Chapter Outline

RL = reinforcement learning

Game Theory and Reinforcement Learning Fundamentals

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

10.3.1 Intelligent Control Framework for Swarms

Textbook: Sec. 10.1.3.1

10.3.2 Reinforcement Learning and Markov Decision Processes

Textbook: Secs. 10.1.3.2–10.1.3.3

10.3.3 Returns and Value Functions

Textbook: Sec. 10.1.3.3, items 3–4

10.3.4 Bellman Equations and Policy Optimization

Textbook: Sec. 10.1.3.3, item 5

10.3.1 Intelligent Control Framework for Swarms: Tasks and Methods

Game Theory and Policy Learning

- **Game theory:** one decision maker's payoff depends on others' choices. Interactions may be cooperative, competitive, or both.
- **Reinforcement learning (RL):** an agent uses rewards from interaction to improve its action-selection policy.
- **Scope:** fix the environment rules and study how one decision maker compares the long-term effects of current actions.

Define the payoff for each decision maker before optimizing a policy.

10.3.1 Intelligent Control Framework for Swarms: Fig. 10.4

Intelligent Policies Between Tasks and Control

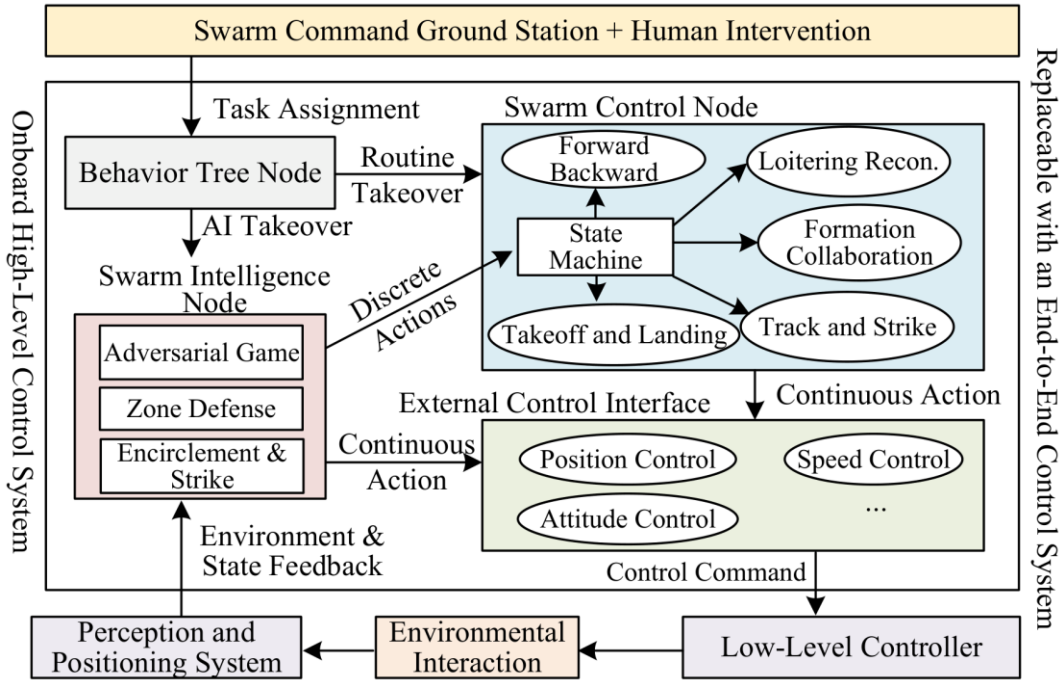


Fig. 10.4 Typical intelligent swarm control framework

- **Task layer:** a behavior tree activates nodes when conditions hold.
- **Decision layer:** intelligent nodes choose actions or references.
- **Control layer:** track references and feed back the state.

10.3.1 Intelligent Control Framework for Swarms: Task Branches

Behavior Trees Activate Intelligent Nodes

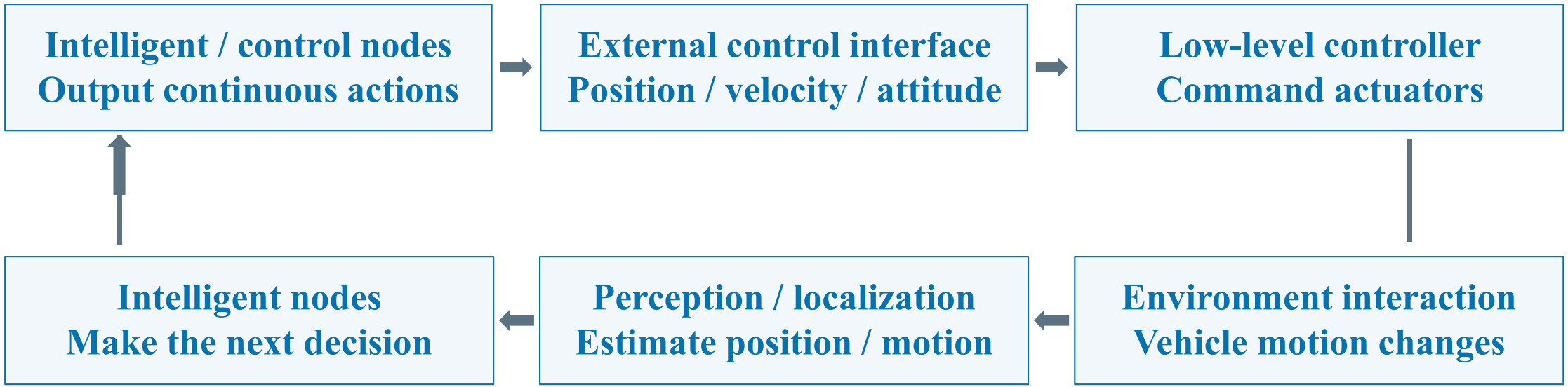
Ground station / operator
Assign tasks
→
Behavior tree
Select by conditions
→
Control / intelligent nodes
Generate behavior

Tree Branch	Decision	Execution
Conventional control	Select a predefined control behavior	Control nodes generate motion references
Intelligent control	A learned policy outputs discrete actions	A state machine checks transitions and activates behaviors

A state machine can organize takeoff and landing, forward and backward motion, loitering for reconnaissance, formation collaboration, and target tracking.

10.3.1 Intelligent Control Framework for Swarms: Feedback

Motion References in the Feedback Loop



Position, velocity, and attitude references use different control modes. Feedback lets current motion affect the next decision.

10.3.1 Intelligent Control Framework for Swarms: Action Execution

Discrete Behaviors and Continuous Control

Intelligent policy
Select behavior / reference
→
State machine / interface
Check modes / set limits
→
Low-level controller
Track position or velocity

Action Type	Examples	Interface Requirements
Discrete behavior	Move forward, wait, return	Allowed modes, end conditions, reference-generation rules
Continuous reference	Position, velocity, attitude	Coordinate frame, units, limits, and command period

Example: the policy selects “forward,” the interface generates a forward velocity, and the controller adjusts actuators using velocity error.

10.3.1 Intelligent Control Framework for Swarms: End-to-End Control

End-to-End Control: Benefits and Costs

Design Aspect	Modular Design	End-to-End Design
Intermediate meaning	Inspect behaviors, modes, and references at each layer	A model learns the input-to-control mapping directly
Computing and energy	Allocate a computing budget to each module	Model size, latency, and onboard power are coupled
Reliability and explanation	Trace faults to modules and interfaces	Decision explanations and fault diagnosis are harder
Task complexity	3D control, multiple actuators, dynamic environments	Still constrained by dynamics and environmental uncertainty

Chapter Outline

Game Theory and Reinforcement Learning Fundamentals

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

10.3.1 Intelligent Control Framework for Swarms

Textbook: Sec. 10.1.3.1

10.3.2 Reinforcement Learning and Markov Decision Processes

Textbook: Secs. 10.1.3.2–10.1.3.3

10.3.3 Returns and Value Functions

Textbook: Sec. 10.1.3.3, items 3–4

10.3.4 Bellman Equations and Policy Optimization

Textbook: Sec. 10.1.3.3, item 5

10.3.2 Reinforcement Learning and Markov Decision Processes

Five Elements of Reinforcement Learning

Element	Meaning	Navigation Example
State	Information for predicting transitions	Position and task variables affecting motion
Action	A choice currently allowed	Go directly to the goal or take a detour
Reward	Numerical feedback per transition	Travel costs and an arrival reward
Policy	A rule for choosing actions from states	Choose the detour at S probabilistically
Value	Expected long-term policy return	Average score obtainable from S

10.3.2 Reinforcement Learning and Markov Decision Processes

A Four-State Navigation Task



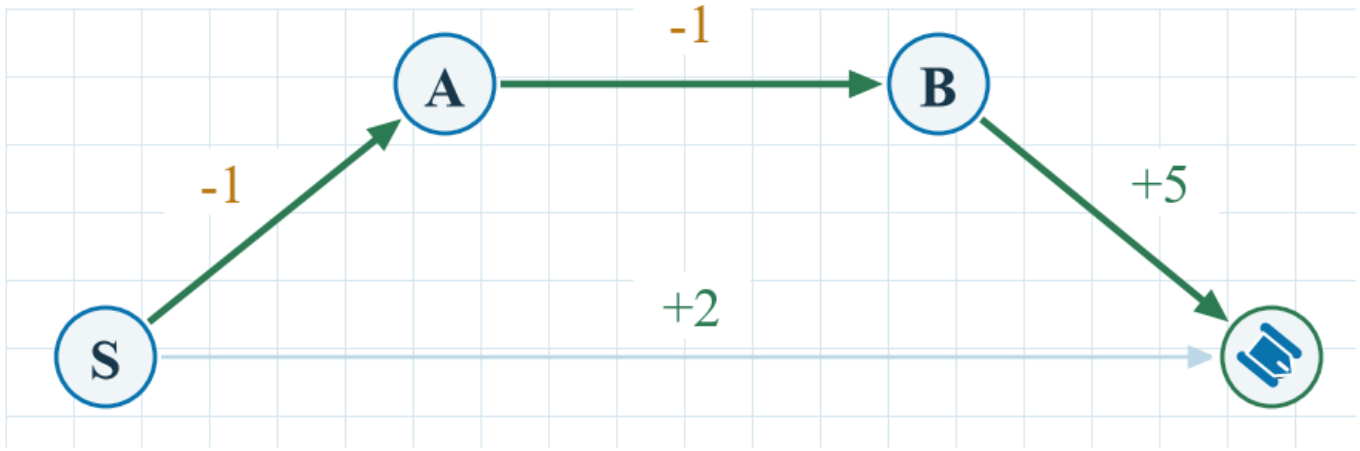
- **Assumptions:** position is the full state. Transitions are deterministic, with no obstacles, inertia, or measurement error.
- **Allowed actions:** S has a direct route or a detour. A and B can only advance. G is terminal.

Rewards are in points. Each arrow is one step. Discount factor: $\gamma = 0.9$.

10.3.2 Reinforcement Learning and Markov Decision Processes

Actions and Experience on the Same Map

One move, one sample



Each segment is one action

Immediate reward

$B \rightarrow G \quad r = +5$

Arrival ends episode

Discounted return $\gamma = 0.9$

Direct 2.00

Detour 2.15

State transitions along the direct route and detour. Route labels give each action's reward.

Choose at each state. G ends the episode. Restart at S for the next episode.

10.3.2 Reinforcement Learning and Markov Decision Processes

Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma) \tag{10.17}$$

Component	Mathematical Meaning	This Example
States / actions	$\mathcal{S}$: state set. $\mathcal{A}$: action set.	4 states. 2 actions at S.
Transition probability	P: next-state distribution for a state and action	Detour: S to A with probability 1.
Reward function	R: transition reward or its conditional mean	S to A: -1. B to G: 5 points.
Discount factor	γ: multiply once more for each step of delay	0.9 in this example

10.3.2 Reinforcement Learning and Markov Decision Processes

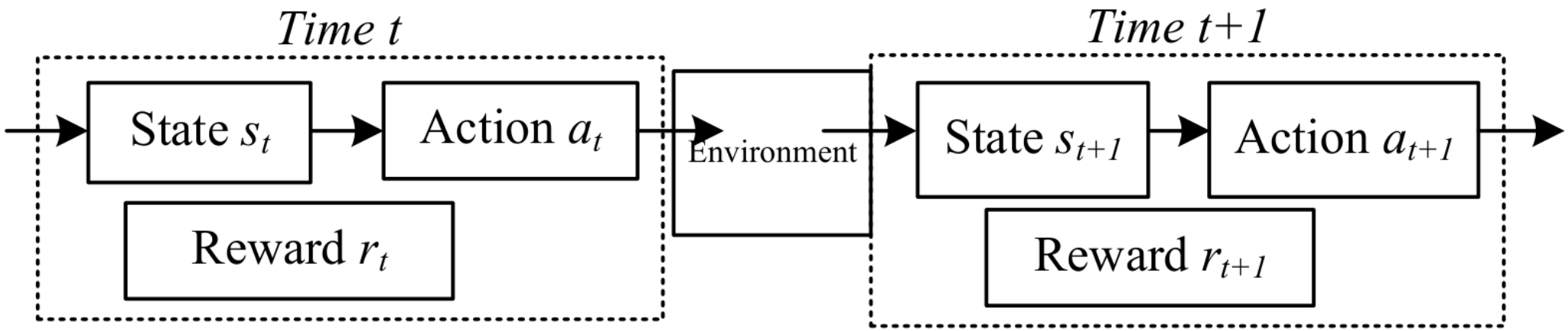
The Markov Property Needs Sufficient State Information

$$P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \dots) = P(s_{t+1} \mid s_t, a_t) \tag{10.18}$$

- **Same position, different velocities:** the next motion may differ. Position alone may omit necessary information.
- **State augmentation:** add velocity, task progress, or past observations to include more information for decisions.
- **History encoding:** recurrent neural networks (RNNs), including long short-term memory (LSTM) networks, encode sequences. Validate the resulting representation.

10.3.2 Reinforcement Learning and Markov Decision Processes

One Interaction Produces One Experience Record



(a) Markov Decision Process Model

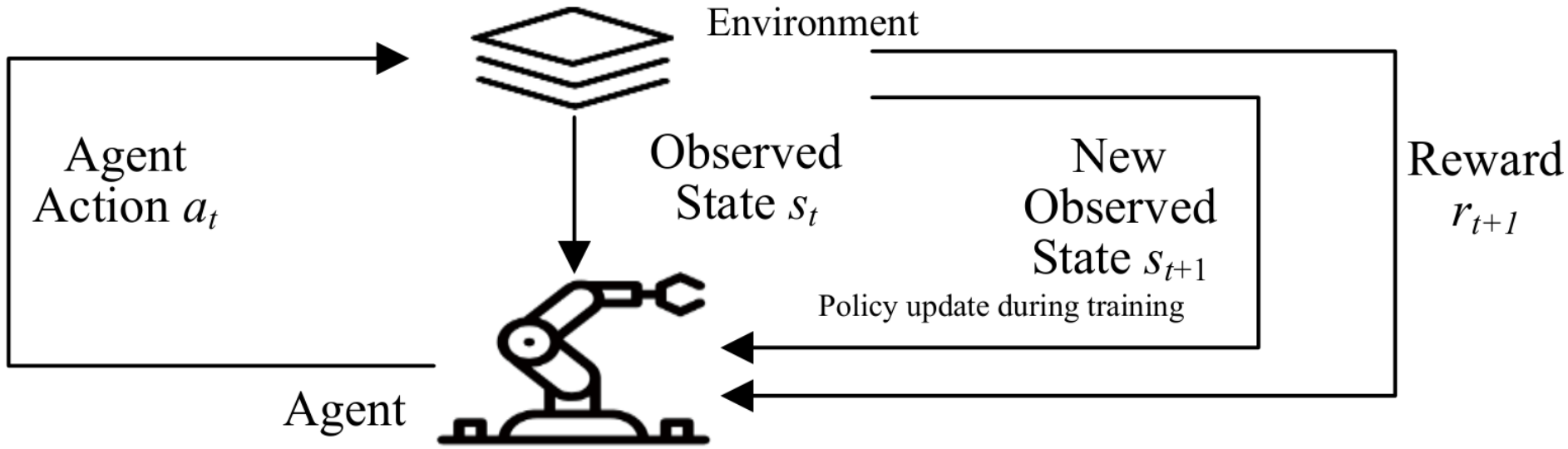
Fig. 10.5(a) Markov decision process model

$$(s_t, a_t, r_{t+1}, s_{t+1}, d_{t+1})$$

Flag 1 marks true termination. A fixed policy still uses feedback without parameter updates.

10.3.2 Reinforcement Learning and Markov Decision Processes

Training Uses Feedback to Improve the Policy



(b) Reinforcement Learning Framework

Fig. 10.5(b) Reinforcement learning framework

Feedback continues each step. Training updates parameters. A fixed policy computes actions.

Expected Values Under Random Transitions

Possible Outcome	Probability	Given Next-State Value	Step Reward
Advance from A to B	0.8	5 points	−1 point
Advance from A, stay at A	0.2	3.5 points	−1 point

$$0.8[-1 + 0.9 \times 5] + 0.2[-1 + 0.9 \times 3.5] = 3.23$$

Use the probabilities to average reward plus discounted future value: 3.23 points.
Only A's one-step transition changes. Given values are estimates. We then resume the deterministic task.

Chapter Outline

Game Theory and Reinforcement Learning Fundamentals

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

10.3.1 Intelligent Control Framework for Swarms

Textbook: Sec. 10.1.3.1

10.3.2 Reinforcement Learning and Markov Decision Processes

Textbook: Secs. 10.1.3.2–10.1.3.3

10.3.3 Returns and Value Functions

Textbook: Sec. 10.1.3.3, items 3–4

10.3.4 Bellman Equations and Policy Optimization

Textbook: Sec. 10.1.3.3, item 5

10.3.3 Returns and Value Functions: Eqs. 10.19–10.20

Immediate Reward and Discounted Return

$$R_{\text{total}} = r_{t+1} + r_{t+2} + \cdots + r_n \tag{10.19}$$

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \tag{10.20}$$

- Finite episode: end at time n. Pad later rewards with 0 for an infinite sum.

R_t is discounted return. k starts at 0, so the first reward is undiscounted.

- **Continuing task:** bounded rewards and a discount below 1 give a finite discounted return. The discount sets time weights.

10.3.3 Returns and Value Functions: Navigation Returns

Returns Along Complete Routes

Route	Reward Sequence	Steps	Undiscounted Return
S directly to G	2	1	2 points
S via A and B to G	−1, −1, 5	3	3 points

$$R_0(\gamma = 0.9) = -1 - 0.9 + 0.9^2 \times 5 = 2.15$$

$$R_0(\gamma = 0.5) = -1 - 0.5 + 0.5^2 \times 5 = -0.25$$

10.3.3 Returns and Value Functions: Eq. 10.21

Action Value for a Fixed First Action

$$Q_\mu(s_t, a_t) = \mathbb{E}_\mu\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \mid s_t, a_t\right] \tag{10.21}$$

μ is the policy. $\mathbb{E}$ is a probability-weighted average. The bar separates the given conditions.

Fix the start state S
Fix the first action
→
Then follow the policy
Allow random transitions
→
Average possible returns
Obtain the action's Q-value

$$Q_\mu(S, \text{direct}) = 2, \quad Q_\mu(S, \text{via}) = 2.15$$

direct: S to G. via: S through A and B to G. A and B have no other actions.

10.3.3 Returns and Value Functions: Eqs. 10.23 and 10.25

State Value Averages the First Action Too

$$V_\mu(s_t) = \mathbb{E}_\mu\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} \mid s_t\right] \tag{10.23}$$

$$V_\mu(s) = \mathbb{E}_{a\sim\mu(\cdot|s)}[Q_\mu(s,a)] \tag{10.25}$$

$$V_\mu(S) = 0.5 \times 2 + 0.5 \times 2.15 = 2.075$$

Stochastic policy: select actions by probability at a state. Deterministic policy: always select the same action at that state.

10.3.3 Returns and Value Functions: Eq. 10.22

The Bellman Relation Separates the First Step

$$R_t = r_{t+1} + \gamma R_{t+1}$$

s', a' are the next state and action. $r = r_{t+1}$ is the reward for this transition.

$$Q_\mu(s,a) = \mathbb{E}_{r,s'|s,a}\left[r + \gamma \mathbb{E}_{a' \sim \mu(\cdot|s')} Q_\mu(s', a')\right] \tag{10.22}$$

- **Outer expectation:** average the environment's possible next states and rewards.
- **Inner expectation:** average the actions the policy may choose at the next state. Continue under the same policy.

10.3.3 Returns and Value Functions: Eq. 10.24

State Values Computed Backward from the Goal

$$V_\mu(s) = \mathbb{E}_{a\sim\mu(\cdot|s)}\mathbb{E}_{r,s'|s,a}[r + \gamma V_\mu(s')] \tag{10.24}$$

State	Calculation Under the Detour Policy	Value
G	No future rewards after termination	0 points
B	$5 + 0.9 \times 0$	5 points
A	$-1 + 0.9 \times 5$	3.5 points
S	$-1 + 0.9 \times 3.5$	2.15 points

10.3.3 Returns and Value Functions: Modeling Choices

Reward Design Changes Optimal Behavior

- **Arrival reward:** defines the value of completing the task. Sparse rewards may leave exploration without positive feedback for a long time.
- **Running costs:** time, energy use, and boundary violations affect policy trade-offs. Their weights set priorities.
- **Termination and truncation:** future value is 0 after true termination. A sampling cutoff retains future value according to the original task.
- **Finite-time task:** if the deadline itself ends the task, remaining time belongs in the decision state.

Chapter Outline

Game Theory and Reinforcement Learning Fundamentals

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

10.3.1 Intelligent Control Framework for Swarms

Textbook: Sec. 10.1.3.1

10.3.2 Reinforcement Learning and Markov Decision Processes

Textbook: Secs. 10.1.3.2–10.1.3.3

10.3.3 Returns and Value Functions

Textbook: Sec. 10.1.3.3, items 3–4

10.3.4 Bellman Equations and Policy Optimization

Textbook: Sec. 10.1.3.3, item 5

10.3.4 Bellman Equations and Policy Optimization: Eqs. 10.26 and 10.29

Optimal Values and Optimal Actions

$$Q^*(s,a) = \max_{\mu} Q_{\mu}(s,a), \qquad V^*(s) = \max_{\mu} V_{\mu}(s) \tag{10.26}$$

$$\mu^*(s) = \operatorname*{argmax}_{a} Q^*(s,a) \tag{10.29}$$

Operation	Example Result	Meaning
Maximum: max	2.15 points	Optimal expected return for a state or action
Best action: arg max	via: take the detour	The action actually selected

10.3.4 Bellman Equations and Policy Optimization: Eq. 10.27

Optimal Q Uses the Best Next Action

$$Q^*(s,a) = \mathbb{E}_{r,s'|s,a}\left[r + \gamma \max_{a'} Q^*(s',a')\right] \tag{10.27}$$

$$Q^*(S, \text{via}) = -1 + 0.9 \times 3.5 = 2.15$$

- **Controllable choice:** select an action at the next state. The maximum represents the optimal decision.
- **Random outcome:** average transitions and rewards by their distribution. Choosing only the luckiest outcome is invalid.

10.3.4 Bellman Equations and Policy Optimization: Eq. 10.28

Optimal V Compares All Current Actions

$$V^*(s) = \max_a \mathbb{E}_{r,s'|s,a}\left[r + \gamma V^*(s')\right] \tag{10.28}$$

$$V^*(S) = \max\{2 + 0.9 \times 0,\ -1 + 0.9 \times 3.5\} = 2.15$$

Choose an action before the random outcome occurs. Compute the expectation for each action, then compare values.

Long-term value puts current costs and future gains on the same scale.

10.3.4 Bellman Equations and Policy Optimization: Value Iteration

Value Iteration Propagates the Arrival Reward

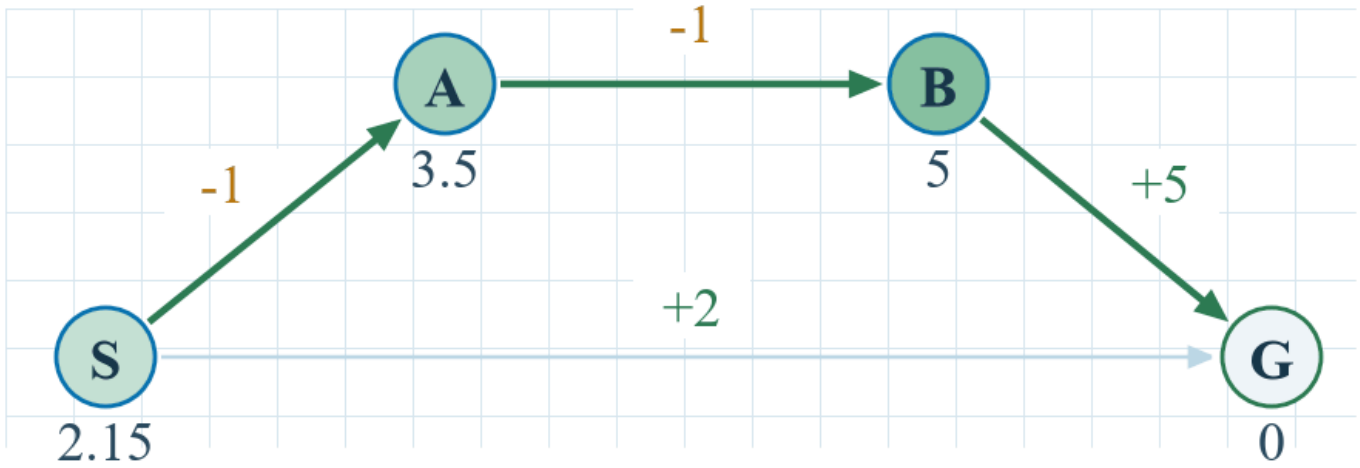
Synchronous Round	Value at S	Value at A	Value at B	Greedy Action at S
0: all values zero	0	0	0	Not yet selected
1	2	−1	5	Directly to G
2	2	3.5	5	Directly to G
3	2.15	3.5	5	Detour via A
4: unchanged	2.15	3.5	5	Detour via A

Each round uses previous values. G stays at 0. Route S→A→B→G returns 2.15 points.

10.3.4 Bellman Equations and Policy Optimization: Value Propagation

How the Arrival Reward Changes the Choice at S

Value backs up along links



Node color: state value

Synchronous sweep 4

$\gamma = 0.9$

Current choice at S

Detour

$V(G) = 0$

$V(S) = 2.15$

Synchronous iteration: colors and readings show values. Gold dots trace backward value flow.

Across rounds, B updates, then A, then S changes its choice. Each round uses previous values.

10.3.4 Bellman Equations and Policy Optimization: Solution Methods

Policy Iteration Alternates Evaluation and Improvement

Fix the current policy
Compute its V or Q
→
Compare every action
Select a higher-value action
→
Evaluate the new policy
Repeat until stable

Stage	Policy at S	Value at S	Basis for Improvement
Evaluate initial policy	Directly to G	2 points	Values at A and B: 3.5 and 5
Improve, then re-evaluate	Detour via A	2.15 points	$-1 + 0.9 \times 3.5 > 2$

For finite state and action sets, bounded rewards, and $0 \leq \gamma < 1$, synchronous exact value iteration converges to the unique optimal value.

10.3.4 Bellman Equations and Policy Optimization: Training and Execution

Model-Based Planning and Learning from Experience

Stage	Information Used	Result
Model-based planning	Known transition probabilities and reward function	A solution from value iteration or policy iteration
Interaction training	Sampled states, actions, and rewards	Value estimates and policy-parameter updates
Fixed-policy evaluation	Multiple episodes excluded from updates	Returns, success rate, and failure distribution
Control execution	Current observation and a fixed policy	Actions sent to the interface and tracked by the controller

Exercises and Discussion

Return Calculations and Sufficient State Information

- **Question 1:** keep rewards unchanged and set the discount to 0.8. What is each route's return? Which route is better?

- **Question 2:** restore the discount to 0.9. At S, choose direct with probability 1/4 and the detour with probability 3/4. What is the state value?

- **Question 3:** two observations have the same position but opposite velocities. Can position alone determine the next-state distribution?

Given: detour rewards −1, −1, 5. Direct reward 2 points. G is terminal.

Reference Answers

Discounting, Averaging, and State Information

$$R_{\text{via}} = -1 - 0.8 + 0.8^2 \times 5 = 1.4 < 2$$

$$V_\mu(S) = \frac{1}{4} \times 2 + \frac{3}{4} \times 2.15 = 2.1125$$

- **Question 3:** position usually does not fully describe motion. Add velocity or use history to estimate the hidden state.
- **Test:** given the current representation and action, does history still change the next-state distribution?

Section Summary

States, Returns, and Policy Optimization

- **System role:** an intelligent policy generates behaviors or references. The external interface and low-level controller execute the motion.
- **Modeling:** the state needs sufficient information. Rewards and discounting define the objective. Termination determines future value.
- **Value relations:** a return is a route outcome. Q fixes the first action. V averages action values under the policy.
- **Common errors:** reward differs from return. Policy evaluation differs from maximization. Better training does not guarantee a global optimum.