share information with nearby neighbors. Common implementations include averaging feature vectors or using graph neural networks to process neighbor information. This creates local coordination patterns that can emerge into global behavior without centralized control.

SUMMARY: A COMPREHENSIVE TAXONOMY OF MARL PARADIGMS



Paradigm	Sub-classification	Representative Algorithm & Key Idea
2***CTCE**	Joint Action Space Methods	**Centralized Q-Learning** - Direct joint optimization
	Multi-Agent MDP Methods	**MMDP** - Theoretical formalization
5***CTDE**	Value Function Factorization	**QMIX** - Monotonic mixing for IGM
	Centralized Critic Methods	**MADDPG** - Centralized critics for stable gradients
	Communication Mechanisms	**TarMAC** - Targeted attention-based communication
	Attention-based Methods	**MAAC** - Attention in centralized critics
	Role Assignment Methods	**ROMA** - Learning structured team roles
5***DTDE**	Independent Learning	**IQL+PS** - Implicit coordination via shared parameters
	Opponent Modeling	**DRON** - Explicit prediction of opponent actions
	Game-theoretic Methods	**Nash-Q** - Learning Nash equilibrium policies
	Meta-Learning	**MAML** - Fast adaptation to new teammates
	Distributed Communication	**Neighbor-based** - Local information sharing

HOMEWORK ASSIGNMENT 1

Please answer the following questions in a brief write-up (2-3 pages).

1. **Paradigm Deep Dive:**
   - Choose one sub-class from CTDE (e.g., Value Function Factorization) and one from DTDE (e.g., Opponent Modeling).
   - Explain the core motivation behind each approach and how they address the fundamental challenges of MARL.
   - Compare the representative algorithms from each sub-class in terms of computational complexity, coordination capability, and scalability.
2. **Scenario Design:**
   - Design a specific multi-agent scenario (e.g., autonomous vehicle coordination, distributed sensor networks, or multi-robot warehouse management).
   - Analyze which paradigm and sub-class would be most appropriate, considering factors like: number of agents, communication constraints, need for coordination, and computational limitations.
   - Justify your choice and discuss potential challenges in implementation.

PREVIEW FOR THE NEXT LECTURE

LESSON 3: LEARNING DYNAMICS - THE "EVOLUTIONARY GAME" BETWEEN AGENTS

Now that we understand the fundamental architectures, we must explore how agents actually learn and evolve when interacting with each other.

Next Time, We Will Explore:

- **Self-Play Training:** "Fighting with Your Own Shadow"
  - Why compare with past versions of yourself?
  - AlphaStar's league training strategy and exploitability testing
- **Policy Gradients in Multi-Agent Settings:** The "Subtle Art" of Coordination
  - Why $\nabla J_i(\theta_1, ..., \theta_n)$ depends on everyone else's parameters
  - Nash-gradient vs Independent gradient: the convergence "mystery"
- **Mean Field Theory:** From Individual Focus to Population Trends
  - Mathematical abstraction for large-scale multi-agent systems

Plus: Live demonstration of multi-agent learning dynamics in action!

Questions?