Chapter Outline

RL = Reinforcement Learning

Multi-Agent Reinforcement Learning and Scaling

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.5.1 Markov Games and Learning Architectures

Textbook: Sec. 10.1.3.5 · Frameworks and Problem Description

10.5.2 MADDPG Data Flow and Updates

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Textbook: Sec. 10.1.3.5 · Network Updates and Training

10.5.3 Large-Scale Swarm Expansion

Textbook: Sec. 10.1.3.6 · Challenges and Optimization

10.5.4 Method Selection and Validation

Textbook: Secs. 10.1.3.5–10.1.3.6 · Method Selection

10.5.1 Markov Games and Learning Architectures: Tasks and Rewards

Multiple Learners Change a Shared Environment

Multi-agent reinforcement learning (MARL): agents interact in a shared environment while improving their own policies.

Running task: two aircraft maintain spacing along an inspection route. Each commands a velocity; its neighbor's action changes their relative distance.

Cooperation and competition: a shared task reward expresses cooperation; conflicting rewards express competition. Both can coexist.

Prerequisites: returns and value functions in slide set 10.3; actor policy networks and critic value networks in slide set 10.4.

10.5.1 Markov Games and Learning Architectures: Modeling

Markov Games for Multi-Agent Decisions

$$\mathcal{G} = \langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^{N}, \{\mathcal{O}_i\}_{i=1}^{N}, \mathcal{P}, \{r_i\}_{i=1}^{N}, \gamma \rangle \tag{10.39}$$

Symbols and Meaning	Inspection Task
$s \in \mathcal{S}, \quad o_i \in \mathcal{O}_i$	All positions/task status; local/neighbor observations
$\mathbf{a} = (a_1, \ldots, a_N)$	All aircraft execute velocity commands simultaneously
$\mathcal{P}(s' \mid s, \mathbf{a}),\ r_i$	Joint actions drive transitions; immediate rewards per agent
$\gamma \in (0,1), \quad o_i = O_i(s)$	Discount; specify observation rules or probabilities

Joint observations aid training but may omit true state. MDP = Markov decision process.

10.5.1 Markov Games and Learning Architectures: A Worked Transition

Joint Actions Determine the Next Spacing Error

Assumptions: 1D point masses; 1 s steps; desired gap 4 m; velocities act immediately.

$$p_i^+ = p_i + v_i \Delta t, \quad e^+ = p_2^+ - p_1^+ - 4, \quad r = -(e^+)^2$$

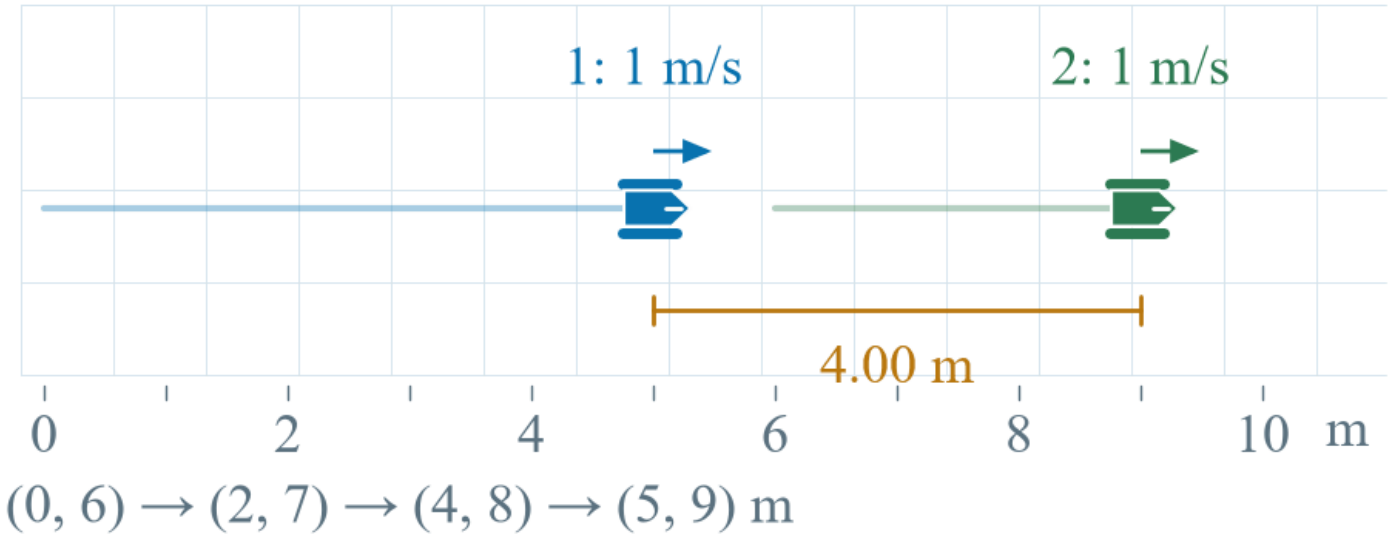
Step	Start Positions / m	Velocities / (m/s)	Next Positions / m	Reward
1	(0, 6)	(2, 1)	(2, 7)	-1
2	(2, 7)	(2, 1)	(4, 8)	0
3	(4, 8)	(1, 1)	(5, 9)	0

Step 1: error 2 m to 1 m. Match speeds at 4 m to avoid overshoot from unequal speeds.

10.5.1 Markov Games and Learning Architectures: Joint Actions

Relative Velocity Determines Spacing

Both agents move



t = 3.00 s

Desired gap: 4 m

p = (5.00, 9.00)

r[3] = 0

Equal speeds keep the gap

Figure: two aircraft in 1D; trajectories, positions, spacing, and step-end rewards share one calculation

Spacing falls from 6 m to 4 m; both then match speeds and keep moving on the fixed grid.

10.5.1 Markov Games and Learning Architectures: Frameworks

Independent Learning, Communication, and CTDE

Approach	Training Inputs and Updates	Execution Inputs
Independent learning	Each agent updates from its own data and reward	Local observations; interactions are not explicitly modeled
Communication-based learning	Learn to use or generate messages	Local observations and permitted neighbor messages
CTDE	Joint information supports a critic or joint loss	Each policy uses only information available at deployment

CTDE = centralized training with decentralized execution. Communication can accompany independent learning or CTDE.

10.5.1 Markov Games and Learning Architectures: Non-Stationarity

Changing Neighbor Policies Shift the Learning Target

Example: two aircraft share one passage. Agent 1 receives -10 if both advance and +2 if only it advances; p is the neighbor's advance probability.

$$\mathbb{E}[r_1 \mid a_1 = \text{forward}] = -10p + 2(1-p) = 2 - 12p$$

Neighbor Advance p	Expected Local Reward	Interpretation
0.2	$2 - 12 \cdot 0.2 = -0.4$	The neighbor usually yields
0.8	$2 - 12 \cdot 0.8 = -7.6$	Changed policy; outdated estimate

Joint information helps explain interactions; it guarantees neither convergence nor stationarity.

10.5.1 Markov Games and Learning Architectures: Policy Changes

The Same Action Can Have a Different Outcome

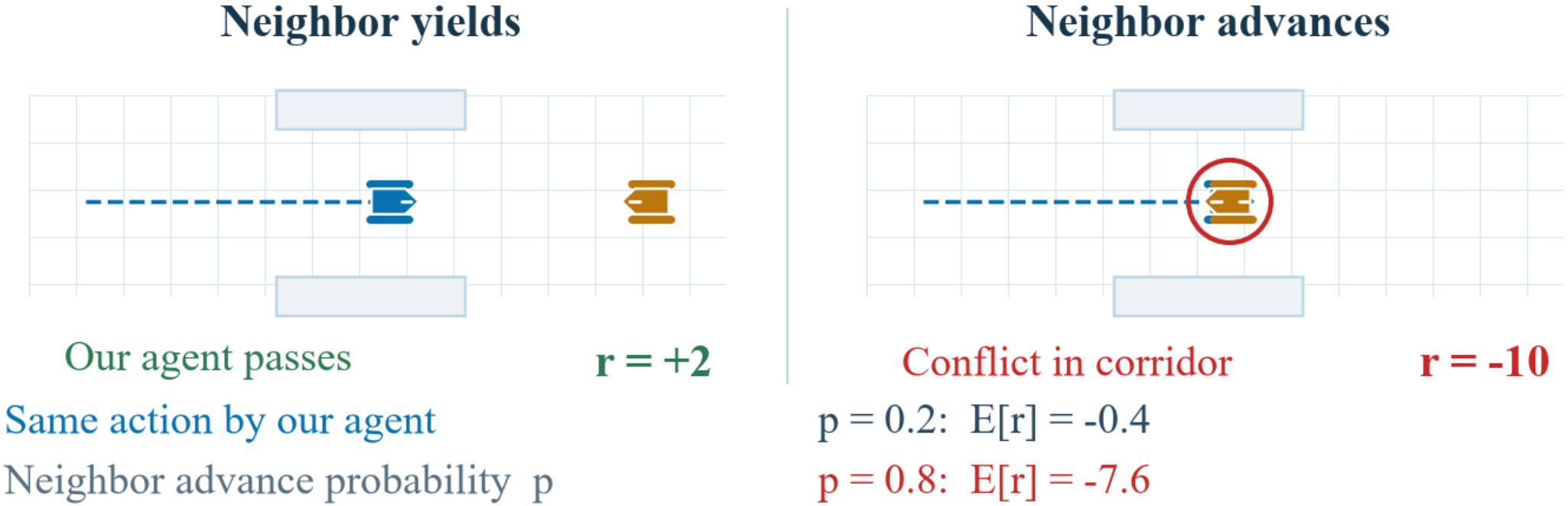


Figure: blue local agent, ocher neighbor; same local initial state and action, different neighbor actions

As the neighbor enters first more often, the local advance action has a lower expected reward.

Chapter Outline

RL = Reinforcement Learning

Multi-Agent Reinforcement Learning and Scaling

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.5.1 Markov Games and Learning Architectures

Textbook: Sec. 10.1.3.5 · Frameworks and Problem Description

10.5.2 MADDPG Data Flow and Updates

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Textbook: Sec. 10.1.3.5 · Network Updates and Training

10.5.3 Large-Scale Swarm Expansion

Textbook: Sec. 10.1.3.6 · Challenges and Optimization

10.5.4 Method Selection and Validation

Textbook: Secs. 10.1.3.5–10.1.3.6 · Method Selection

10.5.2 MADDPG Data Flow and Updates: Multiple Actors and Critics

Deep Deterministic Policy Gradient (DDPG)

TD: temporal-difference

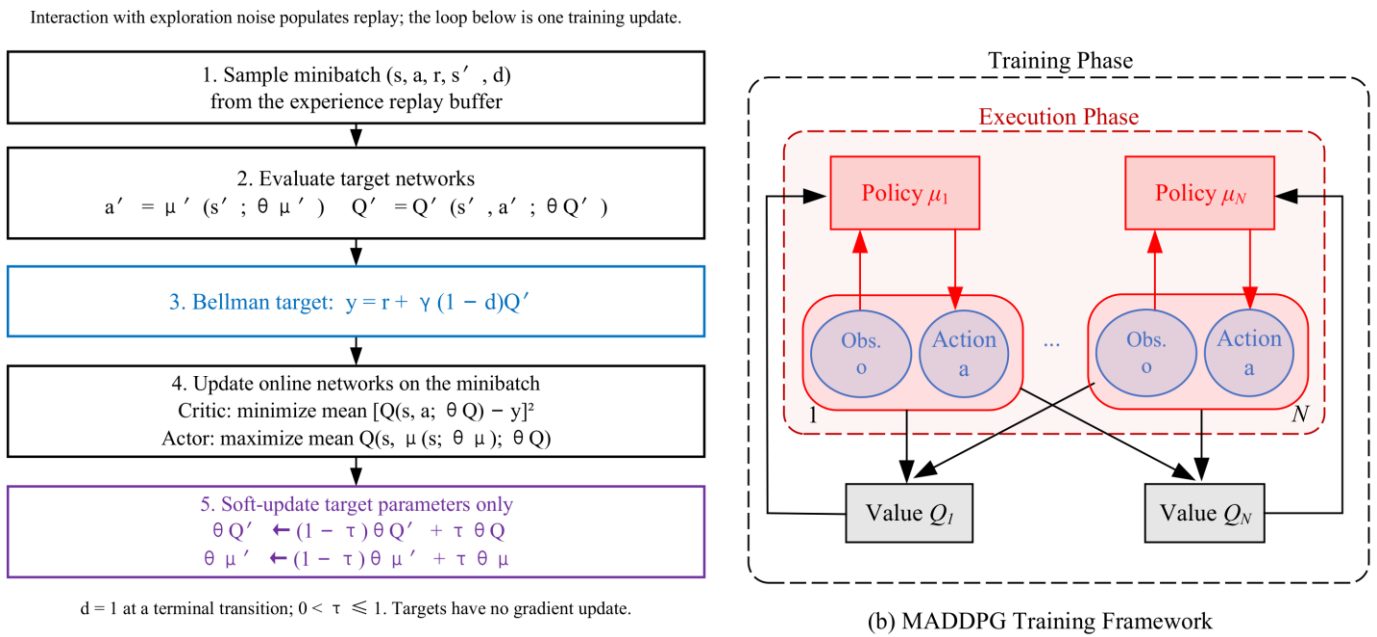
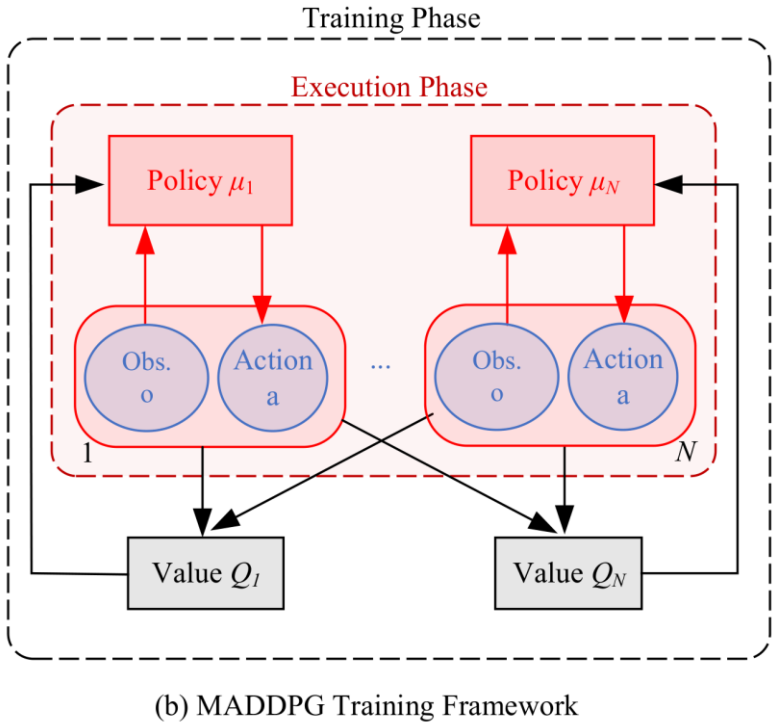


Figure 10.7 DDPG and MADDPG frameworks

Each target softly follows its own online network (Eq. 10.37); the two updates are separate.

10.5.2 MADDPG Data Flow and Updates: Textbook Detail

Two Information Paths in MADDPG



(b) MADDPG Training Framework

Figure 10.7(b) MADDPG training framework

Inside the red boundary: local observations enter the local policy, which outputs the local action.

Value networks: joint observations and actions are used to evaluate each agent's return.

Outer feedback: policies are updated during training; deployment does not require a centralized critic.

10.5.2 MADDPG Data Flow and Updates: CTDE Information Boundaries

Joint Evaluation in Training, Local Decisions in Execution

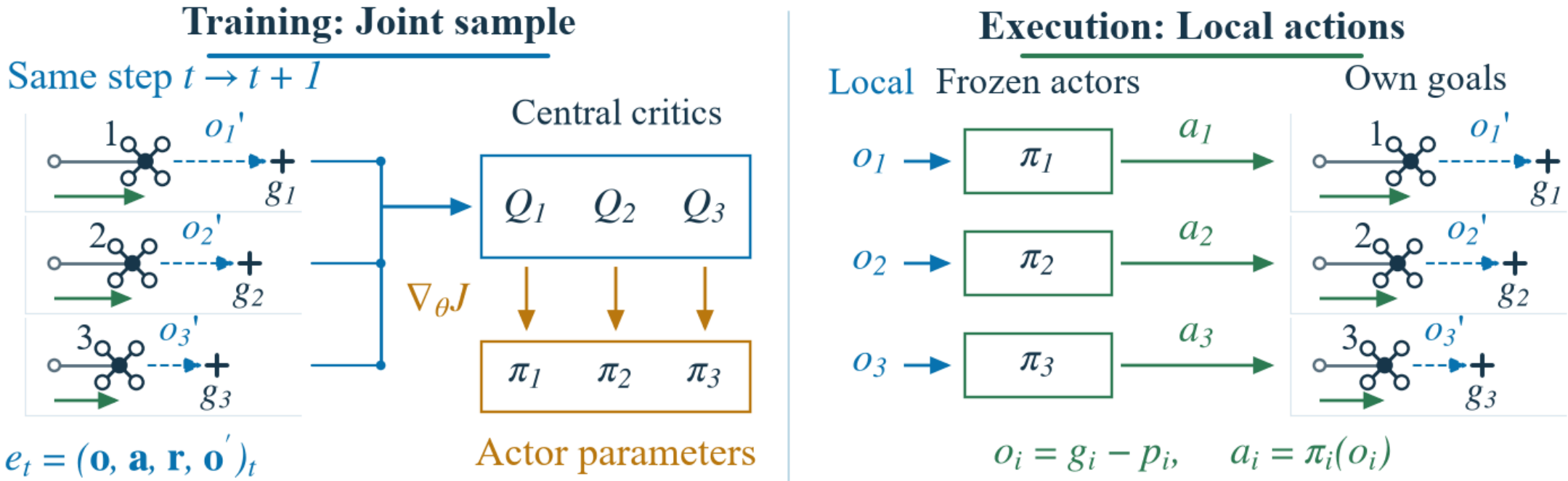


Figure: centralized training with decentralized execution (CTDE)

Critics train on joint data; actors use only local observations, without inter-agent messages.

10.5.2 MADDPG Data Flow and Updates: Experience Replay

A Joint Replay Sample Comes from One Interaction

$$(\mathbf{x}, \mathbf{a}, \mathbf{r}, \mathbf{x}') \tag{10.45}$$

$$\mathbf{x} = (o_1, \ldots, o_N), \quad \mathbf{a} = (a_1, \ldots, a_N), \quad \mathbf{r} = (r_1, \ldots, r_N)$$

Alignment: the same environment, step, and node IDs. Next observations must match the actions actually executed.

Sampling: draw each joint experience as a whole. Each actor reads only its own observation from that sample.

Implementation: record termination flags and valid nodes; store training-only global state separately. Incomplete observations may need history or recurrent networks.

10.5.2 MADDPG Data Flow and Updates: Target Values

The Target Critic Estimates the Next Value

$$y_i = r_i + \gamma Q_i'(s', a_1', \ldots, a_N') \tag{10.40}$$

$$a_j' = \mu_j'(o_j')$$

Notation: this critic uses global state. Target actors generate next actions; the target critic estimates returns. No gradients pass through the target calculation.

Example: immediate reward 1, discount 0.9, and target value 2 give $1 + 0.9 \times 2 = 2.8$.

Termination: Equation 10.40 applies to nonterminal samples. At true termination, the target is the immediate reward; handle time truncation according to the task.

10.5.2 MADDPG Data Flow and Updates: Critic Loss

Critic Updates from TD Error

$$\mathcal{L}_i(\theta_i^Q) = \mathbb{E}_{(s,\mathbf{a},r_i,s')}\left[(y_i - Q_i(s,\mathbf{a};\theta_i^Q))^2\right] \tag{10.41}$$

$$\delta = 2.8 - 1.5 = 1.3, \quad \ell = 1.3^2 = 1.69$$

$$Q = \theta, \quad \theta^+ = \theta - 0.1 \cdot 2(\theta - y) = 1.76$$

Temporal-difference (TD) error is target minus estimate. Here the critic is one parameter with a fixed target. Real training averages a minibatch and applies gradient descent to network parameters.

10.5.2 MADDPG Data Flow and Updates: Policy Gradients

The Actor Updates to Increase Its Agent's Value

$$J_i(\theta_i^\mu) = \mathbb{E}_{(s,o_i)\sim\mathcal{D}}[Q_i(s, a_1, \ldots, a_i, \ldots, a_N)] \tag{10.42}$$

$$\nabla_{\theta_i^\mu} J_i = \mathbb{E}[\nabla_{a_i} Q_i(s, \mathbf{a}) \nabla_{\theta_i^\mu} \mu_i(o_i; \theta_i^\mu)] \tag{10.43}$$

$a_i = \mu_i(o_i; \theta_i^\mu)$. Hold other actions fixed when updating this actor; evaluate the gradient at its current policy output.

Actor training still uses gradients from the joint critic. Execution keeps only the local actor. An inaccurate critic can also misdirect policy improvement.

10.5.2 MADDPG Data Flow and Updates: A Worked Gradient Example

Four Actor Updates with a Fixed Analytic Critic

$$Q_1 = -(a_1 - a_2)^2, \quad a_2 = 2, \quad a_1 = \theta o_1, \quad o_1 = 1$$

$$\theta_{k+1} = \theta_k + 0.1[-2(\theta_k - 2)]$$

Update	Local Action	Analytic Value	Error from Optimal Action
0	0.0000	-4.0000	2.0000
1	0.4000	-2.5600	1.6000
2	0.7200	-1.6384	1.2800
3	0.9760	-1.0486	1.0240
4	1.1808	-0.6711	0.8192

Fixed critic and neighbor action; no action saturation. Values are computed from the recurrence.

10.5.2 MADDPG Data Flow and Updates: Soft Updates

Target Networks Slowly Follow Online Networks

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta' \tag{10.44}$$

$$\tau = 0.01, \quad \theta = 2, \quad \theta' = 1 \quad \Rightarrow \quad \theta'^{+} = 1.01$$

Initialization: the online actor and critic each have a target copy, usually initialized by copying the corresponding online parameters.

Trade-off: a small update coefficient smooths the target but increases lag. The coefficient and update frequency together set the tracking speed.

10.5.2 MADDPG Data Flow and Updates: Training Procedure

MADDPG Training and Evaluation

Stage	Inputs, Computation, and Outputs
1 Initialize and interact	Copy target networks initially; add exploration to local actor outputs to form joint actions
2 Store and sample	Store synchronized experience in replay; sample whole joint transitions at random
3 Critic and actor updates	Compute targets and fit values first; then update each local policy using action gradients
4 Soft update and evaluate	Target copies follow slowly; freeze actors and evaluate separate task episodes

Train actors on execution inputs; deleting information at deployment is insufficient.

Chapter Outline

RL = Reinforcement Learning

Multi-Agent Reinforcement Learning and Scaling

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.5.1 Markov Games and Learning Architectures

Textbook: Sec. 10.1.3.5 · Frameworks and Problem Description

10.5.2 MADDPG Data Flow and Updates

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Textbook: Sec. 10.1.3.5 · Network Updates and Training

10.5.3 Large-Scale Swarm Expansion

Textbook: Sec. 10.1.3.6 · Challenges and Optimization

10.5.4 Method Selection and Validation

Textbook: Secs. 10.1.3.5–10.1.3.6 · Method Selection

10.5.3 Large-Scale Swarm Expansion: Bottlenecks

How Costs Grow with Agent Count

$$d_Q = N(d_o + d_a) = 15N, \quad d_o = 12, \quad d_a = 3$$

Bottleneck	Growth or Mechanism	Design Options
Joint critic input	10 agents: 150 dimensions; 50 agents: 750	Shared encoding; neighborhood aggregation
Separate critics	One network per agent multiplies parameters and updates	Share by role; compress the joint representation
Policy drift and correlation	Simultaneous updates; strongly correlated group experience	Assess old-policy effects; diversify scenarios

Enumerating joint discrete actions gives $|\mathcal{A}| = m^N$, a combinatorial explosion.

10.5.3 Large-Scale Swarm Expansion: Parameter Sharing

Shared Parameters Still Allow Different Actions

$$a_i = \mu(o_i, z_i; \theta), \quad i = 1, \ldots, N$$

What is shared: homogeneous agents share actor parameters. Critic sharing depends on rewards, roles, and input structure.

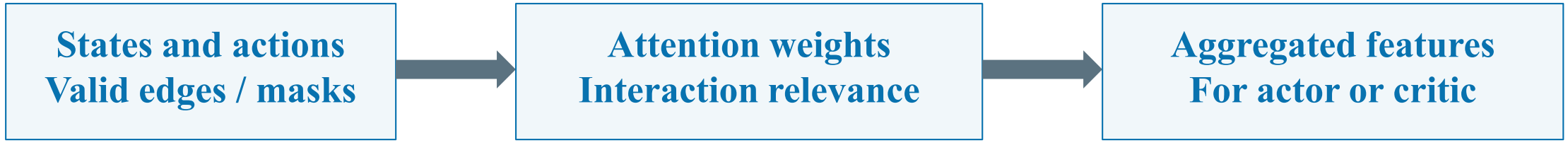
Individual differences: distinct observations and role encodings can still produce different actions.

Example: 100,000 parameters per actor means 5,000,000 for 50 copies. Shared weights need 100,000; inference still processes 50 inputs.

Conditions: compatible observation and action meanings. For heterogeneous platforms, align interfaces first or use separate networks by role.

10.5.3 Large-Scale Swarm Expansion: Attention and Graphs

Attention and GNNs Aggregate Neighbor Information



Attention: weight neighbor features for the task. Weights reflect associations within the model, not direct evidence of causation.

Graph neural networks (GNNs): aggregate node features along edges and allow varying neighbor counts. Results depend on edges, features, and the training distribution.

Deployment: a training critic may use a joint graph. An actor using graph messages must meet actual communication and timing constraints.

10.5.3 Large-Scale Swarm Expansion: A Graph Aggregation Example

A Worked Neighborhood Aggregation

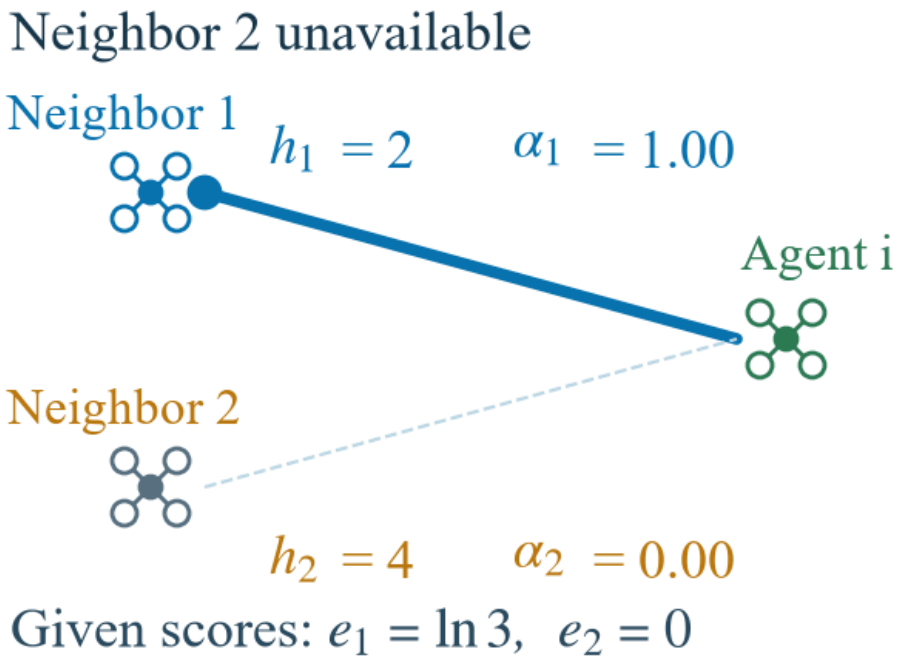
$$h_1 = 2, \quad h_2 = 4, \quad m_{\text{mean}} = (2 + 4)/2 = 3$$

$$\alpha_1 = 0.75, \quad \alpha_2 = 0.25, \quad m = 0.75 \cdot 2 + 0.25 \cdot 4 = 2.5$$

Message Availability	Valid Weights	Aggregation Output
Both neighbors valid	0.75, 0.25	2.5
Second neighbor missing	Renormalized to 1, 0	2.0
No valid neighbors	Local state + missing flag	Do not pass old data as new

10.5.3 Large-Scale Swarm Expansion: Neighbor Attention

Renormalize over Valid Neighbors



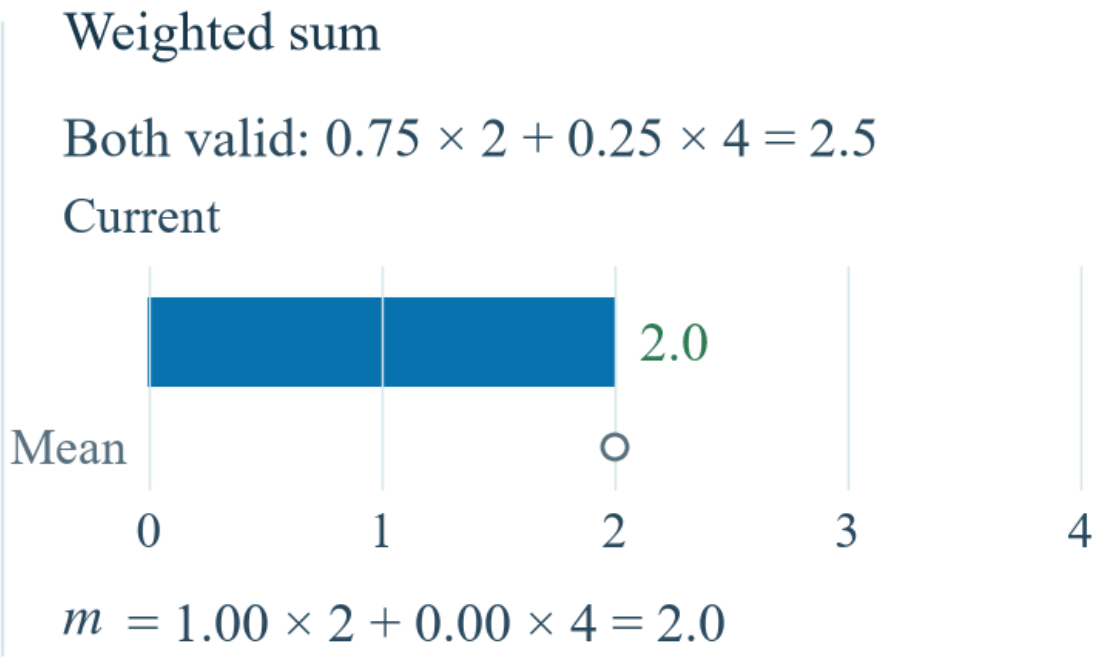


Figure: exponentiated scores are normalized into weights; dots on edges show valid message transfers

Given scores yield output 2.5 with both messages, or 2 with only the first valid message.

10.5.3 Large-Scale Swarm Expansion: Value Decomposition Network (VDN)

Team Values and Local Greedy Choices

$$Q_{\text{tot}} = \sum_{i=1}^{N} Q_i(o_i, a_i) \quad \text{(VDN)}$$

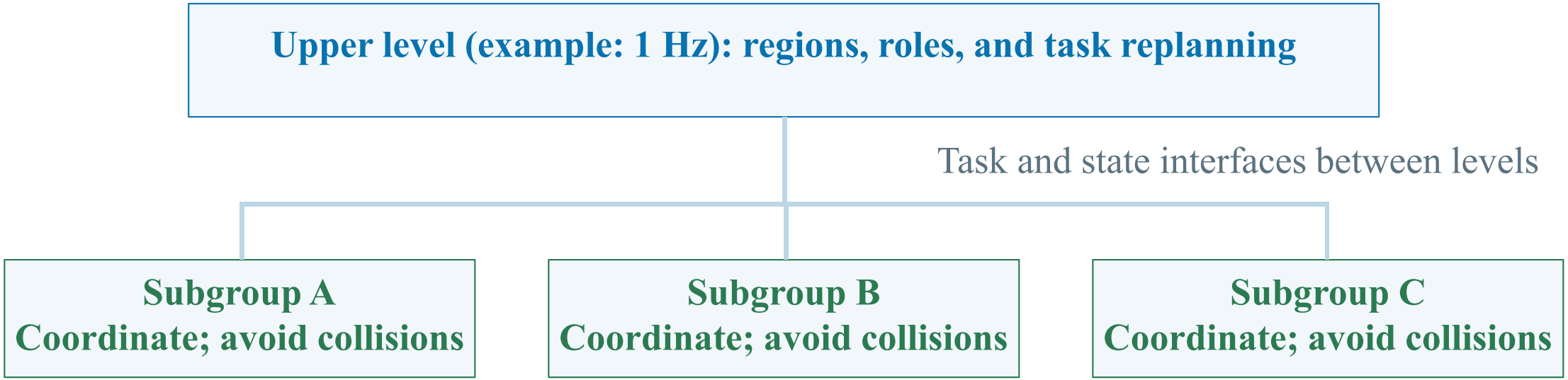
$$Q_{\text{tot}} = f_s(Q_1, \ldots, Q_N), \quad \frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0 \quad \text{(QMIX)}$$

Example Values	Agent 2 Chooses A: 4	Agent 2 Chooses B: 2
Agent 1 chooses A: 1	5	3
Agent 1 chooses B: 3	7 (locally greedy)	5

Shared rewards and discrete actions; nonmonotonic team values may be unrepresentable.

10.5.3 Large-Scale Swarm Expansion: Hierarchical Design

Task Allocation and Fast Control at Different Levels



Lower level (example: 20 Hz): generate actions from local observations. Define goal validity, resource constraints, and failure feedback; upper-level task changes must preserve lower-level safety constraints.

Chapter Outline

RL = Reinforcement Learning

Multi-Agent Reinforcement Learning and Scaling

Chapter 10 Swarm Collaboration and Game-Theoretic Confrontation

10.1 Swarm Fundamentals and Architectures

10.2 Consensus and Formation Control

10.3 Game Theory and RL Fundamentals

10.4 Single-Agent RL Algorithms

10.5 Multi-Agent RL and Scaling

10.6 Technologies and Challenges

10.7 RflySim Framework and Interfaces

10.8 Experiments and Validation

10.9 Exercises and Classroom Discussion

4 Units in This Section

10.5.1 Markov Games and Learning Architectures

Textbook: Sec. 10.1.3.5 · Frameworks and Problem Description

10.5.2 MADDPG Data Flow and Updates

Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Textbook: Sec. 10.1.3.5 · Network Updates and Training

10.5.3 Large-Scale Swarm Expansion

Textbook: Sec. 10.1.3.6 · Challenges and Optimization

10.5.4 Method Selection and Validation

Textbook: Secs. 10.1.3.5–10.1.3.6 · Method Selection

10.5.4 Method Selection and Validation: Selecting a Method

Choose Methods for the Task and Resources

Task Conditions	Candidate Structure	What to Validate
Continuous actions; strong interactions	MADDPG with joint critics	Sample efficiency, value estimates, and policy drift
Homogeneous agents; local interactions	Shared actor + sparse GNN	Role encoding, changing node counts, missing messages
Shared rewards; discrete actions	VDN/QMIX with parameter sharing	Value-factorization assumptions; joint behavior representation
Multiple subtasks and timescales	Hierarchical groups, optionally with graph aggregation	Cross-level interfaces, resource conflicts, link-loss fallback

10 and 50 agents are teaching scale examples; choose by input dimensions and resource limits.

10.5.4 Method Selection and Validation: Evaluation and Extensions

Policy Evaluation and Auxiliary Models

Metrics: freeze the policy, then test task success, minimum spacing over the full episode, energy use, and inference deadlines.

Generalization: vary agent counts, initial conditions, message loss, and localization errors; record performance drops and failure causes.

Large language model (LLM) assistance: task descriptions, data augmentation, and high-level analysis. Outputs remain subject to resource, safety, and timing limits.

Platform: MARLExps (Full Edition or higher) uses single-agent proximal policy optimization (PPO) and parallel environments. Cooperative training needs a shared environment and multi-agent learners.

Exercises and Discussion

Which Inputs Can an Onboard Policy Use?

Discussion Question	Reference Answer
The critic uses all ground-truth data; the actor uses local observations only. Is this CTDE?	Yes. Joint information supports training evaluation, while deployed actor inputs stay consistent.
An actor trains on all ground-truth data. Can it run directly if those inputs are removed at deployment?	Effective execution is not guaranteed. Redesign inputs, then retrain and evaluate.
What should change first when scaling from 8 to 60 agents?	Measure input dimensions, memory, communication, and latency first; then choose sharing, sparse graphs, or hierarchy.

Section Summary

Joint Learning and Local Execution

Model: joint actions drive transitions; rewards express cooperation or competition. Local observations need not form a complete state.

MADDPG: critics read joint information; actors produce local actions. Replay, target networks, and soft updates complete the training loop.

Scaling: sharing reduces parameters; GNNs organize neighbors; value decomposition supports cooperation; hierarchy separates timescales.

Common pitfall: CTDE does not mean centralized execution, nor does it ensure safety, convergence, or generalization across swarm sizes.