FLOWRRA: Flow Recognition-Reconfiguration Agent

Technical Documentation

Abstract

FLOWRRA (Flow Recognition-Reconfiguration Agent) is a multi-agent reinforcement learning system designed for coordinated node movement in dynamic environments with obstacles. The system employs a novel "comet-tail" repulsion mechanism, wave function collapse for stability recovery, and shared reinforcement learning to maintain coherent group behavior while avoiding collisions.

1. System Architecture Overview

The FLOWRRA system consists of six main components:

1. **Node Position Management** (`NodePosition_RL.py`)
2. **Agent Environment** (`EnvironmentA_RL.py`)
3. **External Environment** (`EnvironmentB_RL.py`)
4. **Density Function Estimator** (`DensityFunctionEstimator_RL.py`)
5. **Wave Function Collapse** (`WaveFunctionCollapse_RL.py`)
6. **Reinforcement Learning Orchestrator** (`FLOWRRA_RL.py`)

2. Mathematical Foundations

2.1 Coordinate System and Normalization

The system operates in a normalized coordinate space $[0, 1) \times [0, 1)$ with toroidal topology. For any position vector $\mathbf{p} = (x, y)$:

$$\mathbf{p}_{normalized} = \mathbf{p} \bmod 1$$

Toroidal distance calculation between positions $\mathbf{p}_1$ and $\mathbf{p}_2$:

$$\boldsymbol{\delta} = \mathbf{p}_2 - \mathbf{p}_1$$

$$\boldsymbol{\delta}_{toroidal} = (\boldsymbol{\delta} + 0.5) \bmod 1 - 0.5$$

$$d_{toroidal} = ||\boldsymbol{\delta}_{toroidal}||_2$$

2.2 Node Kinematics

Each node i has state variables:

- Position: $\mathbf{p}_i(t) \in [0,1)^2$
- Discrete angle index: $\theta_{idx,i}(t) \in \{0, 1, ..., N_{angles} - 1\}$
- Target angle index: $\theta_{target,i}(t)$

The continuous angle is: $\theta_i(t) = \frac{2\pi\theta_{idx,i}(t)}{N_{angles}}$

Velocity calculation: $\mathbf{v}_i(t) = \mathbf{p}_i(t) - \mathbf{p}_i(t-1)$ (with toroidal wrapping)

3. Node Position Module

3.1 Movement Dynamics

Rotation Dynamics: The node rotates towards its target angle with maximum rotation speed ω_{max}:

$$\Delta\theta = (\theta_{target} - \theta_{current} + N_{angles}) \bmod N_{angles}$$

If $\Delta\theta \leq N_{angles}/2$ (clockwise optimal):

$$\theta_{new} = \theta_{current} + \min(\Delta\theta, \omega_{max})$$

Otherwise (counter-clockwise optimal):

$$\theta_{new} = \theta_{current} - \min(N_{angles} - \Delta\theta, \omega_{max})$$

Translation Dynamics: Forward movement in current direction:

$$\mathbf{p}_i(t+1) = (\mathbf{p}_i(t) + v_{max} \cdot [\cos\theta_i(t), \sin\theta_i(t)] \cdot \Delta t) \bmod 1$$

3.2 Sensing Mechanism

Each node has sensor range r_{sensor}. For detection of entity j:

1. Calculate toroidal distance: $d_{ij} = ||\boldsymbol{\delta}_{toroidal}||_2$
2. If $d_{ij} < r_{sensor}$:
   - Bearing: $\beta_{ij} = \arctan 2(\delta_y, \delta_x)$

- Relative velocity: $\mathbf{v}_{rel} = \mathbf{v}_j - \mathbf{v}_i$

4. Environment Systems

4.1 Environment A (Agent Environment)

Manages N nodes with discrete action space. Each timestep:

1. Apply actions to nodes
2. Update node positions and orientations
3. Record system snapshot: $S(t) = \{(\mathbf{p}_i(t), \theta_i(t), \mathbf{v}_i(t))\}_{i=1}^N$

4.2 Environment B (External Environment)

Contains static and dynamic obstacles on discrete grid $G \in \mathbb{Z}^{60\times 60}$.

Static obstacles: Fixed positions $\mathcal{O}_{fixed} = \{(x_j, y_j)\}$

Dynamic obstacles: Move randomly with constraint to avoid collisions:

$$\mathbf{p}_{obs}(t+1) \in \{\mathbf{p}_{obs}(t) + \boldsymbol{\delta} : \boldsymbol{\delta} \in \{(-1,0),(1,0),(0,-1),(0,1),(0,0)\}\}$$

subject to collision avoidance.

Continuous coordinates: $\mathbf{p}_{continuous} = \frac{\mathbf{p}_{grid}+0.5}{60}$

5. Density Function Estimator

5.1 Comet-Tail Repulsion Model

The system implements predictive repulsion based on future trajectory projection.

Local Repulsion Grid: Each node i maintains a 4×4 local repulsion grid R_i.

Repulsion Kernel Splatting: For each detected entity j with position $\mathbf{p}_j$ and velocity $\mathbf{v}_j$:

$$\mathbf{p}_{future}^{(k)} = \mathbf{p}_j + k \cdot \mathbf{v}_j \quad \text{for } k = 0, 1, ..., k_f$$

Grid coordinates relative to node i:

$$x_{grid} = \text{clip}((\mathbf{p}_{future,x}^{(k)} - \mathbf{p}_{i,x} + 0.5) \times 4, 0, 3)$$

$$y_{grid} = \text{clip}((\mathbf{p}_{future,y}^{(k)} - \mathbf{p}_{i,y} + 0.5) \times 4, 0, 3)$$

Gaussian Kernel Value:

$$K(k) = \exp\left(-\frac{k^2}{2\sigma_f^2}\right)$$

Repulsion Update:

$$R_i[y_{grid}, x_{grid}] += \eta \cdot \gamma_f^k \cdot K(k)$$

Where:

- η: Learning rate for splatting
- γ_f: Decay factor for future projections
- k_f: Maximum projection steps
- σ_f: Kernel width

5.2 Global Field Maintenance

The global repulsion field $\Phi(\mathbf{r})$ undergoes temporal decay:

$$\Phi(\mathbf{r}, t+1) = (1 - \lambda_{decay})\Phi(\mathbf{r}, t)$$

6. Wave Function Collapse

6.1 Coherence Assessment

The system coherence is calculated using entropy of the combined repulsion grids:

$$H = -\sum_i p_i \log_2(p_i)$$

where p_i are normalized repulsion values: $p_i = \frac{|\Phi_i|}{\sum_j |\Phi_j|}$

Coherence Score:

$$C = 1 - \frac{H}{H_{max}}$$

where $H_{max} = \log_2(N_{grid})$ is maximum possible entropy.

6.2 Manifold Smoothing

When coherence drops below threshold $\tau_{collapse}$ for τ consecutive steps, the system triggers collapse recovery.

Coherent Tail Identification: Find sequence of length L_{tail} where $C(t-L_{tail} :t-1) > \tau_{collapse}$

Gaussian Weighted Averaging:

$$w_k = \frac{\exp(-0.5((L_{tail} - k)/(L_{tail}/4))^2)}{\sum_{j=1}^{L_{tail}} \exp(-0.5((L_{tail} - j)/(L_{tail}/4))^2)}$$

Smoothed Position Recovery:

$$\mathbf{p}_i^{smooth} = \sum_{k=1}^{L_{tail}} w_k \mathbf{p}_i(t-k)$$

7. Reinforcement Learning Framework

7.1 State Representation

Each node contributes a 36-dimensional state vector:

- Position and velocity: $(\mathbf{p}_i, \mathbf{v}_i) \in \mathbb{R}^4$
- Sensor data (4 detections × 4 values): $\mathbf{s}_i \in \mathbb{R}^{16}$
- Local repulsion grid: $R_i \in \mathbb{R}^{16}$ (flattened 4×4)

Total State: $\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, ..., \mathbf{s}_N] \in \mathbb{R}^{36N}$

7.2 Action Space

Combined Action Space: Position movement × Angle adjustment = $4 \times 4 = 16$ actions per node

Action decomposition for node i:

- Position action: $a_{pos,i} = \lfloor a_i/4 \rfloor$

- Angle action: $a_{angle,i} = a_i \bmod 4$

7.3 Two-Stage Action Execution

Stage 1: Position Actions Apply position modifications, update repulsion field, check coherence. If $C < \tau_{collapse}$: trigger WFC, return zero reward.

Stage 2: Angle Actions If Stage 1 coherent, apply angle modifications, final coherence check. If $C < \tau_{collapse}$: trigger WFC, return zero reward. Otherwise: return coherence-based rewards.

7.4 Q-Network Architecture

Network Structure:

- Input: State vector $\mathbf{S} \in \mathbb{R}^{36N}$
- Hidden layers: $\mathbb{R}^{36N} \to \mathbb{R}^{128} \to \mathbb{R}^{128}$
- Output: Q-values $\mathbb{R}^{16N}$ (16 actions per node)

Loss Function:

$$\mathcal{L} = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}[(r + \gamma \max_{a'} Q_{target}(s', a') - Q(s, a))^2]$$

8. System Dynamics and Training

8.1 Training Loop

1. **State Observation:** $\mathbf{S}(t) = \text{get\_state}()$
2. **Action Selection:** $\mathbf{A}(t) = \epsilon\text{-greedy}(Q(\mathbf{S}(t)))$
3. **Two-Stage Execution:** $\mathbf{R}(t), done, info = \text{step}(\mathbf{A}(t))$
4. **Experience Storage:** $(\mathbf{S}(t), \mathbf{A}(t), \mathbf{R}(t), \mathbf{S}(t+1), done) \to \mathcal{D}$
5. **Network Update:** $\theta \leftarrow \theta - \nabla_\theta \mathcal{L}$

8.2 Reward Structure

Base reward is coherence score:

$$r_i(t) = C(t)$$

Special cases:

- WFC triggered: $r_i(t) = 0$ for all i

- Episode termination: Based on final coherence

8.3 Exploration Strategy

Epsilon-greedy with exponential decay:

$$\epsilon(t) = \max(\epsilon_{min}, \epsilon_0 \cdot \gamma_\epsilon^t)$$

9. Deployment and Evaluation

9.1 Deployment Protocol

1. Load trained Q-network weights
2. Set exploration rate $\epsilon = 0$ (pure exploitation)
3. Execute actions greedily: $a_i = \arg\max_a Q(\mathbf{S}, a)$
4. Monitor system coherence and stability

9.2 Performance Metrics

- **Average Coherence:** $\bar{C} = \frac{1}{T}\sum_{t=1}^{T} C(t)$
- **Stability:** Fraction of timesteps without WFC triggers
- **Collision Rate:** Frequency of node-obstacle encounters
- **Formation Maintenance:** Loop connectivity preservation

10. Implementation Details

10.1 Hyperparameters

Parameter	Symbol	Default Value
Learning rate	α	0.001
Discount factor	γ	0.99
Repulsion learning rate	η	0.5
Future projection steps	k_f	5
Collapse threshold	$\tau_{collapse}$	0.25
History length	$L_{history}$	200
Tail length	L_{tail}	15

10.2 Computational Complexity

- **State computation:** $O(N^2 + N \cdot |O|)$ per timestep
- **Repulsion update:** $O(N \cdot k_f \cdot (N + |O|))$
- **Q-network forward pass:** $O(36N \cdot 128 + 128^2 + 128 \cdot 16N)$
- **WFC smoothing:** $O(N \cdot L_{tail})$

11. Conclusion

FLOWRRA represents a novel approach to multi-agent coordination that combines:

1. **Predictive Collision Avoidance:** Through comet-tail repulsion modeling
2. **Stability Recovery:** Via wave function collapse and manifold smoothing
3. **Distributed Learning:** Using shared Q-network architecture
4. **Coherence Maintenance:** Through entropy-based system monitoring

The system demonstrates emergent coordination behaviors while maintaining robustness to environmental perturbations and internal instabilities.

References and Code Structure

The implementation spans multiple Python modules:

- `NodePosition_RL.py`: Individual agent kinematics and sensing
- `EnvironmentA_RL.py`: Multi-agent environment management
- `EnvironmentB_RL.py`: External obstacle dynamics
- `DensityFunctionEstimator_RL.py`: Repulsion field computation
- `WaveFunctionCollapse_RL.py`: Stability recovery mechanism
- `FLOWRRA_RL.py`: Main orchestration and RL integration
- `RLAgent.py`: Deep Q-Network implementation
- `main_runner_rl.py`: Training and deployment execution

The system architecture enables scalable multi-agent coordination with theoretical foundations in dynamical systems, information theory, and reinforcement learning.