

T7. APRENDIZAJE EN



AGENTES Y SISTEMAS MULTIAGENTE

Fidel Aznar Gregori

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

ÍNDICE

- Introducción
- Retos en el aprendizaje de SMA
- Adaptando RL a MA-RL (Deep RL en SMA)
- Métodos evolutivos
- Neuroevolución

INTRODUCCIÓN (I)

- Aprendizaje con un único agente:
 - Aprender de su experiencia
 - Generalmente mapeado entre el estado interno del agente y su acción (E/S)
 - ¿Cómo se puede construir ese mapeado?
 - Aprendizaje por refuerzo
 - NN
 - Algoritmos evolutivos
 - Modelos bioinspirados...

INTRODUCCIÓN (II)

- Extensión a Sistemas Multiagente requiere:
 - ¿Cómo tiene en cuenta un agente las acciones del resto de agentes?
 - ¿Cómo seleccionan los agentes las acciones para maximizar el resultado del equipo?
- Por tanto afecta a:
 - La percepción del agente dentro del grupo
 - La elección de acciones teniendo en cuenta la interacción entre el grupo

RETOS EN EL APRENDIZAJE DE SMA (I)

- La extensión de algoritmos de aprendizaje a SMA conlleva muchos desafíos
- Pensemos en el *aprendizaje por refuerzo*:
 - Se asume que el agente interactúa con un entorno *estático*
 - Con SMA múltiples agentes pueden cambiar el entorno y modificar el estado de otros agentes de manera no predecible.
 - ¡No podemos asegurar la convergencia!

RETOS EN EL APRENDIZAJE DE SMA (II)

- ¿Y si tratamos las acciones del resto de agentes como parte del entorno?
 - Nos permitiría modelarlo para problemas simples
 - No es útil para problemas complejos

ESTADOS, ACCIONES Y ESPACIO DE

ESTADOS

- El aprendizaje se puede ver como un problema de búsqueda de todas las soluciones posibles
 - $A > n^o$ de variables (agentes, estados, acciones,...) $>>$ exponencialmente la complejidad el problema
 - El espacio de estados aumenta exponencial con el n^o de agentes
 - También $>$ espacio de acciones y $>>$ ¡el espacio de acciones conjuntas!

CREDIT ASSIGNMENT PROBLEM EN SMA (I)

¿CUANTO HA CONTRIBUIDO CADA AGENTE DEL EQUIPO A LA SOLUCIÓN?

- Imaginemos un partido de fútbol:
 - Cada jugador ejecuta una acción por segundo de: "pasar balón", "chutar", "regatear"...
 - El resultado del partido es: *pierde, gana o empata*
 - ¿Cuánto aporta agente dadas sus acciones?
 - ¿Y si el efecto de las acciones tiene retraso?
- Problema fundamental del aprendizaje en SMA

CREDIT ASSIGNMENT EN SMA (II)

- Es fundamental diseñar una función de recompensa por lo que
 - Afecta a aprendizaje cooperativo y competitivo en SMA
 - Queremos maximizar el reward común

FUNDAMENTO TEÓRICO DEL APRENDIZAJE

- Para un solo agente: si entorno estacionario y suficiente experimentación del agente se garantiza convergencia a política óptima
- Esto no es válido en SMA:
 - Múltiples agentes aprendiendo en el mismo entorno.
 - Enfrentándose a acciones y recompensas inobservables de otros agentes
 - No estacionariedad del entorno...

Todas estas propiedades de los sistemas multiagente dificultan la ingeniería de algoritmos de aprendizaje capaces de encontrar soluciones óptimas

Veremos:

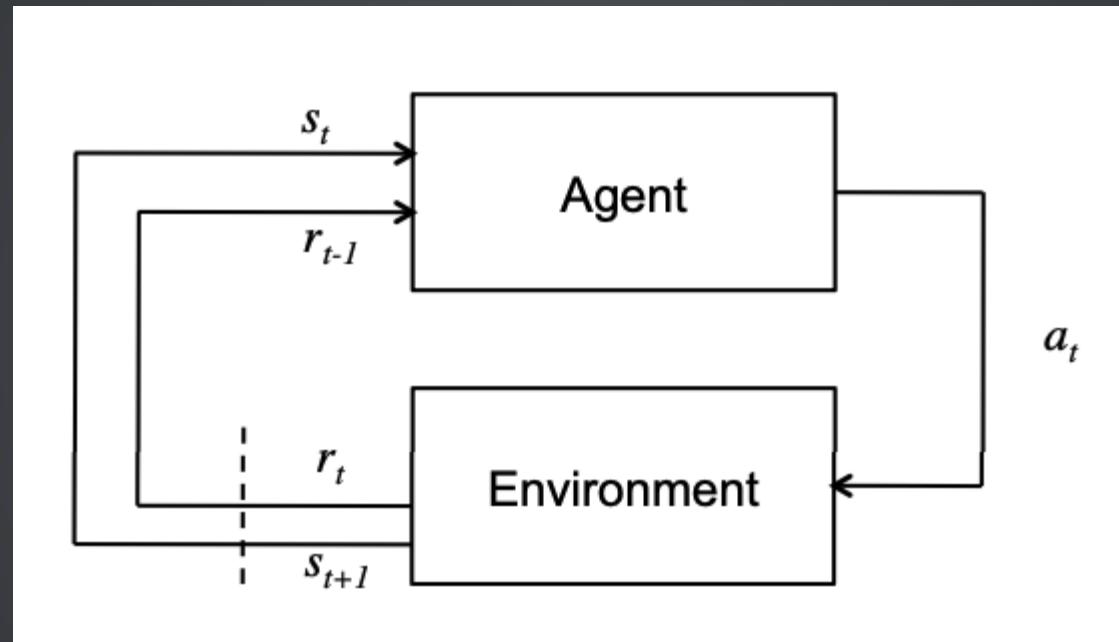
- Adaptación de un algoritmo clásico a SMA
- Métodos de optimización estocásticos para SMA

ADAPTANDO RL A MA-RL

EMPEZAMOS RECORDANDO RL CON UN ÚNICO AGENTE

MODELADO (I)

- Estados: s_t
- Acciones: a_t
- Reward: $r_t = r(s_t, a_t, s_{t+1}) \in \mathbb{R}$
- Transición entre estados:
$$P(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \dots) = P(s_{t+1}|s_t, a_t)$$



MODELADO (II)

- Finalidad de agente aprender la política π :
$$\pi(a|s) = P(a|s)$$
- La política debe maximizar el reward de todos los pasos h_π de la historia del agente:
$$R(h_\pi) = \sum_{t=1}^{T-1} \gamma^{t-1} r(s_t, a_t, s_{t+1})$$
- $\pi^* = \arg \max_\pi R(h_\pi)$

MODELADO (III)

Markov Decision Process (MDP) requiere definir

- El espacio de observación $s_t \in S$
- El espacio de acción $a_t \in A$
- $P(a|s)$: Definición de un modelo Neuronal compatible
- Reward $r_t = r(s_t, a_t, s_{t+1})$

EJEMPLO DE MODELADO MDP

MULTI-AGENT DEEP RL

Asumimos que:

- Los agentes deben maximizar el reward del grupo
- Los agentes tienen observaciones parciales y privadas
- No se pueden comunicar de manera explícita
- Deben aprender una tarea cooperativa mediante la observación

¿Cómo abordamos el problema?

SOL. 1: MODELO CENTRALIZADO (I)

- Asume modelo conjunto de las acciones y observaciones de **todos** los agentes:
 $a_1 \times a_2 \times \dots \times s_1 \times s_2 \times \dots$
- Problemas:
 - Se centraliza tanto el entrenamiento como la ejecución
 - Crecimiento exponencial de A y S con el número de agentes

SOL. 1: MODELO CENTRALIZADO (II)

- Asumimos que las acciones conjuntas se pueden *factorizar* en componentes individuales para cada agente
- Por tanto el controlador representará un conjunto de π independientes para cada agente
- Conseguimos $|A|^n$ pase a ser $n|A|$, aún así tenemos problemas con el espacio de observación S
- Impracticable para problemas complejos

SOL. 2: MODELO CONCURRENTE

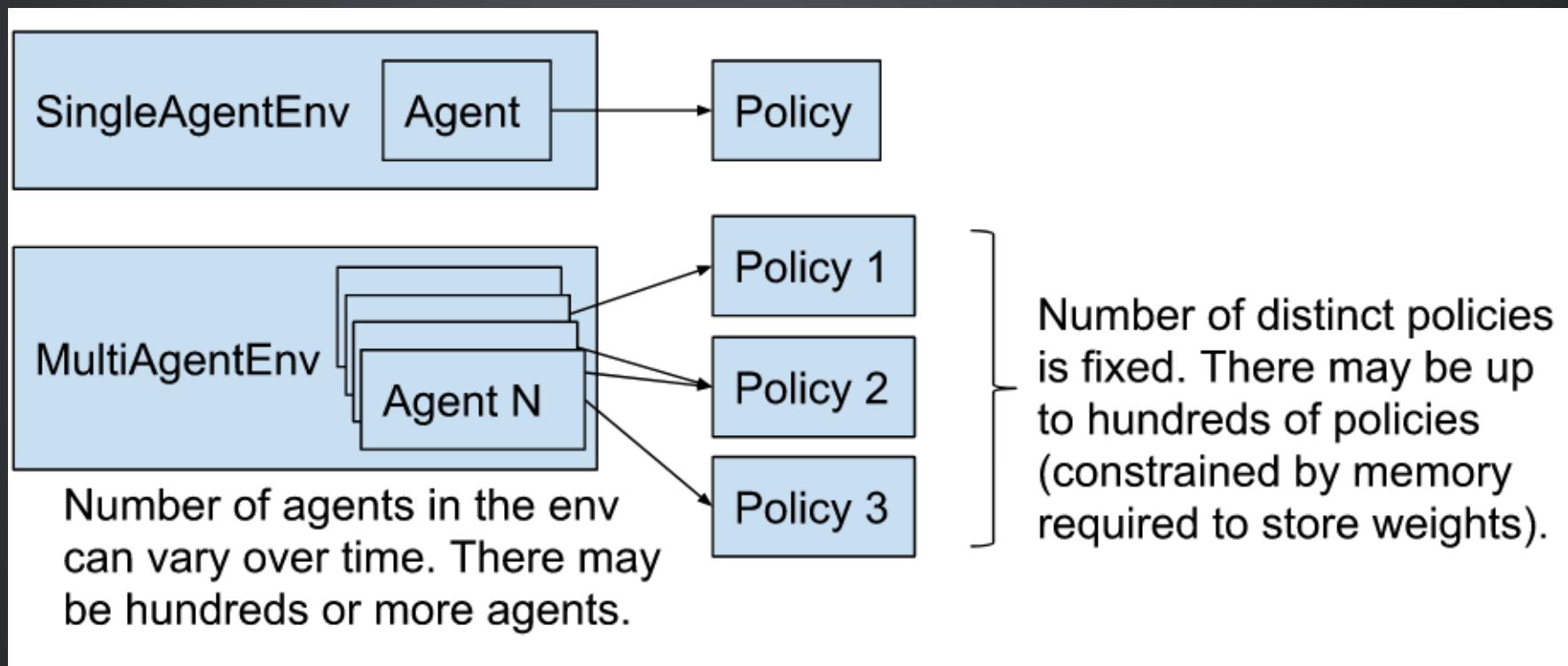
- Cada agente aprende su π individual
- A y S privadas por agente
- Las políticas son por tanto independientes
- Se deben optimizar las π de manera simultánea
- Fácil de usar en sistemas heterogéneos
- Problemas:
 - Poco escalable (aumento de la complejidad y memoria por agente)
 - Como cada agente es individual, el entorno se transforma en dinámico...

SOL. 3: PARÁMETROS COMPARTIDOS (I)

- Asume agentes homogéneos
- Los agentes comparten los parámetros de una política π
- La política se entrena de manera simultánea con la experiencia de todos los agentes
- Permite comportamientos distintos ya que cada uno percibe observaciones distintas
- Opción recomendada

SOL. 3: PARÁMETROS COMPARTIDOS (II)

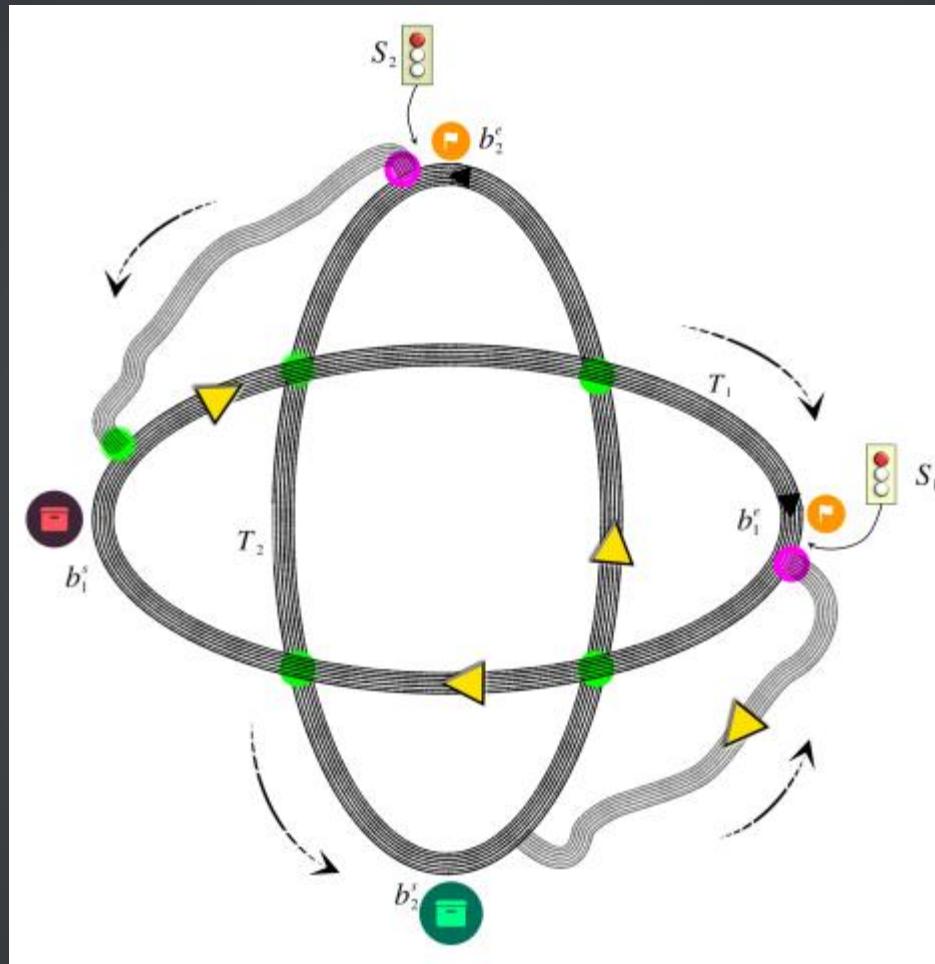
- ¿Y con agentes heterogéneos?



MULTI-AGENT DEEP RL

EJERCICIO. MODELAR EL SIGUIENTE PROBLEMA:

- Disponemos de una carretera *discreta*, formada por celdas, que conecta vehículos de transporte con almacenes
- Existen semáforos en la carretera que modulan el tráfico
- La carretera es la adjunta



VEHÍCULOS (I)

- Percibe vehículo Izq, Der. o en Frente
- Además es capaz de percibir su vel.
- La zona actual del track en el que se encuentra,
- Su estado de carga, si puede o no descargar la mercancía
- La información de un agente semáforo (si existe)
- Y si ha colisionado con el entorno o otro vehículo

VEHÍCULOS (II)

- En cada paso los vehículos pueden:
 - Permanecer parados
 - Desplazarse
 - Cargar o descargare su carga. asociados.

SEMÁFORO

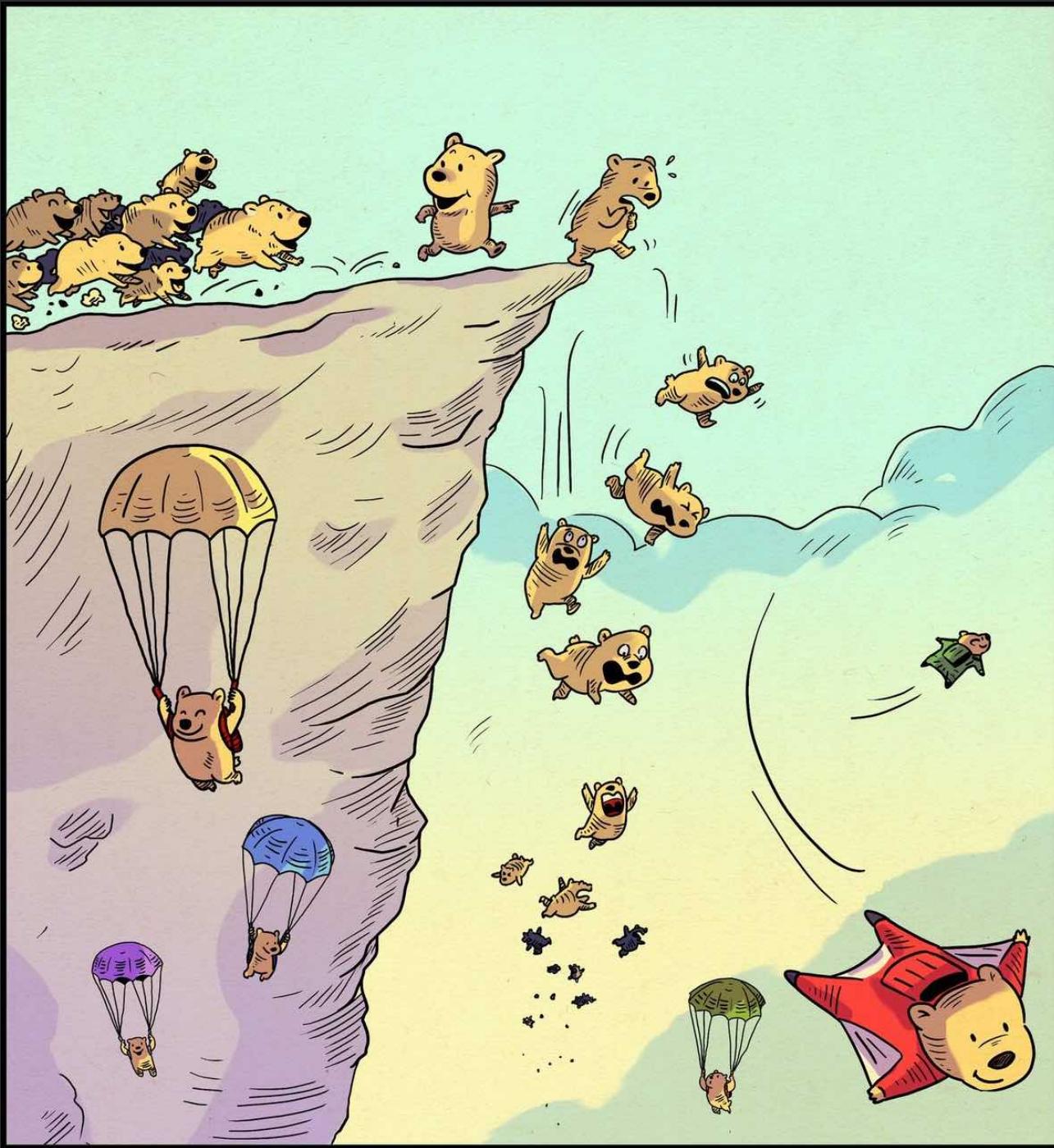
- Perciben la cantidad de vehículos que circulan por los carriles del entorno,
- Valor entero igual o mayor que 0.
- Por otra parte los semáforos pueden dejar que los vehículos circulen por el carril principal o obligarlos a circular por el desvío al que están

EJERCICIO

- Modela el **Markov Decision Process (MDP)** para poder aprender el problema mediante RL
- Usando *Parameter Sharing*: ¿Cuantas políticas se necesitarán si tenemos 10 coches circulando y 2 semáforos?
- ¿Qué entradas y salidas tendría las NNs que utilizariamos para aprender este SMA?

ESTRATÉGIAS EVOLUTIVAS (ES)

<https://blog.otoro.net/2017/10/29/visual-evolution-strategies/>



¿BACKPROPAGATION?

- Redes neuronales muy expresivas y flexibles
- Resolver muchos problemas complejos
- El éxito del aprendizaje profundo: uso de backpropagation:
 - Calcular eficientemente el gradiente de una función objetivo sobre cada parámetro del modelo
- Con estos gradientes, podemos buscar eficientemente en el espacio de parámetros para encontrar una solución
- Pero....

NO TODO ES BACKPROPAGATION

- Hay muchos problemas en los que no se puede utilizar el algoritmo de backpropagation:
 - Máximos y mínimos locales
 - Aprendizaje por refuerzo: estimar gradiente de recompensa de un agente
 - ¡La recompensa tarda muchos pasos!
 - Recompensa en el mundo real es ruidosa y dispersa...
 - A veces es mejor ignorar el gradiente

ESTRATEGIAS EVOLUTIVAS (I)

<https://openai.com/research/evolution-strategies>

We've discovered that evolution strategies (ES), an optimization technique that's been known for decades, rivals the performance of standard reinforcement learning (RL) techniques on modern RL benchmarks (e.g. Atari/MuJoCo), while overcoming many of RL's inconveniences.

ESTRATEGIAS EVOLUTIVAS (II)

Ventajas:

- No need for backpropagation
- Highly parallelizable
- Higher robustness
- Structured exploration
- Credit assignment over long time scales

¿QUÉ ES UNA ES?

- Algoritmo que proporciona un conjunto de soluciones candidatas para evaluar un problema.
- La evaluación se basa en una función objetivo que toma una solución dada y devuelve un único valor de aptitud (*fitness*).
- El algoritmo producirá la siguiente generación de soluciones candidatas (dado su *fitness*)
- El proceso iterativo se detendrá una vez que la mejor solución conocida sea satisfactoria para el usuario.

ALGORITMO GENÉRICO ES

```
solver = EvolutionStrategy()
while True:

    solutions = solver.ask()
    fitness_list = np.zeros(solver.popsize)

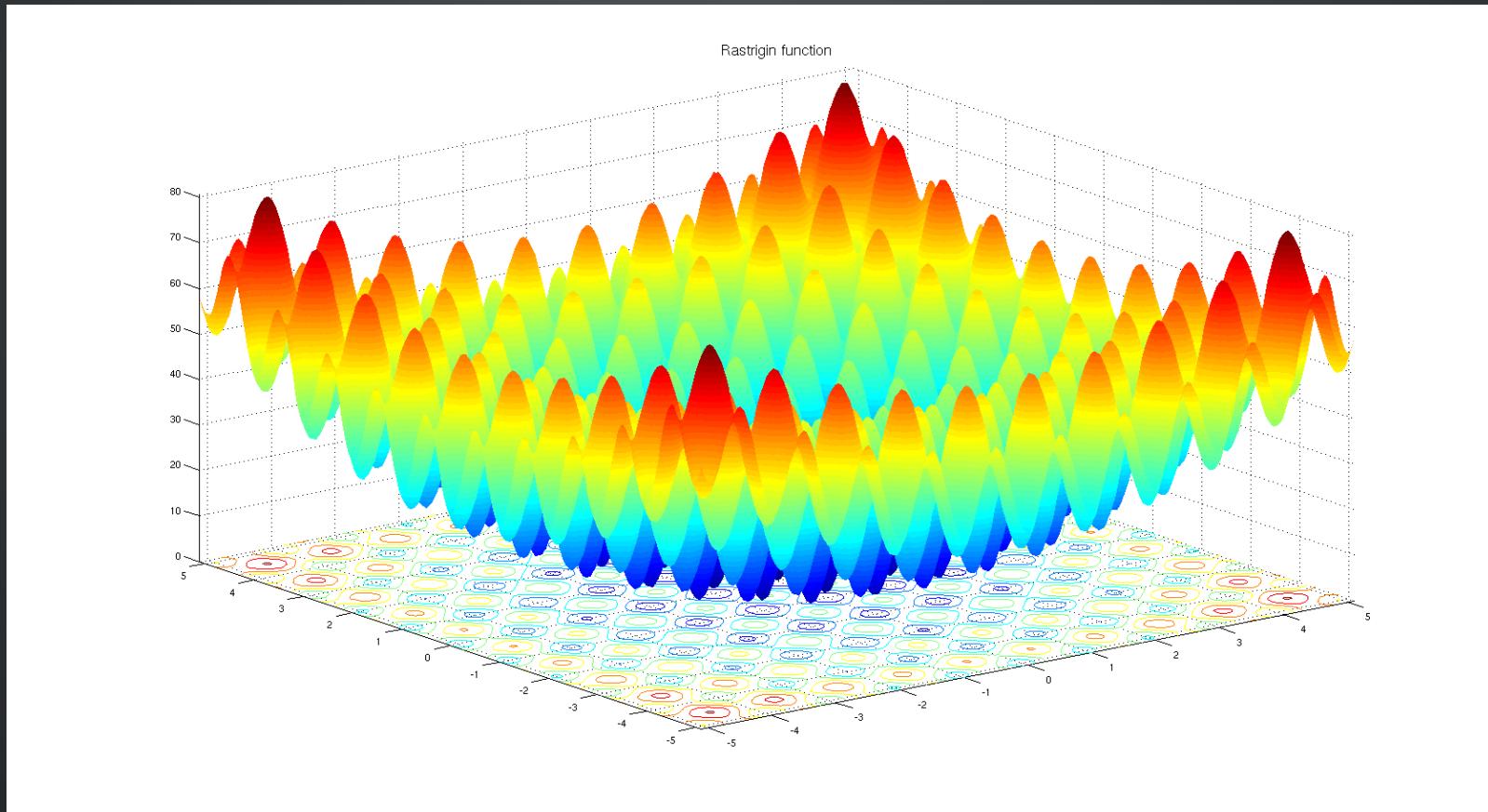
    for i in range(solver.popsize):
        fitness_list[i] = evaluate(solutions[i])

    solver.tell(fitness_list)

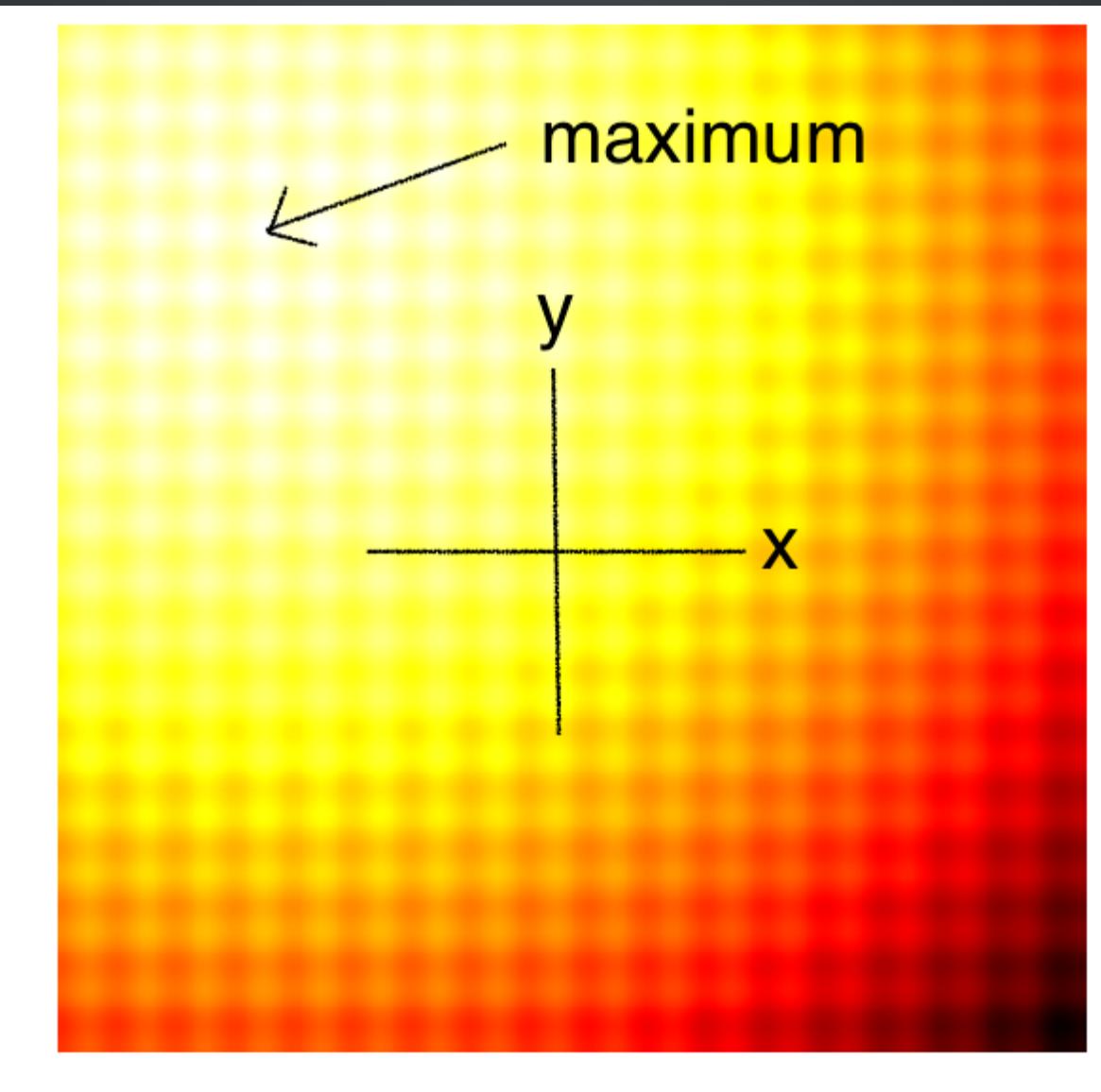
    best_solution, best_fitness = solver.result()
    if best_fitness > MY_REQUIRED_FITNESS:
        break
```

APROX. INTUITIVA (I)

- Intentado obtener el máximo de la función de Schaffer and Rastrigin



APROX. INTUITIVA (II)



SOL 1. APROX. SIMPLE

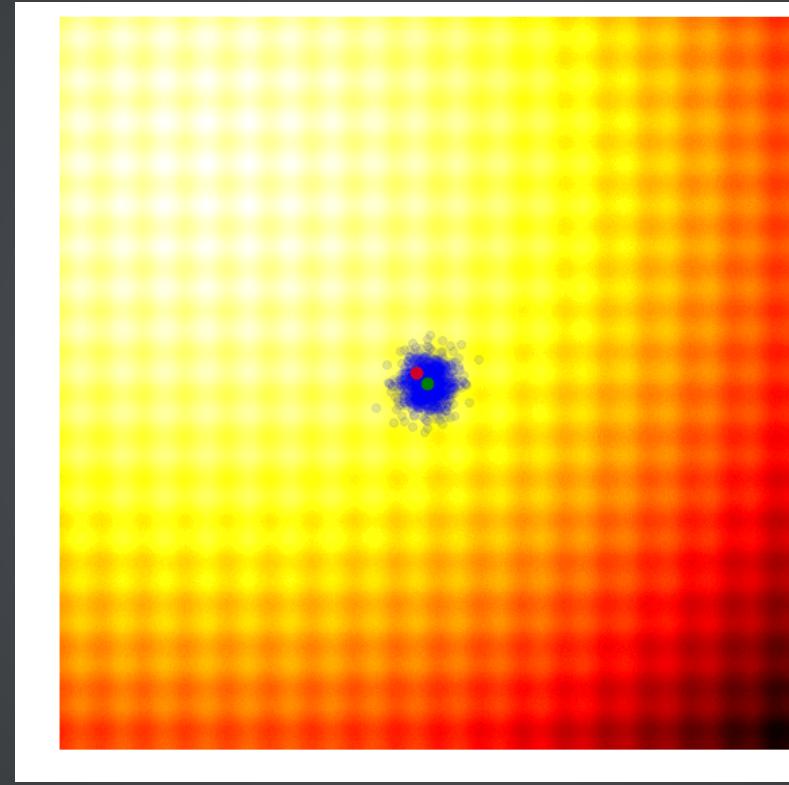
EXTRAER DE UNA DISTRIBUCIÓN NORMAL

- $\mu = (\mu_x, \mu_y)$ y $\sigma = (\sigma_x, \sigma_y)$
- population se extraerá al azar de esa distribución
- Nos quedaremos con el mejor individuo



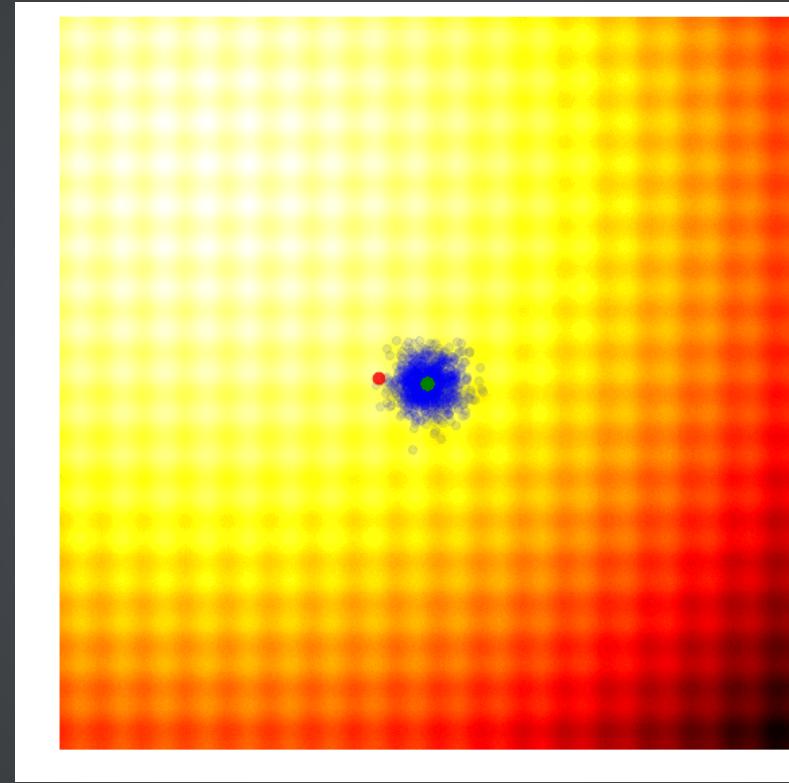
ALGORITMO

```
mu = origen
population = draw(mu, sigma)
for i in num_generaciones
    best = mejor(evaluar(population))
    (mu,sigma) = best
    population = draw(mu, sigma)
```



SOL 2. ALGORITMO GENÉTICO

- 10% de los mejores individuos se copian a la siguiente generación
- Cruce al azar (que coordenada copiar de que individuo)
- Ruido gausiano como mutación



CMA-ES (I)

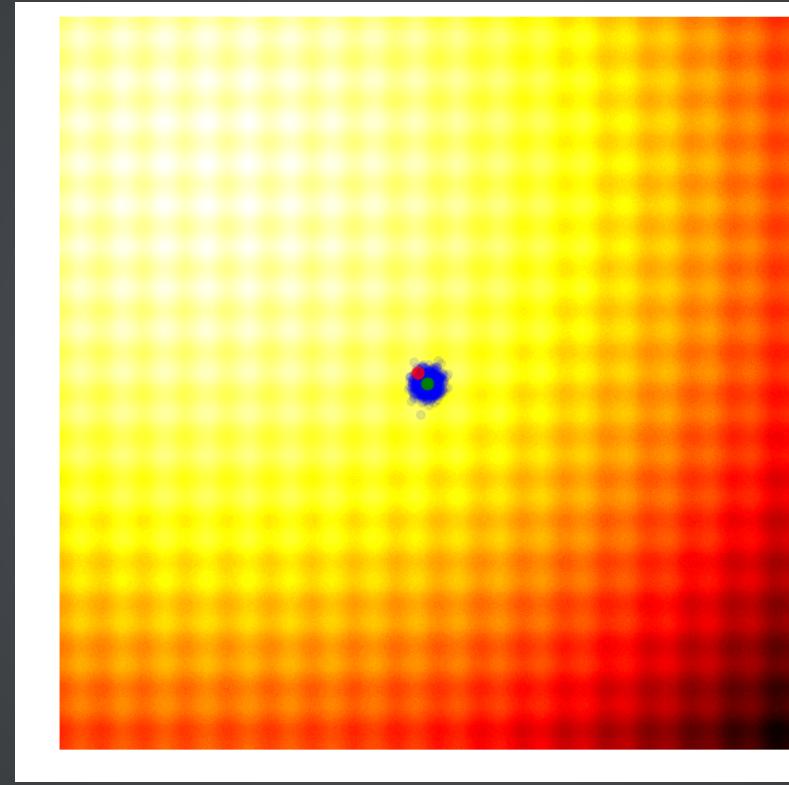
COVARIANCE-MATRIX ADAPTATION EVOLUTION STRATEGY

- ¿Y si se incrementara/decrementara la varianza cuando sea necesario?
- CMA-ES proporcionará una distribución normal de donde extraer la nueva población en cada generación
- Trabajará con una matriz de covarianza

CMA-ES(II)

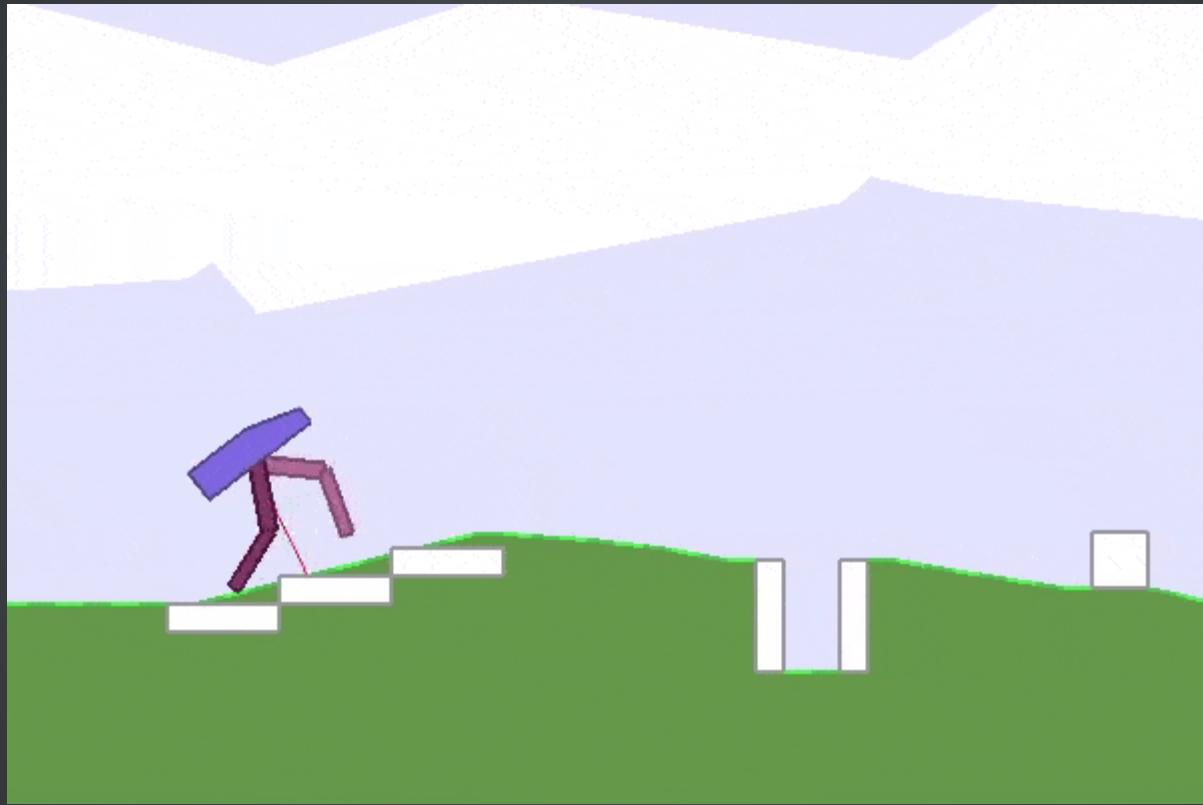
ALGORITMO

1. Calcula el *fitness* de cada individuo de la generación (g).
2. Escoge los 25% mejores de (g).
3. Usando los individuos de (2), con la media de la actual generación $\mu^{(g)}$ calcula la nueva matriz de covarianza $C^{(g+1)}$.
4. Genera un nuevo grupo de soluciones candidatas (individuos) con media $\mu^{(g+1)}$ y covarianza $C^{(g+1)}$
-



VARIAS ESTRATEGIAS ES

- Simple
- Algoritmos genéticos
- CMA-ES
- Natural ES
- OpenAI ES
- ...



APLICANDO *ESA SMA*

- Uso de **parámetros compartidos**
- Ejecutar **todo** el episodio para obtener el reward
- Hiperparámetros en ES
- Paralelismo innato

NEUROEVOLUCIÓN

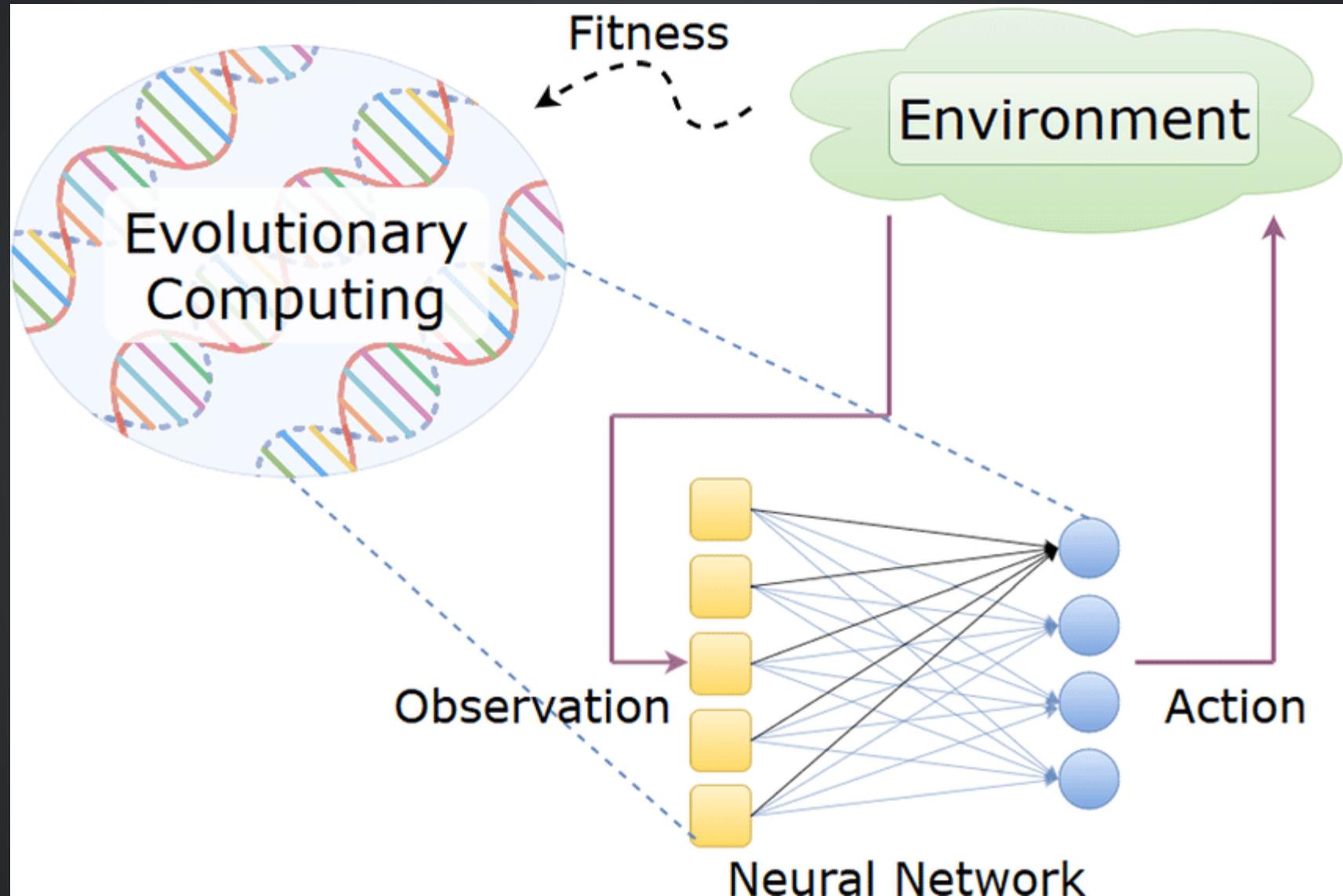
NEUROEVOLUCIÓN (I)

- Uso de ES para entrenar una política
- Se busca entre una población de políticas (NN) para ver la más adecuada
- Se aplican operadores (cruce, mutación...) a NN
- Este tipo de enfoque de "búsqueda de políticas" es más adecuado para dominios continuos en los que los enfoques tradicionales RL suelen encontrar dificultades.

NEUROEVOLUCIÓN (II)

PASOS

1. Representación
2. Selección
3. Generación de nuevas políticas
4. Evaluación
5. Reemplazo



ALGORITMO

```
1 Initialization Phase:  
2 At  $t = 0$   
3 for Each policy  $\pi_k$  where  $k \in (1, N_{pop})$  do  
4   Use policy  $\pi_k$  for a fixed number of steps  
5   Evaluate performance of  $\pi_k$   
6 end  
7 Training Phase:  
8 for  $t < t_{max}$  do  
9   Select a policy  $\pi_i$  from population of policies:  
10  with probability  $\epsilon$ :  $\pi_{current} \leftarrow \pi_i$   
11  with probability  $1 - \epsilon$ :  $\pi_{current} \leftarrow \pi_{best}$   
12  Modify the parameters of  $\pi_{current}$  to produce  $\pi'$  (mutation)  
13  Use policy  $\pi'$  for a fixed number of steps  
14  Evaluate performance of  $\pi'$   
15  Replace  $\pi_{worst}$  with  $\pi'$   
16   $t \leftarrow t + 1$   
17 end
```