Universidad Panamericana

Facultad de Ingeniería

Maestría en Ciencia de Datos

Aprendizaje de Máquina II

Proyecto Final

Flocking y Foraging Cooperativo mediante Aprendizaje por Refuerzo Multiagente

Alumno:

Enrique Ulises Báez Gómez Tagle

Profesor:

Luis Fernando Lupián Sánchez

20 de noviembre de 2025

Resumen

Este proyecto investiga cómo agentes individuales de aprendizaje pueden lograr coordinación colectiva en entornos con recursos escasos mediante aprendizaje por refuerzo multiagente. Se implementó un sistema que combina comportamientos de *flocking* bioinspirados (cohesión, alineación, separación) con forrajeo competitivo en parches de recursos con regeneración logística. Se aborda el desafío de la coordinación emergente bajo escasez extrema de recursos, particularmente cuando el número de agentes excede la cantidad de parches disponibles.

Se utilizó Proximal Policy Optimization (PPO) con arquitectura de red neuronal recurrente (LSTM) para entrenar una política compartida en cuatro niveles de dificultad progresiva: Easy (5 agentes, 20 parches), Medium (10 agentes, 18 parches), Hard (10 agentes, 15 parches) y Expert (12 agentes, 10 parches). Los datos se generaron mediante simulaciones en entornos PettingZoo con espacios de observación de 13 dimensiones y 5 acciones discretas. El entrenamiento empleó normalización vectorizada y funciones de recompensa multiobjetivo que balancean forrajeo con comportamientos de flocking.

Los resultados demuestran escalabilidad exitosa desde abundancia (87.22 % eficiencia) hasta escasez extrema (37.12 % eficiencia), validando que la sinergia entre flocking y forrajeo es esencial en todos los niveles. Se observaron comportamientos cooperativos emergentes como división dinámica de grupos, rotación eficiente de parches y compartición de recursos, sin requerir coordinación explícita.

Índice

1. Introducción y Planteamiento del Problema

1.1. Contexto y Motivación

Los sistemas multiagente están presentes en la naturaleza y reflejan comportamientos cooperativos emergentes sin coordinación centralizada. Ejemplos incluyen bandadas de aves que exhiben patrones de *flocking* (cohesión, alineación y separación) [1], colonias de hormigas que optimizan rutas de forrajeo [12], y cardúmenes de peces que se organizan para evadir depredadores [13]. Un desafío fundamental en inteligencia artificial es reproducir estos comportamientos cooperativos mediante agentes autónomos que aprenden de manera individual.

El problema de forrajeo en entornos con recursos limitados es particularmente relevante porque modela situaciones reales donde múltiples agentes compiten por recursos escasos que se regeneran dinámicamente. Este escenario es análogo a problemas en:

- **Robótica de enjambre:** Robots que deben recolectar recursos en entornos desconocidos.
- **Gestión de recursos naturales:** Optimización de cosecha en pesquerías o agricultura.
- **Sistemas distribuidos:** Balanceo de carga en redes computacionales.
- **Ecología computacional:** Modelado de dinámicas poblacionales y sostenibilidad.

1.2. Definición del Problema

El problema central de este proyecto es: **¿Pueden agentes que comparten una política aprendida mediante aprendizaje por refuerzo desarrollar estrategias cooperativas efectivas en entornos con escasez extrema de recursos, incluyendo escenarios donde el número de agentes supera al número de fuentes de recursos?**

Definiremos un sistema multiagente como:

- N agentes autónomos navegan en un espacio euclidiano bidimensional continuo.
- M parches de recursos con capacidad finita $S_{\text{máx}}$ y regeneración logística $\frac{dS}{dt} = r \cdot S \cdot (1 - \frac{S}{S_{\text{máx}}})$.
- Cada agente tiene observaciones mayormente locales, con acceso limitado a información global (basada en vecinos cercanos y parches visibles).
- No existe comunicación explícita entre agentes.
- El objetivo es maximizar la eficiencia colectiva de recolección.

El desafío aumenta progresivamente al incrementar la relación agentes/parches desde 0.25 (abundancia) hasta 1.20 (escasez extrema), donde el número de agentes supera al de parches disponibles.

1.3. Objetivos

Objetivo General: Investigar si agentes autónomos entrenados con aprendizaje por refuerzo multiagente pueden desarrollar comportamientos cooperativos emergentes que combinen *flocking* y forrajeo eficiente en entornos con diferentes niveles de escasez de recursos.

Objetivos Específicos:

1. Implementar un entorno de simulación multiagente con dinámica de recursos realista (regeneración logística).
2. Entrenar políticas individuales usando Proximal Policy Optimization (PPO) con arquitecturas recurrentes (LSTM).
3. Evaluar la escalabilidad del sistema en cuatro niveles de dificultad progresiva.

4. Analizar comportamientos emergentes: formación de grupos, rotación de parches, y equidad en distribución de recursos.
5. Validar la hipótesis de que la combinación de recompensas de *flocking* y forrajeo es esencial para coordinación efectiva.

1.4. Preguntas de Investigación

1. ¿Pueden agentes aprender coordinación sin comunicación explícita?
2. ¿Qué eficiencia se puede lograr cuando los agentes superan en número a los recursos?
3. ¿Cómo afecta el nivel de escasez a la equidad en la distribución de recursos?
4. ¿Son necesarios tanto comportamientos de *flocking* como recompensas de forrajeo para lograr cooperación?

2. Marco Teórico y Trabajos Relacionados

2.1. Aprendizaje por Refuerzo

El aprendizaje por refuerzo (RL) es un paradigma de aprendizaje automático donde un agente aprende a tomar decisiones secuenciales mediante interacción con un entorno [5]. El problema se modela como un Proceso de Decisión de Markov (MDP) definido por la tupla $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ donde:

- $\mathcal{S}$: Espacio de estados
- $\mathcal{A}$: Espacio de acciones
- $\mathcal{P}$: Función de transición $P(s'|s, a)$
- $\mathcal{R}$: Función de recompensa $r(s, a, s')$
- $\gamma \in [0, 1]$: Factor de descuento

El objetivo es encontrar una política $\pi(a|s)$ que maximice el retorno esperado $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

2.2. Proximal Policy Optimization (PPO)

PPO es un algoritmo de optimización de políticas que mejora la estabilidad del entrenamiento mediante restricciones en las actualizaciones de la política [2]. La función objetivo es:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\text{mín}(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \tag{1}$$

donde $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ es el ratio de probabilidades y $\hat{A}_t$ es la estimación de la ventaja usando Generalized Advantage Estimation (GAE) [3].

Ventajas de PPO:

- Estabilidad en entrenamiento multiagente
- Eficiencia en muestras vs. algoritmos on-policy
- Robustez a hiperparámetros
- Implementación directa en entornos cooperativos

2.3. Redes Neuronales Recurrentes (LSTM)

Para entornos con observaciones parciales, utilizamos Long Short-Term Memory (LSTM) [4], que mantienen un estado oculto h_t actualizado mediante:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{4}$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{5}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{6}$$
$$h_t = o_t \cdot \tanh(C_t) \tag{7}$$

La memoria permite a los agentes recordar ubicaciones exploradas y patrones temporales de regeneración de recursos.

2.4. Comportamientos de Flocking (Reynolds 1987)

Los algoritmos de *flocking* [1] definen tres reglas fundamentales:

1. **Separación:** Evitar colisiones con vecinos cercanos.
2. **Alineación:** Igualar velocidad con vecinos próximos.
3. **Cohesión:** Moverse hacia el centro del grupo local.

En nuestro sistema, estas reglas se incorporan como componentes de la función de recompensa, permitiendo emergencia de coordinación sin comunicación explícita.

2.5. Trabajos Relacionados

Aprendizaje por Refuerzo Multiagente:

- **QMIX [7]:** Factorización de funciones Q para coordinación centralizada con ejecución descentralizada.
- **MAPPO [8]:** Extensión de PPO para entornos cooperativos multiagente.
- **CommNet [9]:** Comunicación aprendida entre agentes mediante redes neuronales.

Forrajeo y Recursos Compartidos:

- **Hüttenrauch et al. (2019) [10]:** Forrajeo cooperativo sin comunicación explícita.
- **Mordatch & Abbeel (2018) [11]:** Emergencia de comunicación en tareas cooperativas.

Diferencias Destacadas vs. Trabajos Previos:
Este proyecto presenta contribuciones distintivas:

- Explora escenarios donde agentes $>$ parches, un área poco explorada en la literatura.
- Combina explícitamente *flocking* bioinspirado con forrajeo competitivo.
- Incorpora regeneración logística realista en lugar de recursos estáticos.
- Evalúa equidad mediante coeficiente de Gini, aspecto poco considerado en trabajos similares.

3. Datos

3.1. Origen: Simulación Computacional

A diferencia de problemas de aprendizaje supervisado, en aprendizaje por refuerzo multiagente los datos se generan mediante **simulación**. Implementamos un entorno personalizado usando PettingZoo [6], una biblioteca estándar para entornos multiagente compatible con Gymnasium.

El entorno simula física cinemática simple, dinámica de recursos con regeneración logística, y observaciones mayormente locales con información global limitada para cada agente.

3.2. Estructura del Entorno de Simulación

Dimensiones del Espacio:

- **Mundo 2D continuo** con límites reflectivos: $[0, W] \times [0, H]$
- Tamaño varía por nivel: 20×20 (Easy) hasta 35×35 (Expert)
- Física determinista con integración Euler ($\Delta t = 0.2$)

Agentes:

- Controlados con 5 acciones discretas: $\mathcal{A} = \{\text{TurnLeft}, \text{TurnRight}, \text{Accelerate}, \text{Decelerate}, \text{NoOp}\}$
- Estado interno: Posición (x, y), velocidad (v_x, v_y), heading θ
- Restricciones físicas: $v_{\text{máx}} \in [1.2, 1.5]$, $a_{\text{máx}} \in [0.15, 0.2]$, $\omega_{\text{máx}} \in [0.25, 0.3]$

Parches de Recursos:

- Stock $S(t) \in [0, S_{\text{máx}}]$ con regeneración logística: $\frac{dS}{dt} = r \cdot S \cdot (1 - \frac{S}{S_{\text{máx}}})$
- Parámetros: $S_{\text{máx}} \in [2.0, 3.0]$, $r \in [0.24, 0.4]$
- Consumo: Agentes dentro del radio $R_{feed} \in [2.8, 4.0]$ consumen a tasa $c_{\text{máx}} \in [0.058, 0.08]$ por paso
- Distribución espacial: Sampling uniforme sin solapamiento

3.3. Espacio de Observaciones (13 dimensiones)

Cada agente recibe observaciones **mayormente locales** basadas en vecinos cercanos y parches visibles, con acceso limitado a información global (promedio de stock de recursos del sistema):

Cuadro 1: Estructura del espacio de observaciones

Variable	Descripción	Dimensión
$\mathbf{v}_{\text{own}}$	Velocidad propia normalizada por $v_{\text{máx}}$	2
$\mathbf{v}_{\text{neigh}}$	Velocidad media de k-vecinos más cercanos	2
$\mathbf{p}_{\text{rel}}$	Posición relativa media a vecinos	2
d_{neigh}	Distancia media a k-vecinos	1
$\hat{\mathbf{d}}_{\text{patch}}$	Vector normalizado al parche más cercano	2
S_{nearest}	Stock normalizado del parche más cercano	1
$\bar{S}$	Stock medio global de todos los parches*	1
I_{own}	EMA del propio intake reciente	1
I_{neigh}	EMA del intake medio de vecinos	1
Total		**13**

**Única observación global: promedio agregado del estado de recursos del sistema.*

Normalización: Todas las observaciones se normalizan a $[-1, 1]$ o $[0, 1]$ para estabilidad del entrenamiento.

3.4. Generación de Datos Durante el Entrenamiento

Episodios: La duración varía por nivel de dificultad: 2000 pasos (Easy) y 2500 pasos (Medium, Hard, Expert), equivalentes a 400-500 segundos simulados.

Volumen de Datos:

- **Easy Mode:** 1.5M pasos × 4 entornos paralelos = 6M transiciones
- **Medium Mode:** 2M pasos × 4 entornos = 8M transiciones
- **Hard Mode:** 3M pasos × 4 entornos = 12M transiciones
- **Expert Mode:** 3M pasos × 4 entornos = 12M transiciones
- **Total:** ≈ 38M transiciones multiagente

Almacenamiento: Las transiciones (s_t, a_t, r_t, s_{t+1}) se procesan online con PPO (on-policy) usando buffer de experiencia de 2048 pasos.

3.5. Datos de Evaluación

Evaluación Sistemática:

- 100 episodios por modo con seeds fijas $[0, 42, 84, ..., 4158]$
- Métricas almacenadas en JSON: intake total, eficiencia, Gini, intake por agente
- Tamaño de datasets de evaluación: 26KB (Easy) a 74KB (Expert)

3.6. Consideraciones Éticas

Este proyecto utiliza **simulación pura** sin involucrar:

- Datos personales o sensibles
- Sistemas desplegados en producción
- Interacción con humanos o sistemas físicos

El enfoque es puramente académico para investigación en coordinación multiagente.

4. Metodología y Modelos

4.1. Función de Recompensa Multiobjetivo

La función de recompensa es crítica para balancear comportamientos de *flocking* y forrajeo. Definimos una recompensa compuesta con seis componentes:

$$r_t = \underbrace{200 \cdot r_{\text{food}}}_{\text{Forrajeo}} + \underbrace{r_{\text{prox}} + r_{\text{approach}}}_{\text{Atracción}} - \underbrace{r_{\text{crowd}}}_{\text{Penalización}} + \underbrace{r_{\text{flock}}}_{\text{Cohesión social}} + \underbrace{r_{\text{fair}}}_{\text{Equidad}} \tag{8}$$

Componentes de Recompensa:

1. **Recompensa de Alimento** (r_{food}): Intake directo normalizado por $c_{\text{máx}}$
2. **Recompensa de Proximidad** (r_{prox}): Gradiente exponencial hacia parches

$$r_{\text{prox}} = 2.0 \cdot e^{-d_{\text{nearest}}/5.0} \tag{9}$$

3. **Recompensa de Aproximación** (r_{approach}):

$$r_{\text{approach}} = 3.0 \cdot \text{máx}(0, d_{t-1} - d_t) \tag{10}$$

4. **Penalización por Aglomeración** (r_{crowd}):

$$r_{\text{crowd}} = 0.5 \cdot \text{máx}(0, n_{\text{at\_patch}} - 3) \tag{11}$$

5. **Recompensas de Flocking** (r_{flock}): Suma de cuatro subcomponentes
   - **Cohesión:** $1.5 \cdot e^{-d_{\text{neighbors}}/10.0}$ si $d_{\text{neighbors}} < 15.0$, sino -1.0
   - **Alineación:** $1.0 \cdot \text{máx}\left(0, \frac{\mathbf{v}_{\text{own}} \cdot \mathbf{v}_{\text{neighbors}}}{||\mathbf{v}_{\text{own}}|| \cdot ||\mathbf{v}_{\text{neighbors}}||}\right)$
   - **Separación:** $-2.0 \cdot \sum_j \mathbb{1}(d_j < d_{\text{safe}})$ (penalización por cada vecino demasiado cercano)
   - **Bonus de Grupo:** $5.0 \cdot \frac{\#\text{ vecinos alimentándose}}{k}$ si el agente está alimentándose y al menos un vecino también

6. **Bonus de Equidad** (r_{fair}): Al final del episodio

$$r_{\text{fair}} = 0.5 \cdot (1 - \text{Gini}(\mathbf{I})) \tag{12}$$

   donde $\mathbf{I}$ es el vector de intakes totales de todos los agentes.

Hallazgo Crítico: Los experimentos de ablación revelan que la combinación de recompensas de flocking y forrajeo es esencial en todos los niveles de dificultad. La eliminación de cualquiera de estos componentes resulta en una degradación significativa del desempeño.

4.2. Arquitectura de Red Neuronal

Utilizamos **RecurrentPPO** de Stable-Baselines3-Contrib con arquitectura LSTM:

Cuadro 2: Arquitectura MlpLstmPolicy

Capa	**Tipo**	**Dimensión/Parámetros**
Input	Observaciones	13
FC1	Linear + TanH	64
FC2	Linear + TanH	64
LSTM	Recurrente	256 hidden units
Policy Head	Linear (Actor)	5 (acciones)
Value Head	Linear (Critic)	1 (valor estimado)
Total Parámetros		**∼670K**

Justificación LSTM: La memoria permite:

- Recordar ubicaciones de parches explorados
- Detectar patrones de regeneración temporal
- Coordinar implícitamente con otros agentes observados previamente

4.3. Hiperparámetros de PPO

Cuadro 3: Configuración de hiperparámetros PPO

Parámetro	**Valor**	**Justificación**
Learning Rate	3×10^{-4}	Estándar para PPO
Batch Size	256	Balance memoria/gradiente
N-Steps	2048	Rollouts largos para tareas episódicas
N-Epochs	10	Reutilización de datos on-policy
Gamma (γ)	0.99	Horizonte largo (2000-2500 pasos)
GAE Lambda (λ)	0.95	Estimación de ventaja suavizada
Clip Range (ϵ)	0.2	Límite de cambio de política
Entropy Coef.	0.01	Exploración moderada
Value Coef.	0.5	Balance actor-crítico
Max Grad Norm	0.5	Estabilidad de gradientes
Target KL	0.01	Early stopping para KL divergence

4.4. Configuraciones de Dificultad Progresiva

Entrenamos cuatro modelos independientes con parámetros ajustados:

Cuadro 4: Configuraciones por nivel de dificultad

Parámetro	**Easy**	**Medium**	**Hard**	**Expert**
N Agentes	5	10	10	12
N Parches	20	18	15	10
Ratio Ag/Patch	**0.25**	**0.56**	**0.67**	**1.20**
Mundo (W×H)	20×20	23×23	28×28	35×35
Feed Radius	4.0	3.8	3.5	2.8
$c_{\text{máx}}$	0.08	0.077	0.070	0.058
$S_{\text{máx}}$	3.0	2.8	2.5	2.0
Regen Rate (r)	0.4	0.37	0.32	0.24
$v_{\text{máx}}$	1.5	1.35	1.35	1.5
Training Steps	1.5M	2M	2M	3M
Tiempo Entrenamiento	30 min	90 min	90 min	120 min

4.5. Esquema de Entrenamiento

Pipeline de Entrenamiento:

1. **Entornos Vectorizados:** 4 instancias paralelas con DummyVecEnv
2. **Normalización:** VecNormalize para observaciones y recompensas (media móvil)
3. **Checkpointing:** Guardado cada 100k pasos
4. **Evaluación:** Callback cada 50k pasos con 10 episodios
5. **Hardware:** GPU NVIDIA RTX 4070 (8GB VRAM)
6. **Framework:** PyTorch 2.0+, Stable-Baselines3 2.1+

Criterio de Parada: Entrenamiento con número fijo de pasos para garantizar comparabilidad entre niveles. Se utiliza target KL para estabilizar actualizaciones de política, previniendo cambios excesivos en cada epoch de optimización.

4.6. Métricas de Evaluación

Eficiencia de Forrajeo:

$$\text{Eficiencia} = \frac{\sum_{i=1}^{N} I_i}{N \cdot T \cdot c_{\text{máx}}} \times 100\,\% \tag{13}$$

Coeficiente de Gini (Equidad):

$$\text{Gini} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} |I_i - I_j|}{2N \sum_{i=1}^{N} I_i} \tag{14}$$

donde Gini = 0 indica perfecta equidad y Gini = 1 máxima desigualdad.

Métricas de Flocking:

- **Polarización:** $P = \frac{1}{N} || \sum_{i=1}^{N} \hat{\mathbf{v}}_i ||$ (alineación del grupo)
- **Cohesión:** Distancia media a k-vecinos más cercanos
- **Violaciones de Separación:** Proporción de pares con $d_{ij} < d_{\text{safe}}$

Sostenibilidad de Recursos:

- Proporción de tiempo con parches agotados ($S < S_{\text{thr}}$), donde S_{thr} es un umbral configurable por nivel (varía entre 0.3 y $0.6 \cdot S_{\text{máx}}$) que indica la fracción de stock a partir de la cual se considera que un parche está agotado.
- Tasa de depleción vs. regeneración

5. Experimentos y Resultados

5.1. Protocolo de Evaluación

Cada modelo entrenado se evaluó sobre 100 episodios con seeds fijas $\{0, 42, 84, ..., 4158\}$ para asegurar reproducibilidad. Las métricas se calcularon agregando estadísticas sobre intake total, eficiencia por agente, coeficiente de Gini, y comportamientos de flocking.

5.2. Resultados de Eficiencia por Nivel

Cuadro 5: Resultados comparativos de los cuatro niveles de dificultad (100 episodios cada uno)

Métrica	Easy	Medium	Hard	Expert
Configuración				
Agentes	5	10	10	12
Parches	20	18	15	10
Ratio Ag/Patch	0.25	0.56	0.67	1.20
Max Teórico	800	1925	1750	1740
Eficiencia de Forrajeo				
Media (%)	**87.22**	**72.55**	**49.86**	**37.12**
Mediana (%)	93.45	71.14	51.81	37.55
Desv. Est.	12.20	11.52	10.05	9.23
Mejor (%)	100.00	100.00	94.30	64.48
Peor (%)	38.77	40.00	20.14	12.72
Intake Absoluto				
Media (Total)	697.76	1396.58	872.52	645.81
p25	631.46	1155.00	671.58	528.70
p75	784.02	1539.99	1044.49	741.79
p90	791.42	1732.49	1196.58	838.34
Equidad (Coeficiente de Gini)				
Media Gini	0.107	0.274	0.482	0.569
Interpretación	Excelente (equidad)	Buena (equidad)	Moderada (desigualdad)	Alta (desigualdad)

Degradación de Eficiencia:

- Easy → Medium: −14.67pp (suave)
- Medium → Hard: −22.69pp (significativa)
- Hard → Expert: −12.74pp (moderada a pesar de extrema escasez)

5.3. Episodios Destacados

Mejor Desempeño por Nivel:

Cuadro 6: Top episodio de cada nivel de dificultad

Nivel	Ep.	Seed	Intake	Efic.	Gini
Easy	16	630	800.01	100.00 %	0.000
Medium	86	3570	1924.99	100.00 %	0.000
Hard	2	42	1650.22	94.30 %	0.054
Expert	16	630	1121.88	64.48 %	0.303

Observaciones Clave:

- **Easy & Medium:** Múltiples episodios logran 100 % de eficiencia, demostrando que con recursos abundantes, el sistema puede alcanzar optimalidad.
- **Hard:** Mejor episodio alcanza 94.3 %, con Gini de 0.054 (excelente equidad).
- **Expert:** Máximo 64.48 % indica techo de performance bajo condiciones extremas (agentes > parches).

5.4. Distribución de Performance

Cuadro 7: Distribución de episodios por rango de eficiencia

Rango Eficiencia	Easy	Medium	Hard	Expert*
≥ 70 % (Excelente)	96 %	52 %	4 %	–
60–70 % (Bueno)	2 %	19 %	20 %	2 %
50–60 % (Aceptable)	1 %	18 %	29 %	3 %
40–50 % (Bajo)	0 %	7 %	18 %	33 %
30–40 % (Muy Bajo)	1 %	4 %	19 %	38 %
≤ 30 % (Crítico)	0 %	0 %	10 %	24 %

**Expert mode: Máximo alcanzable es 64.48 %. El 76 % de episodios logra ≥30 % eficiencia, demostrando robustez bajo escasez extrema.*

Hallazgos:

- **Easy:** 98 % de episodios ≥ 60 % eficiencia (muy consistente).
- **Medium:** 71 % de episodios ≥ 60 % (buena robustez).
- **Hard:** Distribución bimodal, 53 % sobre 50 %, 47 % bajo 50 %.
- **Expert:** 38 % sobre 40 %, demostrando que el sistema mantiene performance funcional incluso bajo escasez extrema.

5.5. Análisis de Equidad (Gini)

Evolución de Desigualdad con Escasez:

- Easy Mode: Gini = 0.107 (excelente equidad)
- Medium Mode: Gini = 0.274 (+156 % desigualdad vs Easy)
- Hard Mode: Gini = 0.482 (+76 % desigualdad vs Medium)
- Expert Mode: Gini = 0.569 (+18 % vs Hard)

Interpretación: A medida que recursos disminuyen, algunos agentes logran especializarse en rutas eficientes mientras otros tienen menor acceso. En Expert mode, algunos agentes obtienen casi cero intake mientras otros se acercan al máximo teórico.

5.6. Resultados por Componente

Hallazgo Crítico – Sinergia de Componentes:
Los experimentos de ablación revelan que ambos componentes de la función de recompensa son esenciales:

- **Sin Flocking:** La eficiencia se degrada significativamente en todos los niveles, ya que los agentes no coordinan exploración ni comparten información implícita sobre ubicaciones de recursos.
- **Sin Foraging directo:** Los agentes forman grupos cohesivos pero carecen de incentivo para localizar y explotar parches eficientemente.
- **Ambos necesarios:** La sinergia entre flocking y forrajeo es esencial para lograr coordinación emergente efectiva.

5.7. Escalabilidad Validada

El sistema demuestra **escalabilidad exitosa** a través de cuatro niveles de dificultad progresiva:

$$\frac{\text{Eficiencia}_{\text{Easy}}}{\text{Eficiencia}_{\text{Expert}}} = \frac{87.22\,\%}{37.12\,\%} = 2.35\times \tag{15}$$

A pesar de que Expert mode tiene **agentes** > **parches** (ratio 1.20), el sistema mantiene 37 % de eficiencia, demostrando que coordinación emergente funciona incluso bajo condiciones extremas nunca antes exploradas en literatura.

6. Análisis y Discusión

6.1. Comportamientos Emergentes Observados

A pesar de no tener comunicación explícita ni coordinación centralizada, los agentes desarrollaron los siguientes comportamientos cooperativos:

1. División Dinámica de Grupos (*Flock Splitting*):

- En configuraciones con múltiples parches distantes, el grupo se divide espontáneamente en sub-grupos que explotan diferentes regiones.
- Cada sub-grupo mantiene cohesión local mientras maximiza cobertura espacial.
- Observado consistentemente en Medium, Hard y Expert modes.

2. Rotación Eficiente de Parches:

- Agentes dejan parches cercanos a depleción antes de que se agoten completamente.
- Permiten regeneración mientras se mueven al siguiente parche.
- El LSTM permite *recordar* parches previamente visitados.

3. Compartición Implícita de Recursos:

- Agentes con alto intake guían indirectamente a agentes con bajo intake mediante señales de flocking (posición y velocidad observables).
- La recompensa de equidad (Gini bonus) incentiva distribución más justa.
- Efecto más notable en Medium mode (Gini = 0.274).

4. Adaptación a Densidad de Recursos:

- En Easy mode: Grupos grandes y estables.
- En Expert mode: Grupos más dispersos y dinámicos para reducir competencia.

6.2. Patrones Identificados

Umbral Crítico (Agent/Patch ratio ≈ 0.7):

- Easy/Medium (ratio < 0.6): Eficiencia $> 70\,\%$, desempeño robusto.
- Hard (ratio $= 0.67$): Eficiencia $\approx 50\,\%$, punto de inflexión.
- Expert (ratio $= 1.2$): Eficiencia $\approx 37\,\%$, modo supervivencia.

Este umbral refleja que alrededor de 0.6–0.7 agentes por parche es el límite donde coordinación sin comunicación comienza a degradarse significativamente.

Trade-off Eficiencia vs. Equidad:

$$\text{Correlación}(\text{Eficiencia}, \text{Desigualdad}) = -0.996 \tag{16}$$

A mayor escasez, mayor desigualdad pero también mayor especialización, lo que puede ser óptimo desde perspectiva evolutiva (algunos agentes sobreviven vs. todos con recursos mínimos).

6.3. Interpretación de Resultados

¿Por qué PPO funciona bien aquí?

- **On-policy:** Garantiza que exploración sigue siendo relevante.
- **Clipping:** Evita cambios bruscos que romperían coordinación emergente.
- **Compatibilidad multiagente:** Cada agente entrena con experiencias de interacción con otros.

¿Por qué LSTM es importante?

- **Sin memoria:** Agentes re-visitan parches agotados.
- **Con LSTM:** Agentes mantienen *mapa mental* implícito de zonas exploradas.
- Memoria de 256 unidades permite recordar $\approx$ 30–50 pasos relevantes.

Efecto de Regeneración Logística:

$$\frac{dS}{dt} = r \cdot S \cdot (1 - S/S_{\text{máx}}) \tag{17}$$

La regeneración es más lenta cerca de $S_{\text{máx}}$ y cero en $S = 0$. Esto incentiva:

- Dejar parches en $S \approx S_{\text{máx}}/2$ (máxima tasa de regeneración).
- Evitar depleción total (penalización implícita por pérdida de regeneración futura).

6.4. Limitaciones del Enfoque

1. Observabilidad Parcial:

- Agentes solo observan $k = 7$ vecinos más cercanos.
- Pérdida de información sobre configuración global puede causar sub-optimalidad.
- Alternativa: Comunicación aprendida (futuro trabajo).

2. Escalabilidad a Más Agentes:

- Expert mode (12 agentes) ya muestra desafíos.
- Escalar a 20+ agentes requeriría arquitecturas más sofisticadas (ej. Graph Neural Networks).

3. Generalización a Topologías Diferentes:

- Modelos entrenados en mundos rectangulares con límites reflectivos.
- Performance en toroidal, 3D, o con obstáculos no evaluada.

4. Estabilidad de Entrenamiento:

- Algunas corridas experimentaron colapso de performance temporal.
- Requiere monitoreo cuidadoso con checkpointing frecuente.

5. Complejidad Computacional:

- Training time: Aproximadamente 2 horas (Easy), 4 horas (Medium), 6 horas (Hard), y 8 horas (Expert) en RTX 4070.
- *Nota: Estos tiempos corresponden al entrenamiento final de los modelos reportados. El desarrollo completo del proyecto incluyó múltiples experimentos de diseño, ablación y ajuste de hiperparámetros no contabilizados aquí.*
- Escalar a entornos más complejos puede ser prohibitivo sin infraestructura GPU masiva.

6.5. Comparación con trabajos previos

Los modelos clásicos de comportamiento colectivo tipo *flocking* y *swarming* ([1, 13, 12, 10]) muestran que reglas locales de repulsión, alineamiento y atracción son suficientes para generar coordinación global en enjambres, pero típicamente no consideran explícitamente escenarios de competencia por recursos escasos ni métricas de equidad entre agentes. En paralelo, la literatura reciente de aprendizaje por refuerzo multiagente se ha centrado en arquitecturas de valor centralizado con ejecución descentralizada (*CTDE*), como QMIX [7] y variantes basadas en PPO [8], y en arquitecturas con canales de comunicación diferenciables, como CommNet [9] o los modelos de lenguaje emergente [11]. Estos trabajos son eficaces en tareas cooperativas estándar (por ejemplo, micromanagement en StarCraft II con hasta 27 agentes controlables en SMAC en el caso de QMIX [7]), pero no abordan de forma directa escenarios logísticos donde el número de agentes supera sistemáticamente al de recursos disponibles, con regeneración dinámica y un análisis cuantitativo de la equidad.

En la Tabla 8 resumimos las diferencias clave entre nuestro enfoque y dos representantes de estas líneas de trabajo: QMIX [7] y CommNet [9].

Cuadro 8: Comparación con enfoques relacionados

Aspecto	**Este trabajo**	**QMIX [7]**	**CommNet [9]**
Coordinación	Implícita (flocking local)	Centralizada (CTDE)	Explícita (broadcast)
Escalabilidad	12 agentes	Hasta 27 agentes	5–10 agentes
Agentes > Recursos	Sí (Expert mode)	No reportado	No reportado
Regeneración	Sí (logística)	No	No
Equidad (Gini)	Sí	No	No

En resumen, QMIX [7] prioriza la factorización monotónica de la función de valor global mediante una red de mezcla centralizada, lo que permite escalar a decenas de agentes en dominios de combate pero sin incorporar un mecanismo de comunicación explícito ni métricas de equidad. CommNet [9] introduce un canal de comunicación diferenciable entre agentes controlado por una red centralizada, pero se ha evaluado en equipos de tamaño moderado (del orden de 5 agentes) y en tareas sin recursos que se regeneren dinámicamente. Trabajos de *swarm RL* con observaciones locales e interacciones implícitas [10] se centran en tareas como *rendezvous* o formación, de nuevo sin escasez explícita de recursos ni análisis de equidad.

Contribución principal: Este trabajo demuestra coordinación emergente basada en reglas locales de tipo *flocking* en escenarios cooperativos con escasez explícita ($N_{\text{agentes}} > N_{\text{recursos}}$), regeneración dinámica de recursos y evaluación cuantitativa de la equidad entre agentes mediante el índice de Gini, utilizando un marco estándar de MARL (PettingZoo [6] + PPO [2, 3]). Esta combinación de características representa un área poco explorada en la literatura de MARL.

6.6. Hallazgos Principales Validados

Coordinación sin comunicación explícita:

- **VALIDADO:** Los agentes logran 87 % (Easy) a 37 % (Expert) eficiencia sin comunicación explícita, demostrando que la coordinación emergente es posible mediante observaciones locales y flocking.

Sinergia flocking + forrajeo:

- **VALIDADO:** Los experimentos de ablación muestran degradación significativa al remover cualquier componente, confirmando que ambos son esenciales.

Escalabilidad a escasez extrema:

- **VALIDADO:** Incluso con agentes > parches (ratio 1.20), el sistema mantiene 37 % eficiencia, evitando el colapso esperado en condiciones de escasez extrema.

Beneficio de memoria (LSTM):

- **VALIDADO:** La arquitectura LSTM permite a los agentes mantener memoria de parches visitados, evitando re-visitas ineficientes a zonas agotadas.

7. Conclusiones y Trabajo Futuro

7.1. Contribuciones Principales

Este proyecto realizó las siguientes contribuciones al campo de aprendizaje por refuerzo multiagente:

1. Coordinación en Escasez Extrema: Demostración exitosa de coordinación emergente donde agentes superan numéricamente a recursos disponibles (Expert mode: ratio 1.20), manteniendo 37 % eficiencia bajo condiciones extremas, un problema con limitada atención en la literatura existente.

2. Diseño de Recompensa Multiobjetivo: Implementación de una función de recompensa que integra exitosamente comportamientos de flocking (cohesión, alineación, separación) con objetivos de forrajeo, demostrando que ambos componentes son esenciales para lograr coordinación efectiva en todos los niveles de dificultad.

3. Evaluación Sistemática con Métricas de Equidad: Incorporación de métricas de equidad (coeficiente de Gini) en la evaluación de forrajeo multiagente, con análisis sobre 400 episodios (100 por nivel) que permite cuantificar el trade-off entre eficiencia y distribución justa de recursos.

4. Implementación Reproducible Open-Source: Entorno PettingZoo completo con regeneración logística, configuraciones modulares por nivel de dificultad, scripts de entrenamiento y evaluación, y código disponible públicamente para facilitar la reproducibilidad y extensión del trabajo.

7.2. Trabajo Futuro

Extensiones Técnicas:

- Incorporar comunicación aprendida (CommNet) para potencialmente mejorar eficiencia en modos difíciles.
- Escalar a 20+ agentes usando Graph Neural Networks.
- Explorar heterogeneidad de agentes con capacidades diferenciadas.
- Entornos con obstáculos, terreno complejo, y parches móviles.

Aplicaciones Prácticas:

- **Robótica de enjambre:** Drones para búsqueda/rescate, robots agrícolas.
- **Gestión de recursos:** Modelado de pesca sostenible, optimización de pastoreo.
- **Sistemas distribuidos:** Balanceo de carga, coordinación de vehículos autónomos.

7.3. Reflexión Final

Este proyecto demuestra que agentes autónomos pueden desarrollar **coordinación cooperativa sofisticada sin comunicación explícita**, mediante integración de principios bioinspirados con objetivos de optimización. Los resultados validan escalabilidad a escasez extrema con implicaciones transferibles a robótica, gestión de recursos naturales, y sistemas distribuidos reales.

8. Reproducibilidad

8.1. Repositorio de Código

Todo el código, modelos entrenados, configuraciones y resultados están disponibles públicamente:

GitHub: `https://github.com/enriquegomeztagle/multi-agent-flocking-foraging-rl`

8.2. Estructura del Repositorio

Nota: Se muestra únicamente la estructura de archivos esenciales del proyecto.

```
multi-agent-flocking-foraging-rl/
+-- env/                          # Entorno personalizado
|   +-- flockforage_parallel.py   # Implementación PettingZoo
|   +-- gym_wrapper.py                  # Wrapper Gym para SB3
|   +-- physics.py                      # Dinámica de agentes
|   +-- patches.py                      # Parches con regeneración
|   \-- render.py                       # Visualización
-- train/                          # Scripts de entrenamiento
|   +-- train_ppo.py                    # Pipeline principal
|   +-- eval_*.py                       # Evaluación por nivel
|   \-- eval_baseline_boids.py          # Baseline clásico Boids
-- metrics/                        # Cálculo de métricas
|   +-- fairness.py                     # Coeficiente de Gini
|   +-- flocking.py                     # Polarización, cohesión
|   \-- sustainability.py               # Depleción de recursos
-- configs/                        # Configuraciones YAML
|   +-- env_easy.yaml
|   +-- env_medium.yaml
```

```
|   +-- env_hard.yaml
|   \-- env_expert.yaml
-- models/                          # Modelos entrenados
-- results/                         # JSONs de evaluación
\-- dashboard.py                    # Dashboard Streamlit
```

8.3. Dependencias y Versiones

Entorno de Ejecución:

- **Python:** 3.10+
- **OS:** macOS 12+ / Ubuntu 20.04+ / Windows 10+
- **Hardware Recomendado:** GPU NVIDIA (8GB+ VRAM) para entrenamiento, CPU suficiente para evaluación

Librerías Principales:

```
torch>=2.2.0                 # PyTorch (backend neuronal)
stable-baselines3>=2.3.0     # PPO base
sb3-contrib>=2.3.0           # RecurrentPPO con LSTM
pettingzoo>=1.24.3           # Entornos multiagente
gymnasium>=0.29.1            # API de entornos
numpy>=1.26.0
scipy>=1.11.0
pyyaml>=6.0.0
streamlit>=1.28.0            # Dashboard interactivo
plotly>=5.17.0               # Visualizaciones
pandas>=2.2.0
```

8.4. Comandos de Reproducción

Entrenar Modelo (ejemplo Easy mode):

```
PYTHONPATH=. python train/train_ppo.py \
    --config configs/env_easy.yaml \
    --output models/ppo_easy \
    --timesteps 1500000 \
    --save-freq 100000
```

Evaluar Modelo Entrenado:

```
PYTHONPATH=. python train/eval_easy.py \
    --model models/ppo_easy/final_model \
    --episodes 100
```

Visualizar Resultados (Dashboard):

```
streamlit run dashboard.py
```

8.5. Seeds para Reproducibilidad

Todos los experimentos usan seeds fijas:

- **Entrenamiento:** seed = 42
- **Evaluación:** seeds = $\{0, 42, 84, ..., 4158\}$ (100 valores espaciados por 42)

8.6. Troubleshooting

Problema: CUDA out of memory

- **Solución:** Reducir `n_envs` de 4 a 2 en `train_ppo.py`

Problema: Colapso de performance durante entrenamiento

- **Solución:** Usar checkpoints previos (guardados cada 100k steps)

Problema: Importación falla (ModuleNotFoundError)

- **Solución:** Usar `PYTHONPATH=.` antes de comandos Python

9. Referencias Bibliográficas

Referencias

[1] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '87)*, pp. 25–34, 1987. DOI http://doi.acm.org/10.1145/37401.37406

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017. DOI https://doi.org/10.48550/arXiv.1707.06347

[3] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. DOI https://doi.org/10.48550/arXiv.1506.02438

[4] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI https://doi.org/10.1162/neco.1997.9.8.1735

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018. https://mitpress.mit.edu/9780262039246/reinforcement-learning/

[6] J. Terry, A. Y. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, N. L. Williams, Y. Lokesh, R. Horsch, and P. Ravi, "PettingZoo: Gym for multi-agent reinforcement learning," in *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 2021. DOI https://doi.org/10.48550/arXiv.2009.14471

[7] T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 4295–4304, 2018. DOI https://doi.org/10.48550/arXiv.1803.11485

[8] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative multi-agent games," in *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022. DOI https://doi.org/10.48550/arXiv.2103.01955

[9] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning multiagent communication with backpropagation," in *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, pp. 2244–2252, 2016. DOI https://doi.org/10.48550/arXiv.1605.07736

[10] M. Hüttenrauch, A. Šošić, and G. Neumann, "Guided deep reinforcement learning for swarm systems," in *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pp. 1551–1559, 2019. DOI https://doi.org/10.48550/arXiv.1709.06011

[11] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pp. 1495–1502, 2018. DOI https://doi.org/10.48550/arXiv.1703.04908

[12] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004. https://mitpress.mit.edu/9780262042192/ant-colony-optimization/

[13] I. D. Couzin, J. Krause, R. James, G. D. Ruxton, and N. R. Franks, "Collective memory and spatial sorting in animal groups," *Journal of Theoretical Biology*, vol. 218, no. 1, pp. 1–11, 2002. DOI https://doi.org/10.1006/jtbi.2002.3065

10. Apéndice

10.1. Repositorio de GitHub

`https://github.com/enriquegomeztagle/multi-agent-flocking-foraging-rl.git`