Algorithm Comparison Report: 4-Dimensional Observation Space

Experimental Setup

Observation Space: Each agent observes only $[own_x, own_y, goal_x, goal_y]$ (4 dimensions)

- **Removed**: Other agent's position information
- **Implication**: Agents must coordinate without direct visibility of each other

Environment: 5×5 Meeting Gridworld

- Agents: 2 agents
- Task: Both agents must reach the goal simultaneously
- Reward: +10.0 for success, -0.01 per step
- Max Steps: 50 per episode

Training Configuration:

- Episodes: 1000 per seed
- Seeds: 5 seeds (2025-2029)
- Evaluation: Final 100 episodes (training performance)

Algorithm Descriptions

1. Independent Q-Learning (IQL)

Approach: Each agent learns independently, treating other agents as part of the environment.

Architecture:

- Separate Q-network per agent
- Separate optimizer per agent
- Separate replay buffer per agent
- No explicit coordination mechanism

Key Characteristics:

- **Decentralized**: Each agent learns its own Q-function $Q_i(o_i, a_i)$
- **Simple**: Straightforward extension of single-agent Q-learning
- **Non-stationary**: Environment changes as other agents learn
- **Strengths**: Simple, scalable, works well when coordination is implicit
- **Weaknesses**: Struggles with non-stationarity, no explicit coordination

2. Parameter-Shared DQN (PS-DQN)

Approach: All agents share a single Q-network, enabling parameter sharing and sample efficiency.

Architecture:

- Single shared Q-network for all agents
- Single shared optimizer
- Single shared replay buffer
- Parameter sharing across agents

Key Characteristics:

- **Shared Learning**: All agents learn from all agent experiences
- **Sample Efficient**: Each timestep contributes n_agents samples
- **Homogeneous:** Assumes agents are similar/homogeneous
- **Strengths:** Sample efficient, can learn faster with shared parameters
- **Weaknesses:** May constrain policy diversity, struggles when agents need different strategies

3. QMIX

Approach: Centralized training with decentralized execution, using a mixing network to combine agent Q-values.

Architecture:

- Individual agent Q-networks (decentralized execution)
- Mixing network with hypernetworks (centralized training)
- Centralized replay buffer with global state

- Monotonicity constraint ensures decentralized execution validity

Key Characteristics:

- **CTDE**: Centralized Training, Decentralized Execution
- **Explicit Coordination**: Mixing network learns how to combine agent Q-values
- **Monotonicity**: The derivative of total Q with respect to individual Q is greater than or equal to zero, ensuring greedy per-agent actions maximize total Q
- **Global State**: Mixing network uses global state during training
- **Strengths**: Explicit coordination, leverages global information during training, maintains decentralized execution
- **Weaknesses**: More complex, requires global state, computationally more expensive

---

Results Comparison

Performance Summary

Algorithm	Success Rate	Episode Length	Return
QMIX	**54.00% ± 39.92%**	31.9 ± 13.5	5.09 ± 4.13
IQL	40.00% ± 10.88%	37.7 ± 3.6	3.63 ± 1.12
PS-DQN	3.80% ± 1.60%	49.0 ± 0.5	-0.11 ± 0.17

QMIX: **54.00% ± 39.92% Success Rate**

- Success Rate: 54.00% (highest, but high variance)
- Episode Length: 31.9 steps (most efficient when successful)
- Return: 5.09 (highest average return)
- Variance: Very high (39.92% std) - indicates instability across seeds

Analysis:

- Best Performance: Highest success rate and return when it works
- High Variance: Large standard deviation suggests some seeds perform very well while others fail
- Efficiency: Shortest episode length when successful, indicating good coordination
- Why It Works: Mixing network enables explicit coordination despite partial observability

IQL: 40.00% ± 10.88% Success Rate

- Success Rate: 40.00% (solid, consistent performance)
- Episode Length: 37.7 steps (moderate efficiency)
- Return: 3.63 (moderate return)
- Variance: Low (10.88% std) - most stable across seeds

Analysis:

- Consistent: Low variance indicates reliable performance across seeds
- Moderate Performance: Good success rate, though lower than QMIX's best
- Stability: Most stable algorithm - consistent results across different seeds
- Why It Works: Independent learning allows agents to adapt to each other implicitly through the environment

PS-DQN: 3.80% ± 1.60% Success Rate

- Success Rate: 3.80% (very poor performance)
- Episode Length: 49.0 steps (near timeout - agents not learning)
- Return: -0.11 (near zero, indicating no learning)
- Variance: Very low (1.60% std) - consistently poor across all seeds

Analysis:

- Poor Performance: Near-zero success rate indicates failure to learn coordination
- No Learning: Episode length near 50-step timeout suggests agents are not learning effective policies
- Consistently Bad: Low variance means it fails across all seeds, not just some
- Why It Fails: Parameter sharing may prevent agents from learning distinct coordination strategies needed when they can't see each other

Key Findings

1. QMIX Performs Best (When It Works)
   - Highest success rate (54%) and return (5.09)
   - Mixing network enables explicit coordination despite partial observability
   - However: High variance (39.92%) indicates instability - some seeds fail completely
2. IQL is Most Reliable
   - Consistent 40% success rate across seeds (low variance: 10.88%)
   - Independent learning allows implicit coordination through environment feedback
   - Trade-off: Lower peak performance but more stable
3. PS-DQN Fails Completely
   - Only 3.80% success rate - essentially not learning
   - Parameter sharing prevents learning distinct coordination strategies
   - Problem: Without seeing other agents, shared parameters can't learn agent-specific coordination behaviors
4. Impact of Removing Other Agent Position
   - IQL: Performance dropped from ~63% (6D obs) to 40% (4D obs) - still functional
   - PS-DQN: Performance dropped from ~6% (6D obs) to 3.8% (4D obs) - already poor, now worse
   - QMIX: Performance varies widely (54% average, but high variance) - benefits from mixing network's coordination

---

Conclusions

1. QMIX shows the best potential (54% success) but needs stabilization to reduce variance
2. IQL is the most reliable (40% success, low variance) - best choice for consistent performance
3. PS-DQN fails (3.8% success) - parameter sharing is detrimental when agents can't see each other

Removing other agent positions makes coordination harder, but:

- IQL adapts through environment feedback (implicit coordination)
- QMIX can learn explicit coordination through mixing network
- PS-DQN fails because parameter sharing prevents learning distinct coordination strategies

The results demonstrate that explicit coordination mechanisms (QMIX) or independent learning (IQL) work better than parameter sharing (PS-DQN) when agents have limited observability.