T8 INTELIGENCIA DE ENJAMBRE

AGENTES Y SISTEMAS MULTIAGENTE

Fidel Aznar Gregori

Departamento de Ciencia de la Computación e Inteligencia Artificial.
Universidad de Alicante

INDICE

- Definición y características
- Algoritmos de inteligencia de enjambre
  - PSO
  - ACO
  - BCA
- Metalearning
- ABM + SI, un ejemplo: flocking
- Desafios

INTELIGENCIA DE ENJAMBRE (I)

A paradigm that considers collective intelligence as a behavior that emerges through the interaction and cooperation of large numbers of lesser intelligent agents.

INTELIGENCIA DE ENJAMBRE (II)

- Son un modelo alternativo, inspirado en la *vida artificial*
- Los agentes individuales pueden ser *no-cognitivos* pero complejos
- La interacción emerge de la interacción

DEFINICIÓN (I)

- Entenderemos por enjambre la **auto-organización útil de multiples entidades a través de interacciones locales**.
- Se ha definido también como **un mecanismo para solucionar problemas inspirado en los comportamientos colectivos de animales sociales**.

DEFINICIÓN (II)

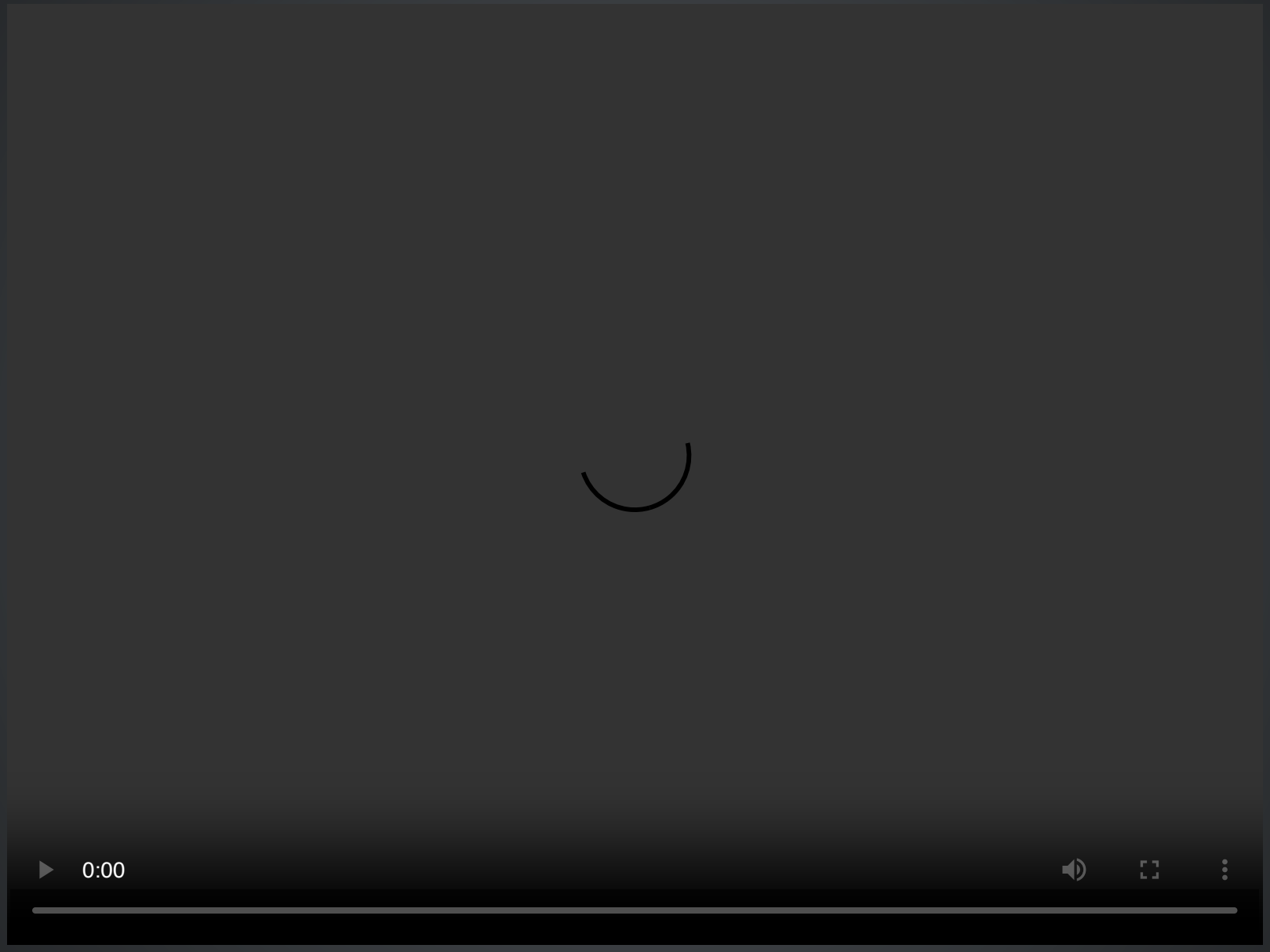
- **Útil**. Enfatiza su uso a nivel de ingeniería
- **Organización**. Reducir la entropía del sistema
- **Auto-organización**.
  - Proceso que reduce la entropía sin ninguna interacción externa.
  - No se requiere que sea el resultado de procesos reactivos, pueden ser deliberativos (equipos).
- **Emergencia**. Subtipo de *auto-organización* que describe la aparición de estructuras de alto nivel, no presentes en los componentes de bajo nivel.

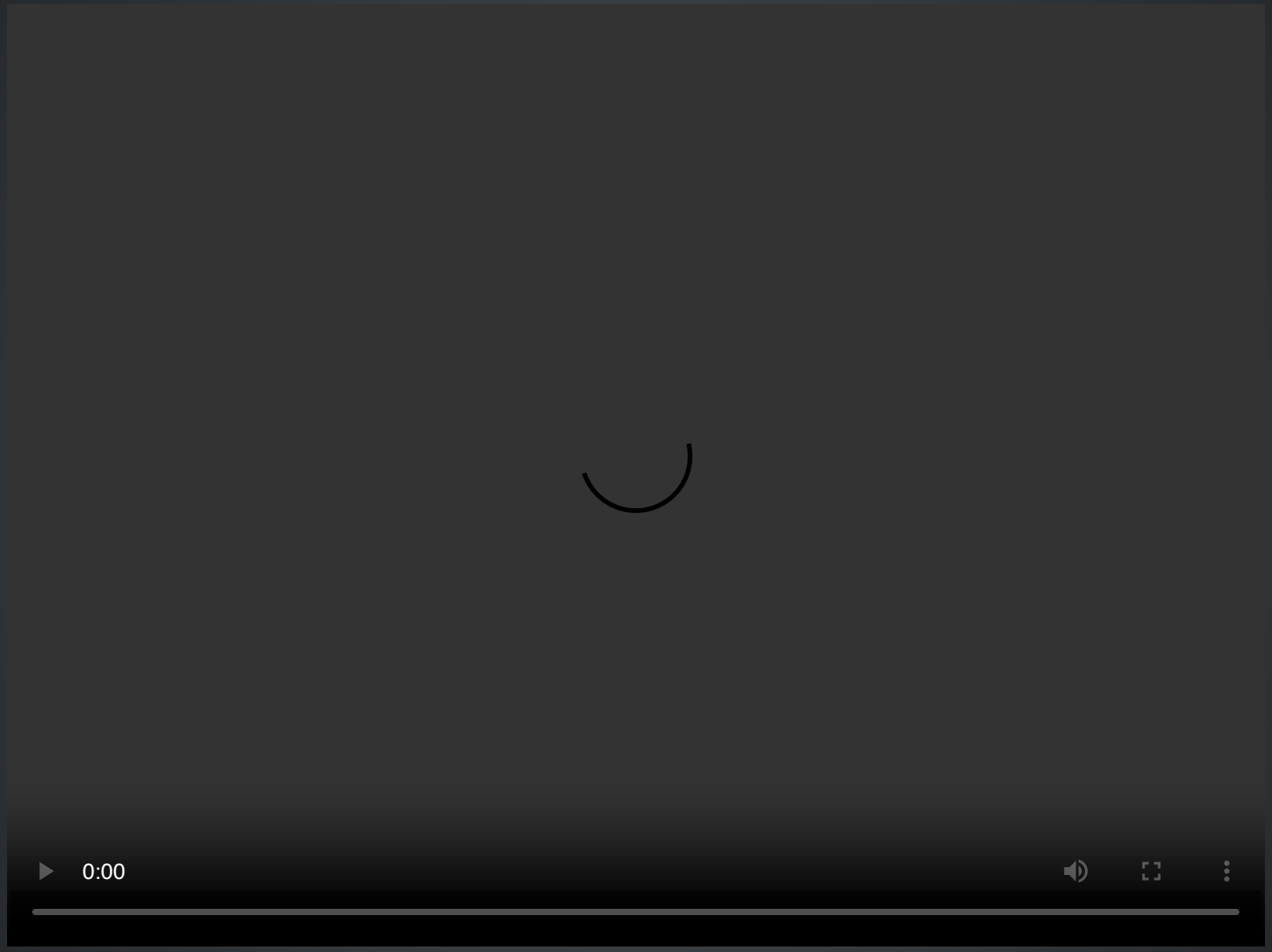
EJEMPLOS EN LA NATURALEZA (I)

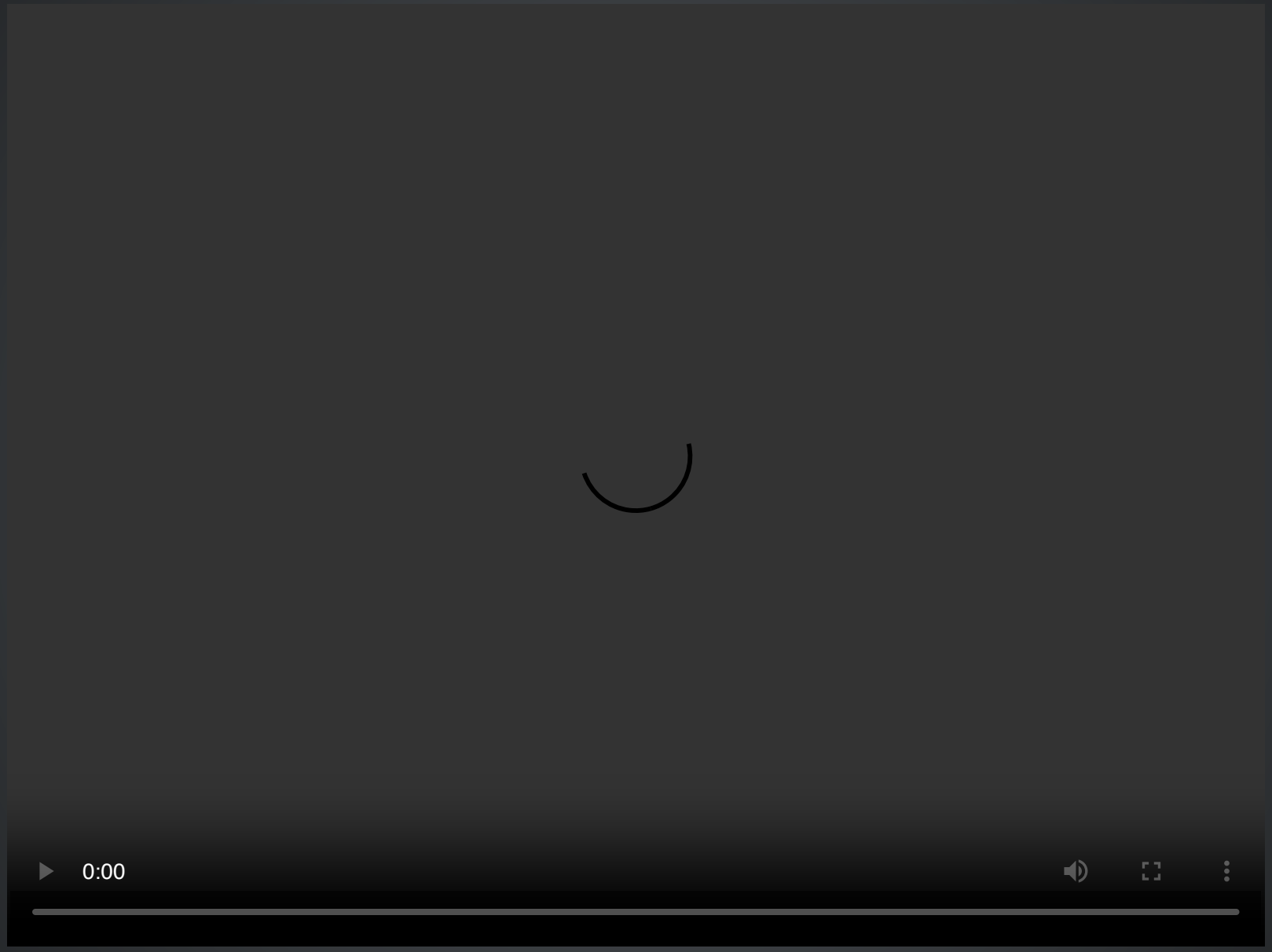
Comportamiento	Entidades
Termoregulación	Abejas
Asignación de tareas	Termitas
Sincronización	Libelulas
Construcción de redes	Arañas
Flocking	Pájaros
Rodear a la presa	Lobos

EJEMPLOS EN LA NATURALEZA (II)

Comportamiento	Entidades
Generación de patrones	Bacterias
Formación de caminos	Hormigas
Ordenación del nido	Hormigas
Transporte cooperativo	Hormigas
Elección fuentes comida	Hormigas


0:00


0:00


0:00

CARACTERÍSTICAS DESTACADAS

- **Robustez**.
  - Redundancia del sistema
  - Coordinación descentralizada
  - Simplicidad de los individuos
  - Mutiplicidad de sensores
- **Flexibilidad**. Capacidad de adaptarse a los cambios
- **Escalabilidad**. Habilidad para dar soporte a un número mayor de individuos

ALGORITMOS BASADOS EN INTELIGENCIA DE ENJAMBRE

PARTICLE SWARM OPTIMIZATION (PSO)

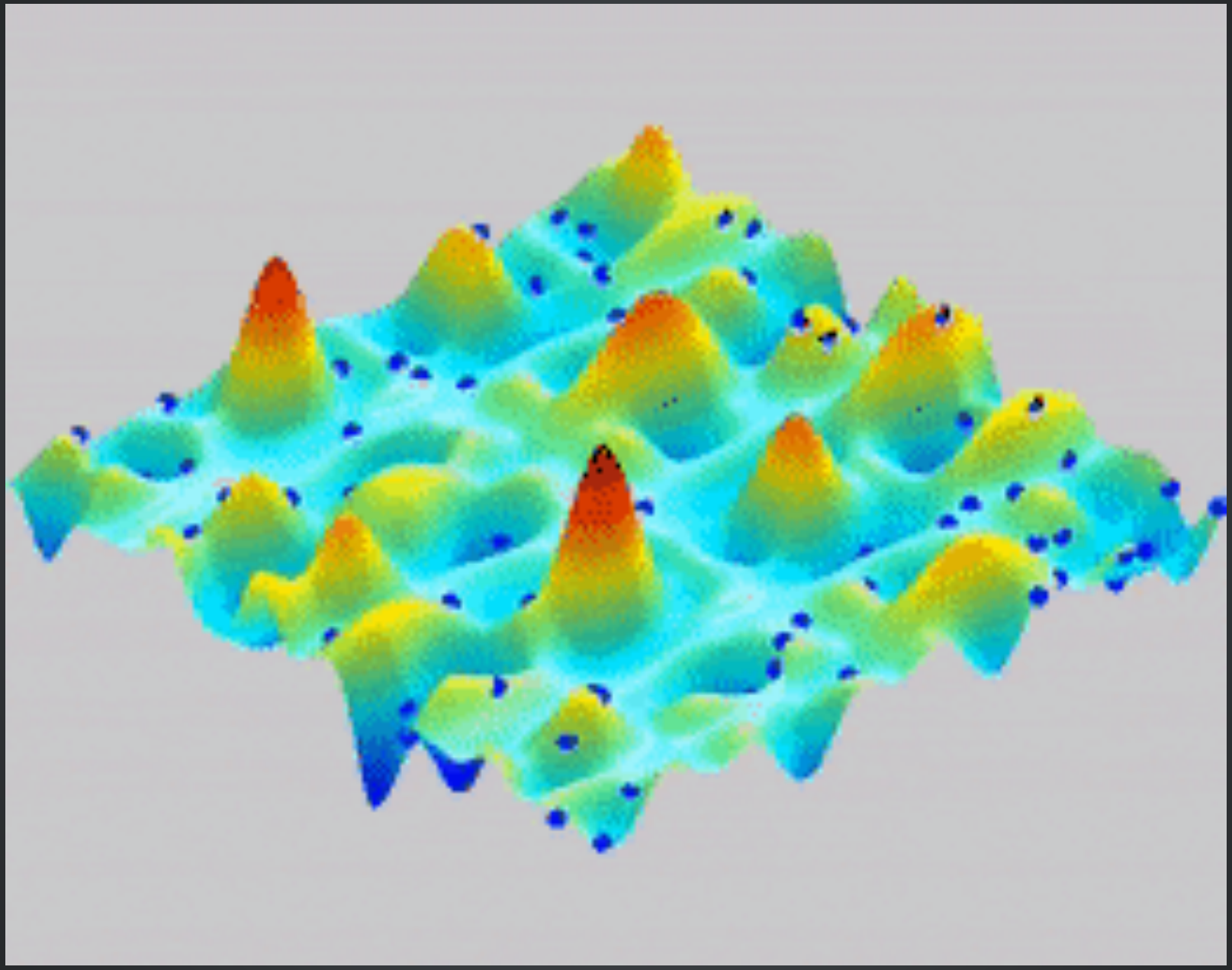
PARTICLE SWARM OPTIMIZATION

- Se inspira en el comportamiento social de búsqueda de alimento:
  - Algunos animales pájaros y los bancos de peces
- Las partículas del enjambre *vuelan* a través de un entorno
- Siguen a los individuos más aptos
- En general, orientando su movimiento hacia zonas históricamente buenas de su entorno.

Input: ProblemSize, $Population_{size}$
Output: P_{g_best}

```
1  Population ← ∅;
2  P_g_best ← ∅;
3  for i = 1 to Population_size do
4  |   P_velocity ← RandomVelocity();
5  |   P_position ← RandomPosition(Population_size);
6  |   P_p_best ← P_position;
7  |   if Cost(P_p_best) ≤ Cost(P_g_best) then
8  |   |   P_g_best ← P_p_best;
9  |   end
10 end
```

```
11 while ¬StopCondition() do
12 |   foreach P ∈ Population do
13 |   |   P_velocity ← UpdateVelocity(P_velocity, P_g_best, P_p_best);
14 |   |   P_position ← UpdatePosition(P_position, P_velocity);
15 |   |   if Cost(P_position) ≤ Cost(P_p_best) then
16 |   |   |   P_p_best ← P_position;
17 |   |   |   if Cost(P_p_best) ≤ Cost(P_g_best) then
18 |   |   |   |   P_g_best ← P_p_best;
19 |   |   |   end
20 |   |   end
21 |   end
22 end
23 return P_g_best;
```



ANT COLONY OPTIMIZATION

ANT COLONY OPTIMIZATION

- Inspirado en el comportamiento de alimentación de las hormigas (foraging)
- Concretamente en la comunicación de feromonas entre hormigas:
  - Camino bueno entre la colonia y una fuente de alimento en un entorno marcado.
  - Este mecanismo se denomina estigmergia.

FUNCIONAMIENTO

- Inicialmente las hormigas vagan al azar
- Una vez localizado el alimento, la hormiga empieza a depositar feromona en el entorno.
- Se realizan numerosos viajes entre la comida y la colonia y, si se sigue la misma ruta que lleva a la comida, se deposita más feromona.
- Las feromonas se descomponen
- Retroalimentación positiva dirige cada vez más hormigas hacia caminos productivos que, a su vez, se perfeccionan con el uso.

```
Input: ProblemSize, Population_size, m, ρ, α, β
Output: P_best
P_best ← CreateHeuristicSolution(ProblemSize);
Pbest_cost ← Cost(S_h);
Pheromone ← InitializePheromone(Pbest_cost);
while ¬StopCondition() do
    Candidates ← ∅;
    for i = 1 to m do
        S_i ← ProbabilisticStepwiseConstruction(Pheromone,
        ProblemSize, α, β);
        Si_cost ← Cost(S_i);
        if Si_cost ≤ Pbest_cost then
            Pbest_cost ← Si_cost;
            P_best ← S_i;
        end
        Candidates ← S_i;
    end
    DecayPheromone(Pheromone, ρ);
    foreach S_i ∈ Candidates do
        UpdatePheromone(Pheromone, S_i, Si_cost);
    end
end
return P_best;
```

EJEMPLO GAMA

BEE COLONY OPTIMIZATION

- Se inspira en el comportamiento de búsqueda de alimento de las abejas melíferas.
- Las abejas de la miel recolectan néctar en vastas zonas alrededor de su colmena (más de 10 kilómetros).
- La colonia envía abejas a recoger néctar en función de la cantidad de alimento en cada zona.
- Las abejas se comunican entre sí en la colmena mediante una danza que informa a las demás abejas de la colmena de la dirección, la distancia y la calidad del alimento

FUNCIONAMIENTO

- Las abejas recogen néctar de flores
- La colmena envía exploradoras que localizan zonas de flores, regresan a la colmena e informan a otras abejas sobre la aptitud y la ubicación de una fuente de alimento
- La exploradora regresa a las flores con abejas seguidoras.
- Un pequeño número de exploradoras sigue buscando nuevas zonas, mientras que las abejas que regresan de otras zonas de flores siguen comunicando la calidad de la zona.

```
# Ejemplo de parámetros
tamaño_colonia = 10
num_iteraciones = 100
num_abejas_exploradoras = 5
num_abejas_observadoras = tamaño_colonia -
num_abejas_exploradoras limite_evaluaciones = 500

# Función objetivo a ser optimizada
def funcion_objetivo(solucion) : ...

mejor_solucion_global = null
mejor_valor_global = infinito

for abeja in colonia:
    generar_solucion_aleatoria()
```

```
# Ciclo principal del algoritmo
for i in range(num_iteraciones):

    # 1) Fase de exploración
    # 2) Fase de observación
    # 3) Actualizar posición de las abejas exploradoras con mejor
    # 4) Actualizar posición de las abejas observadoras con mejor
    # 5) Actualizar la mejor solución global
    # 6) Verificar criterio de parada

# Resultados finales
return mejor_solucion_global, mejor_valor_global
```

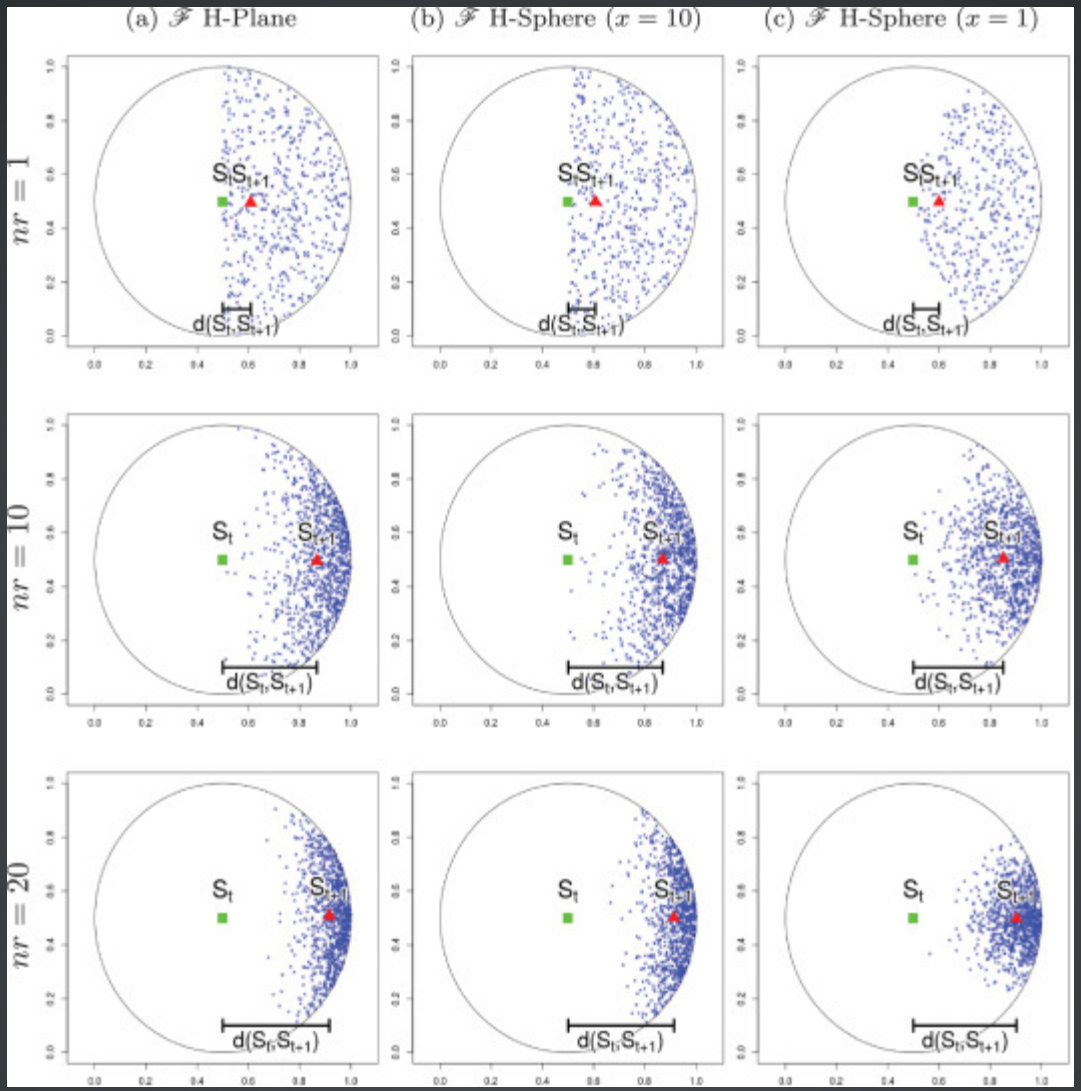
```
# Fase de exploración
for abeja in exploradora:
    generar_solucion_vecina()
    evaluar_solucion()
    if valor_objetivo < mejor_valor_local:
        mejor_solucion_local = solucion_actual
        mejor_valor_local = valor_objetivo
```

```
# Fase de observación
    for abeja in observadora:
        seleccionar_abeja_fuente()
        generar_solucion_vecina()
        evaluar_solucion()
        if valor_objetivo < mejor_valor_local:
            mejor_solucion_local = solucion_actual
            mejor_valor_local = valor_objetivo
```

```
# Actualizar posición de las abejas exploradoras
# con mejores soluciones locales
    for abeja in exploradora:
        actualizar_posicion_exploradora()

    # Actualizar posición de las abejas observadoras con mejores
    # soluciones locales
    for abeja in observadora:
        actualizar_posicion_observadora()

    # Actualizar la mejor solución global
    if mejor_valor_local < mejor_valor_global:
        mejor_solucion_global = mejor_solucion_local
        mejor_valor_global = mejor_valor_local
```



An analysis of the search mechanisms of the bees algorithm - Luca Baronty et al.

METALEARNING

(EN INTELIGENCIA DE ENJAMBRE)

¿QUÉ ES EL META-APRENDIZAJE (MA)?

- Subcampo del Aprendizaje Automático (AA) que busca diseñar modelos que puedan aprender de manera más rápida y eficiente.
- El objetivo es crear sistemas capaces de adaptarse a nuevas tareas con pocos ejemplos de entrenamiento, en lugar de requerir grandes conjuntos de datos, como es común en los enfoques tradicionales de AA.

¿PARA QUÉ SIRVE EL META-APRENDIZAJE?

- Permite que los sistemas de AA se adapten a nuevas tareas o dominios que no se habían considerado durante su entrenamiento original.
- Esto puede ser útil en muchos campos, como la medicina, donde las tareas pueden variar considerablemente y los datos pueden ser escasos.
- En Inteligencia colectiva muy útil para que un comportamiento sea aplicable a distintas tareas sin necesidad de reentrenar.

MA E INTELIGENCIA DE ENJAMBRE (I)

- **Aprendizaje más rápido:** Aprender más rápidamente de la experiencia. En lugar de tener que aprender de cero cada vez, pueden aprovechar los conocimientos adquiridos en tareas anteriores para acelerar el aprendizaje en nuevas tareas.
- **Adaptación flexible:** Capacidad para adaptarse a nuevas situaciones o tareas. Esto es especialmente útil en la inteligencia de enjambre, donde los agentes a menudo deben trabajar juntos para abordar problemas nuevos o desconocidos.

MA E INTELIGENCIA DE ENJAMBRE (II)

- **Generalización mejorada:** El meta-aprendizaje puede mejorar la capacidad de los sistemas de inteligencia colectiva para generalizar a partir de la experiencia, permitiéndoles manejar una gama más amplia de tareas o problemas.
- **Descentralización:** En la inteligencia colectiva, cada agente individual puede llevar a cabo su propio proceso de meta-aprendizaje, lo que permite un aprendizaje más descentralizado y distribuido.

FORTALEZAS Y DEBILIDADES

- Fortalezas:
  - Adaptabilidad: Puede manejar una gama de tareas diferentes con pocos datos de train.
  - Eficiencia: Aprende rápidamente nuevas tareas.
- Debilidades:
  - Overfitting
  - Complejidad: Más complejos y difíciles de entender que AA convencionales.

¿CÓMO FUNCIONA EL META-APRENDIZAJE?

- En términos generales, el meta-aprendizaje implica un proceso de dos niveles:
    1. Nivel de tarea: Se realizan múltiples tareas de aprendizaje, cada una con su propio conjunto de datos de entrenamiento y prueba.
    2. Nivel de meta: Se utiliza el rendimiento en las tareas de aprendizaje para actualizar el modelo de meta-aprendizaje.
- En cada iteración, el modelo MA intenta mejorar su capacidad para adaptarse a nuevas tareas.

APLICACIONES DEL META-APRENDIZAJE

- Aprendizaje por refuerzo: En situaciones donde hay una recompensa diferida, el meta-aprendizaje puede ayudar a aprender estrategias de toma de decisiones.
- Procesamiento del lenguaje natural: En tareas como la traducción automática, el meta-aprendizaje puede ayudar a adaptarse a nuevos idiomas con pocos datos.
- Clasificación de imágenes: El meta-aprendizaje puede aprender a clasificar nuevas categorías de imágenes con pocos ejemplos.

EJEMPLOS DE META-APRENDIZAJE

- MAML (Model-Agnostic Meta-Learning): Un algoritmo que busca encontrar un punto de inicialización de los parámetros del modelo que sea bueno para aprender rápidamente una variedad de tareas.
- Meta-SGD (Meta Stochastic Gradient Descent): Un algoritmo que no solo aprende los parámetros iniciales, sino también los hiperparámetros de un optimizador de descenso de gradiente estocástico.
- Meta-aprendizaje Multiagente

MAML (I)

- Método de meta-aprendizaje que tiene como objetivo encontrar una inicialización de parámetros que se pueda afinar rápidamente para una variedad de tareas.
- El algoritmo ajusta los parámetros del modelo en la dirección que reduce el error en la **mayor cantidad posible de tareas**.

MAML (II)

```
theta = ... # Inicializar parámetros del modelo
num_tasks_per_batch = # N. de tareas por lote

for i in range(num_iter):
    task_gradients = []
    for task in range(num_tasks_per_batch):
        # Muestrear una tarea Ti
        Ti = sample_task()
        # Gradientes error de la tarea Ti con respecto a theta
        theta_prime = compute_gradients(Ti, theta)
        task_gradients.append(theta_prime)

    # Actualizar theta usando el promedio de los gradientes
    # de todas las tareas Ti
    theta = update_theta(theta, task_gradients)
```

META-APRENDIZAJE MULTIAGENTE (I)

Enfoque que combina conceptos de meta-aprendizaje y aprendizaje multi-agente:

1. **Agentes individuales como aprendices**: agente/aprendiz, capaz de aprender y adaptarse a su entorno. Cada agente tiene la capacidad de observar su entorno, tomar decisiones y aprender de la retroalimentación que recibe.
2. **Aprendizaje a través de la experiencia**: Los agentes aprenden a través de la experiencia, adaptando sus políticas de acción basadas en la retroalimentación que reciben del entorno.

META-APRENDIZAJE MULTIAGENTE (II)

3. **Meta-aprendizaje para adaptación rápida**:
   - Agentes usan meta-aprendizaje: no sólo aprenden a realizar tareas específicas, sino que también aprenden cómo el proceso de aprendizaje puede ser mejorado (uso de MAML, train en múltiples tareas).
   - Esto puede permitir a los agentes adaptarse rápidamente a nuevas tareas o situaciones.
4. **Aprendizaje cooperativo y competitivo**: Los agentes pueden aprender a mejorar sus propias políticas de acción, así como a predecir y responder a las acciones de otros agentes.

META-APRENDIZAJE MULTIAGENTE (III)

5. **Comunicación y coordinación**: Los agentes también pueden aprender a comunicarse y coordinarse entre sí.
6. **Aprendizaje distribuido**: El meta-aprendizaje multi-agente es inherentemente un proceso distribuido, donde cada agente realiza su propio aprendizaje y adaptación. Esto puede permitir un aprendizaje más escalable y flexible en comparación con los enfoques centralizados.

META-APRENDIZAJE MULTIAGENTE (IV)

```
for i in range(num_iter):
    for agent in agents:
        task_gradients = []
        for task in range(num_tasks_per_batch):
          Ti = sample_task() # Muestrear una tarea Ti
          # Error de la tarea Ti con respecto a los parámetros
          # del agente
          theta_prime = compute_gradients(Ti, agent.parameters)
          task_gradients.append(theta_prime)

        # Actualizar los parámetros de A. usando el promedio
        # de los gradientes de todas las tareas Ti
        agent.parameters =
        update_parameters(agent.parameters, task_gradients)
```

DESAFÍOS FUTUROS DEL META-APRENDIZAJE

- Mejorar la eficiencia: Los algoritmos de meta-aprendizaje pueden ser computacionalmente costosos y requerir grandes cantidades de memoria.
- Reducir el sobreajuste: prevenir el sobreajuste para que pueda generalizar a nuevas tareas.
- Comprensibilidad: difíciles de interpretar, lo que plantea desafíos para la comprensibilidad y la confianza.

SWARM INTELLIGENCE EN ABM (I)

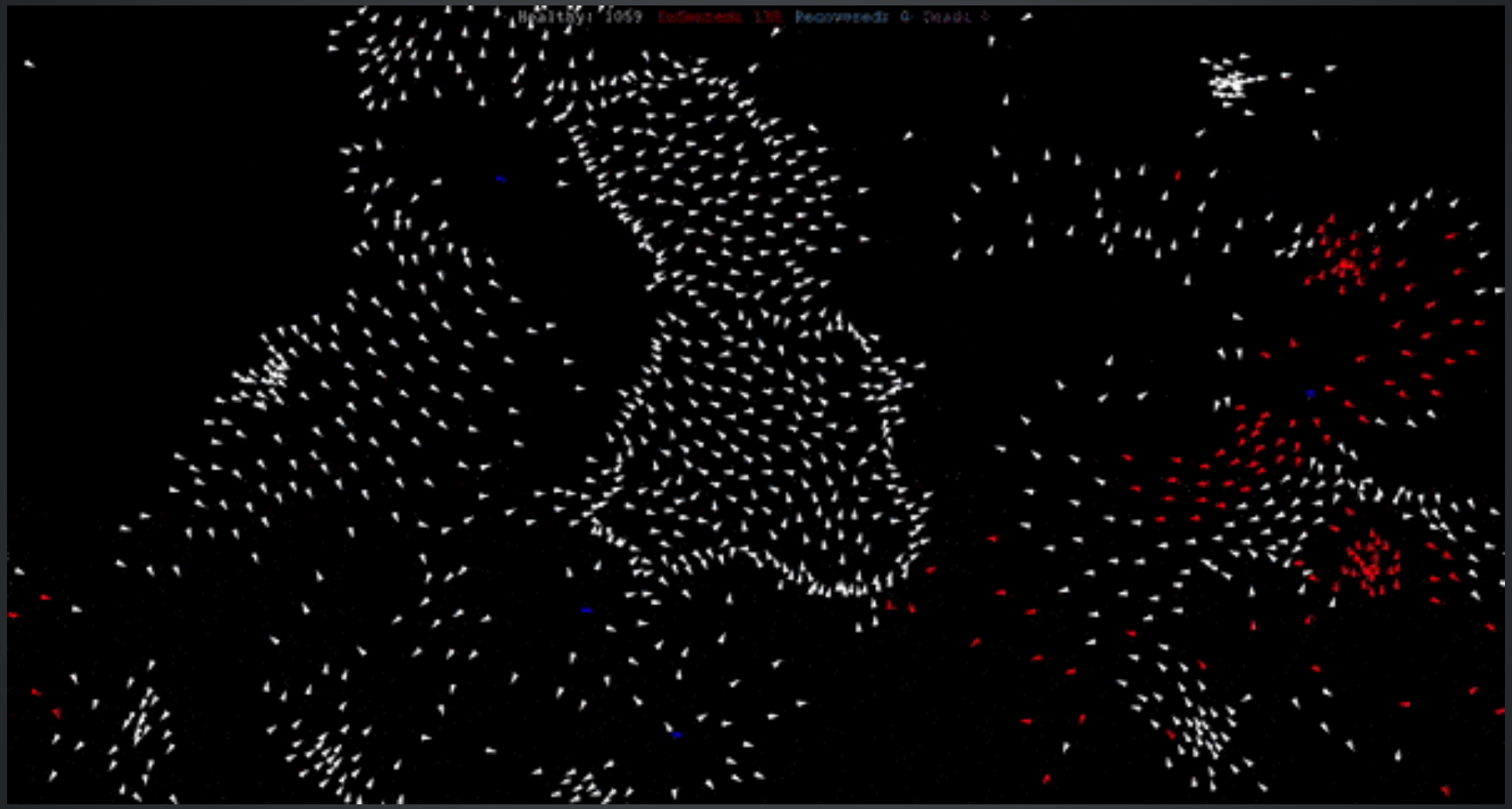
- Agentes se comportan inspiración natural.
- Diseñar agentes virtuales con reglas inspiradas en enjambres naturales.
- ¿Cómo funciona?
  - Agentes interactúan con entorno y otros agentes según reglas específicas: comunicación entre agentes, la cooperación, la atracción hacia ciertos objetivos o la evitación de obstáculos...
  - Emergen patrones de comportamiento colectivo en el sistema.

SWARM INTELLIGENCE EN ABM (II)

- Ejemplo: imagina un escenario donde los agentes representan automóviles autónomos:
  - Siguen reglas de tráfico y evitan colisiones.
  - A pesar de que cada automóvil sigue reglas simples, el flujo de tráfico resultante es complejo y autoorganizado
  - Similar a cómo los pájaros vuelan en formación en la naturaleza.

EJEMPLO DE ABM: FLOCKING

- Introducción al Flocking:
  - Fenómeno natural de comportamiento colectivo.
  - Grupos de entidades que se mueven de manera coordinada.
  - Ejemplos en la naturaleza.
- Importancia en la computación y simulación:
  - Inspiración para algoritmos de inteligencia artificial.
  - Aplicaciones en robótica y simulación de multitudes.

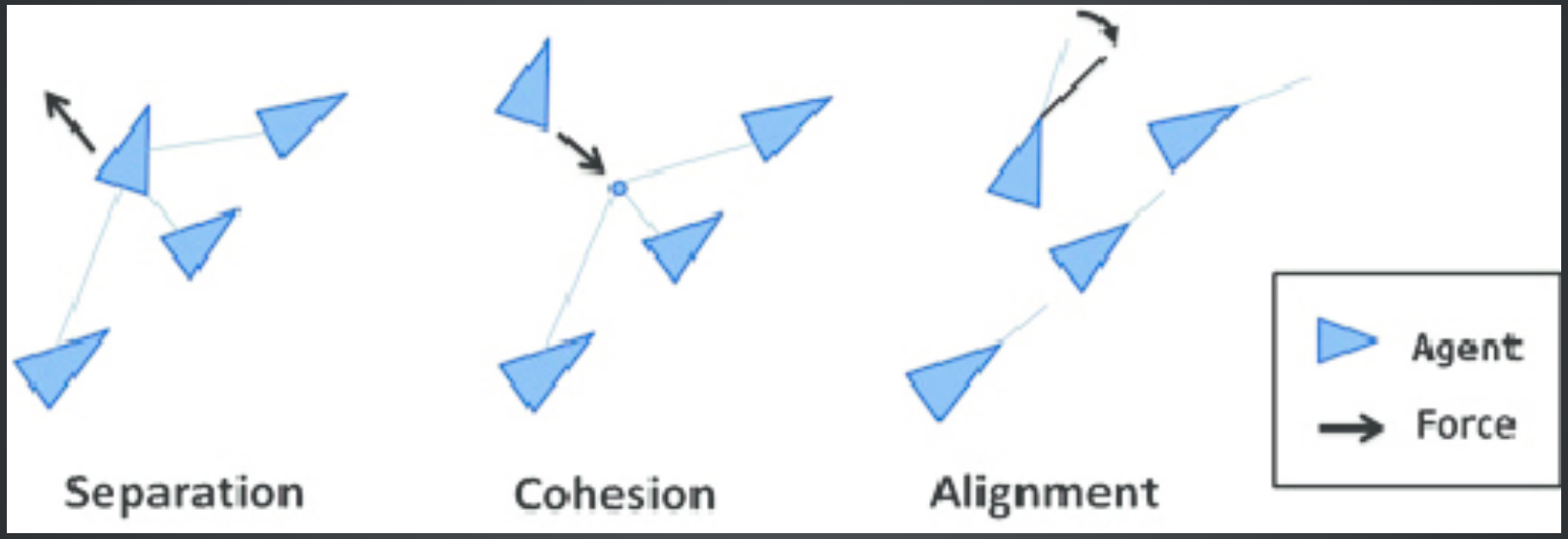

Healthy: 3059 Infected: 138 Recovered: 0 Dead: 0

COMPONENTES DEL FLOCKING (I)

- Individuos (agentes):
  - Cada entidad es un agente con posición y velocidad.
  - Interactúan con otros agentes en su entorno.
- Vecindario y visión:
  - Los agentes tienen un campo de visión limitado.
  - Solo pueden interactuar con agentes dentro de su rango de visión.

COMPONENTES DEL FLOCKING (II)

- Reglas de comportamiento:
  - Regla de **Cohesión**. Atracción hacia el centro del grupo.
  - Regla de **Separación**. Evitar colisiones y superposiciones.
  - Regla de **Alineación**. Imitar la dirección y velocidad de los vecinos.


Separation
Cohesion
Alignment
Agent
Force

REGLA DE COHESIÓN

- Explicación de la regla:
  - Los agentes se sienten atraídos hacia el centro promedio del grupo.
  - Fomenta la cohesión y evita la dispersión.
- Mantener la distancia adecuada:
  - Agentes ajustan velocidad para mantener distancia óptima con otros miembros.
- Ejemplo visual: Bandadas de aves migratorias que vuelan juntas en un patrón unificado.

REGLA DE SEPARACIÓN

- Explicación de la regla:
  - Los agentes evitan colisiones y superposiciones manteniendo una distancia mínima entre ellos.
- Mantener espacio entre los individuos:
  - Si agente se acerca demasiado a otro, ajusta dirección o velocidad para evitar colisión
- Ejemplo visual: Peces en un banco que se esquivan para evitar choques.

REGLA DE ALINEACIÓN

- Explicación de la regla:
  - Los agentes imitan la dirección y velocidad promedio de sus vecinos cercanos.
- Mantener la dirección y velocidad promedio del grupo:
  - Agentes ajustan movimiento para coincidir con el comportamiento colectivo.
- Ejemplo visual: Bandada de pájaros que cambian de dirección en conjunto.

IMPLEMENTACIÓN EN ALGORITMOS

- Algoritmo de Flocking de Reynolds:
  - Creado por Craig Reynolds en 1987.
  - Basado en las tres reglas fundamentales.
  - Actualización de posiciones y velocidades en cada paso de tiempo.
- Impacto de parámetros:
  - La distancia de visión y la velocidad máxima de los agentes influyen en el comportamiento resultante.

APLICACIONES DEL FLOCKING

- Simulación de comportamiento animal:
  - Utilizado en películas y videojuegos para simular manadas y bandadas realistas.
- Robótica móvil autónoma:
  - Robots que se mueven de manera cooperativa en almacenes o entornos desconocidos.
- Tráfico inteligente:
  - Control de semáforos adaptativo y gestión de flujos de tráfico.

DESAFÍOS EN EL FLOCKING

- Gestión de la escala:
  - El comportamiento de Flocking a gran escala puede ser computacionalmente costoso.
- Adaptación a entornos cambiantes:
  - Las reglas deben adaptarse a condiciones variables.
- Evitar la formación de patrones no deseados:
  - Asegurar que el Flocking no conduzca a patrones de comportamiento peligrosos o indeseados.

FLOCKING EN LA VIDA REAL

- Estudio de sistemas biológicos:
  - Investigaciones sobre comportamiento de aves y peces (ecología).
- Implementaciones en la vida cotidiana:
  - Sistemas de control de tráfico.
  - Simulaciones de evacuación desastres.
- Impacto en la ciencia y la tecnología:
  - Avances en robótica colaborativa.
  - IA y aprendizaje automático.

DESAFÍOS ACTUALES

- Introducción a los desafíos que enfrenta la Inteligencia de Enjambre en la actualidad.
- Algunos desafios:
  - Escalabilidad
  - Adaptabilidad
  - Robustez
  - Interfaces H/M

DESAFÍO 1: ESCALABILIDAD

- Problema: Manejar grandes cantidades de agentes en sistemas de enjambres puede ser computacionalmente costoso.
- Desarrollo de algoritmos y técnicas eficientes para abordar la escalabilidad en la simulación y el control de enjambres.

DESAFÍO 2: ADAPTABILIDAD Y COMBINACIÓN DE ESTRATEGIAS

- Problema: Los enjambres deben adaptarse a entornos cambiantes y a condiciones impredecibles.
- Investigación en la mejora de la capacidad de los enjambres para ajustar su comportamiento en tiempo real.

DESAFÍO 3: ROBUSTEZ

- Problema: Los enjambres deben mantener su funcionamiento incluso en presencia de fallas individuales o perturbaciones externas.
- Diseño de enjambres que puedan recuperarse y mantener la cohesión ante adversidades.

DESAFÍO 4: INTERACCIÓN HUMANO-ENJAMBRE

- Problema: Integrar enjambres en aplicaciones donde interactúan con humanos, como la robótica social.
- Consideración de la seguridad, ética y comunicación efectiva en entornos mixtos humano-enjambre.