

# UN ACERCAMIENTO EVOLUTIVO DE APRENDIZAJE PROFUNDO A MONOPOLIO

Carlos Andrés  
Coronado  
Arrázola  
Universidad Privada  
Boliviana

Práctica  
Internacional

# Contenido de la presentación

¿Las máquinas pueden pensar?

Alan Turing

Descripción del Problema

Presentación de la Hipótesis

Metodología

Espacio de Estado y Acción

Estructura Interna del Agente

Planteamiento de SEARL  
para el entrenamiento

Resultados

Conclusiones

# ¿Cuál es la importancia de los juegos en Aprendizaje Reforzado?

Presentan un entorno de fácil  
planteamiento, ideal para dar los  
primeros pasos en un proyecto

Ambiente definido



Objetivos claros



Reglas claras



¿Que modelado de Agente de Aprendizaje Reforzado es el más óptimo para juegos en los que el éxito depende de una interacción estrecha con los demás jugadores?

# Descripción del Problema

Comercio



Carrera por territorio



Carrera por recursos



# Caso de estudio: Monopolio

Es un juego en el que la estrategia , el comercio y la aleatoriedad juegan un rol clave en el desarrollo de sus partidas



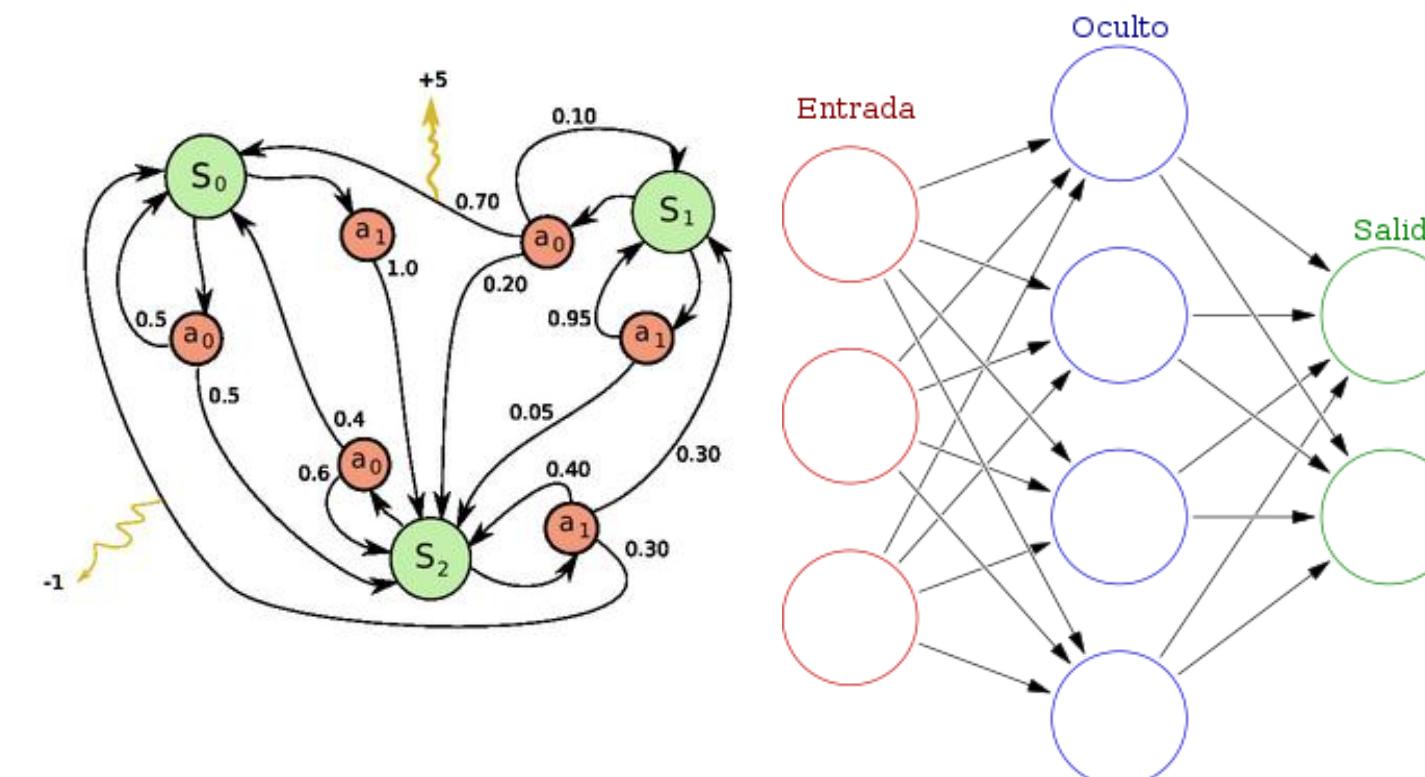
# Presentación de la Hipótesis

El modelo óptimo:

Proceso de Decisión de Markov

Redes Neuronales

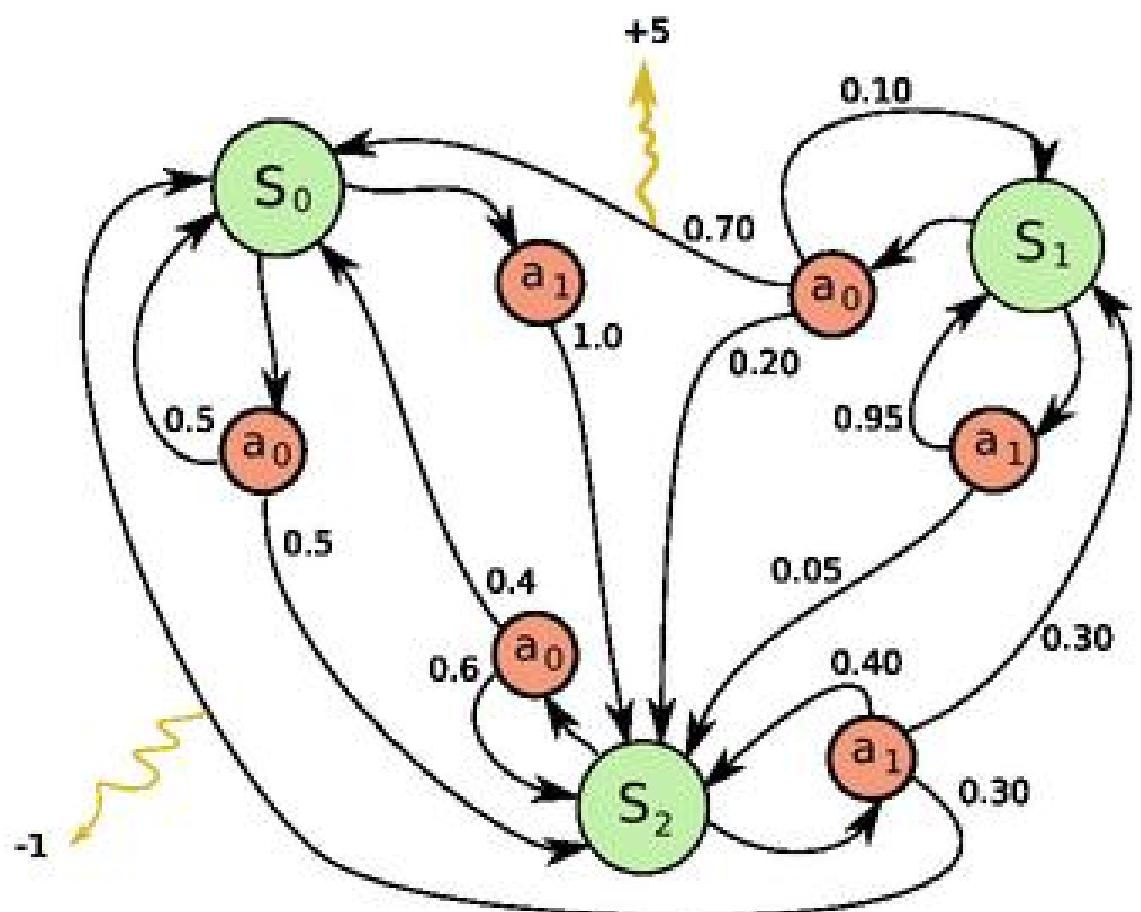
Entrenamiento con un algoritmo genético



# Metodología



# Proceso de Decisión de Markov



Permite abstraer representaciones matemáticas

Ideal para afrontar problemas de optimización dinámica, debido a esto va de la mano con el aprendizaje reforzado

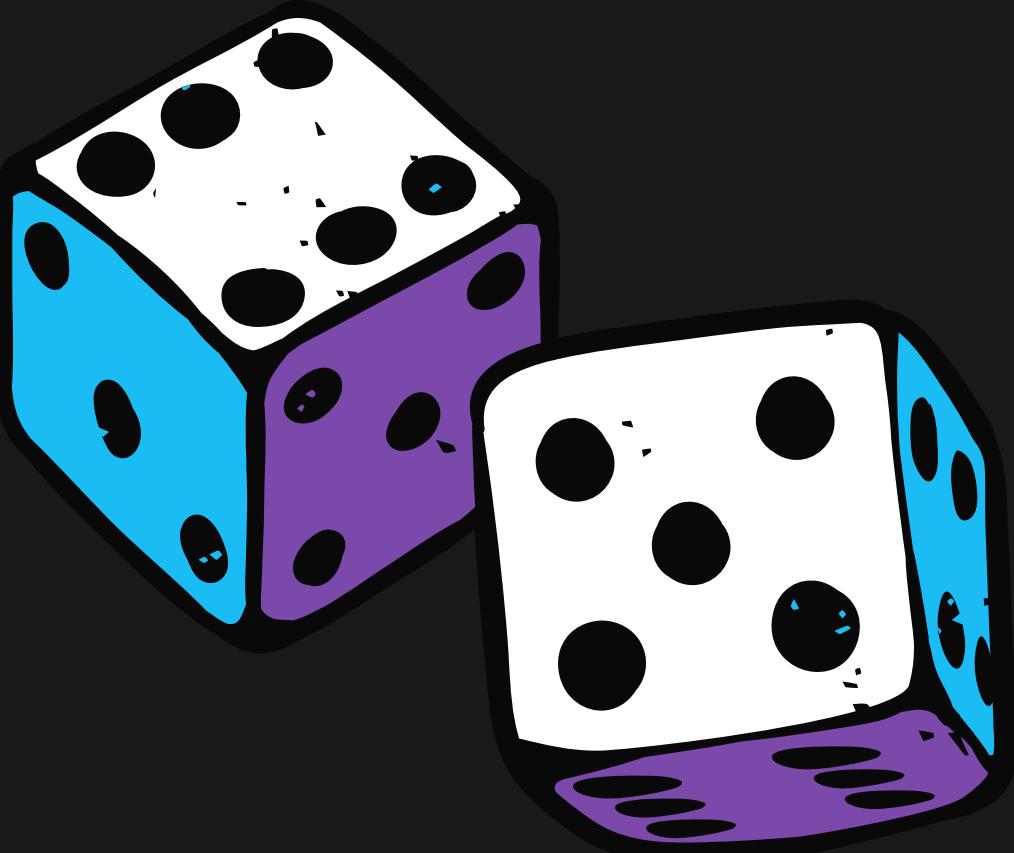
Componentes:

- Set de Estados
- Set de Acciones para el estado dado
- Probabilidad de Transición

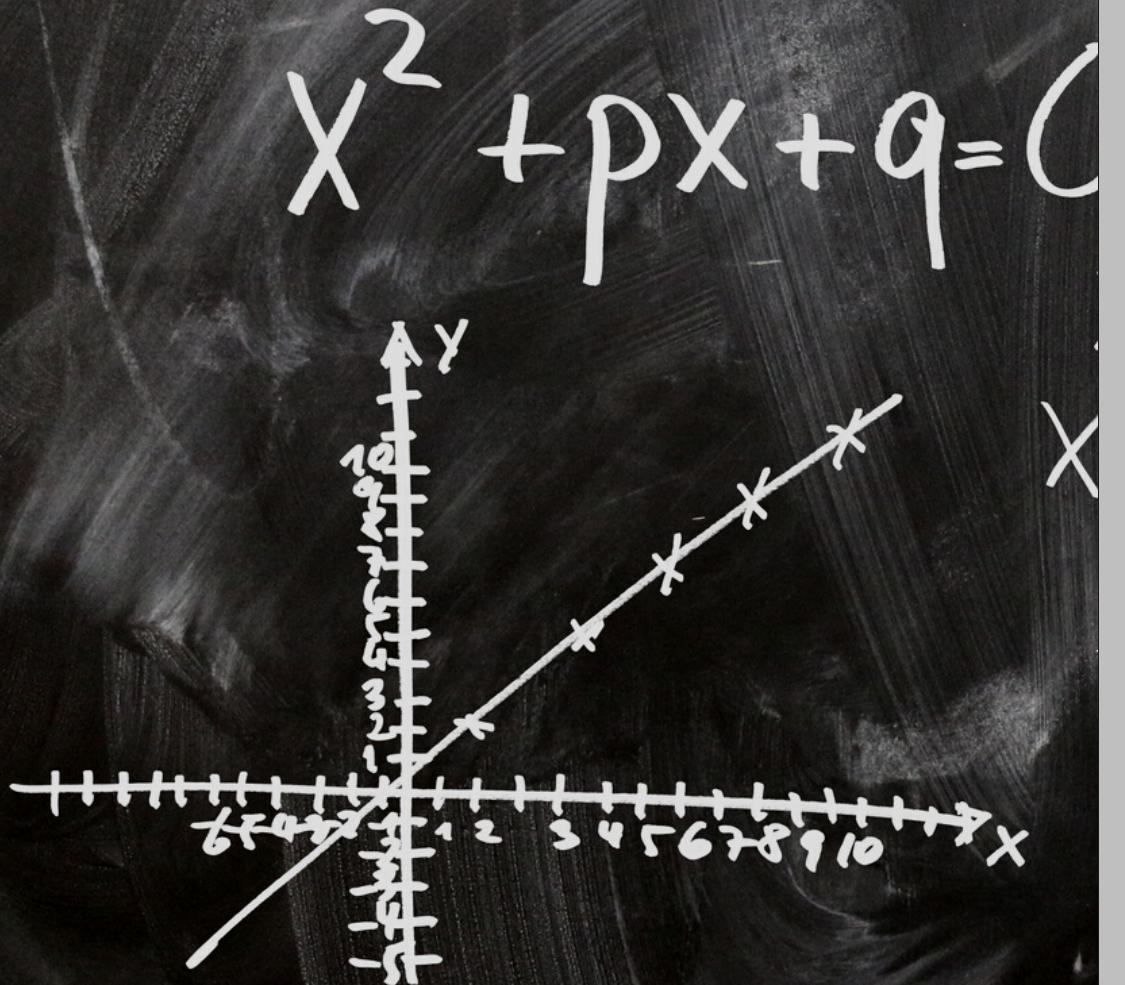
# ¿Que acciones puede tomar el jugador?

Tipo de Acción	Iniciada por	Fase del Juego
Realizar Oferta	Jugador	Pre-turno, Fuera-de-turno
Continuar en Subasta	Banco	Subasta
Construir casa	Jugador	Pre-turno, Fuera-de-turno
Vender casa	Jugador	Pre-turno, Post-turno, Fuera-de-turno
Hipotecar Propiedad	Jugador	Pre-turno, Post-turno, Fuera-de-turno
Pagar Hipoteca	Jugador	Pre-turno, Post-turno, Fuera-de-turno
Concluir Acciones	Jugador	Pre-turno, Post-turno, Fuera-de-turno
(Carta) Salir de la Cárcel	Jugador	Pre-turno
Pagar Fianza	Jugador	Pre-turno
Aceptar Oferta	Banco, Oponente	Pre-turno, Fuera-de-turno
Comprar Propiedad	Jugador	Post-turno

# Espacio de Acción



# Representación Matemática



## Representación general de las acciones

Tipo de Acción	Propiedades asociadas	Dimensiones
Realizar Oferta	Todas	61
Continuar en Subasta	Ninguna	2
Construir casa	Grupo-de-color	22
Vender casa	Grupo-de-color	22
Hipotecar Propiedad	Todas	28
Pagar Hipoteca	Todas	28
Concluir Acciones	Ninguna	1
(Carta) Salir de la Cárcel	Ninguna	1
Pagar Fianza	Ninguna	1
Aceptar Oferta	Ninguna	1
Comprar Propiedad	Ninguna	1

## Representación de Realizar una Oferta

Campo	Dimensiones
Propiedades ofrecidas	28
Propiedades deseadas	28
Dinero ofrecido	1
Dinero deseado	1
Jugador Objetivo	3

## Representación de Continuar en Subasta

Campo	Dimensiones
Continuar	1
Cantidad a ofertar	1



# Espacio de Estado

## Jugador:

- Posición
- Dinero
- (Carta) Salir gratis de la cárcel
- Cantidad de sets completados

## Propiedades Representación Binaria:

- 1: Posesión
- 0: No poseido

## Propiedades Representación Completa:

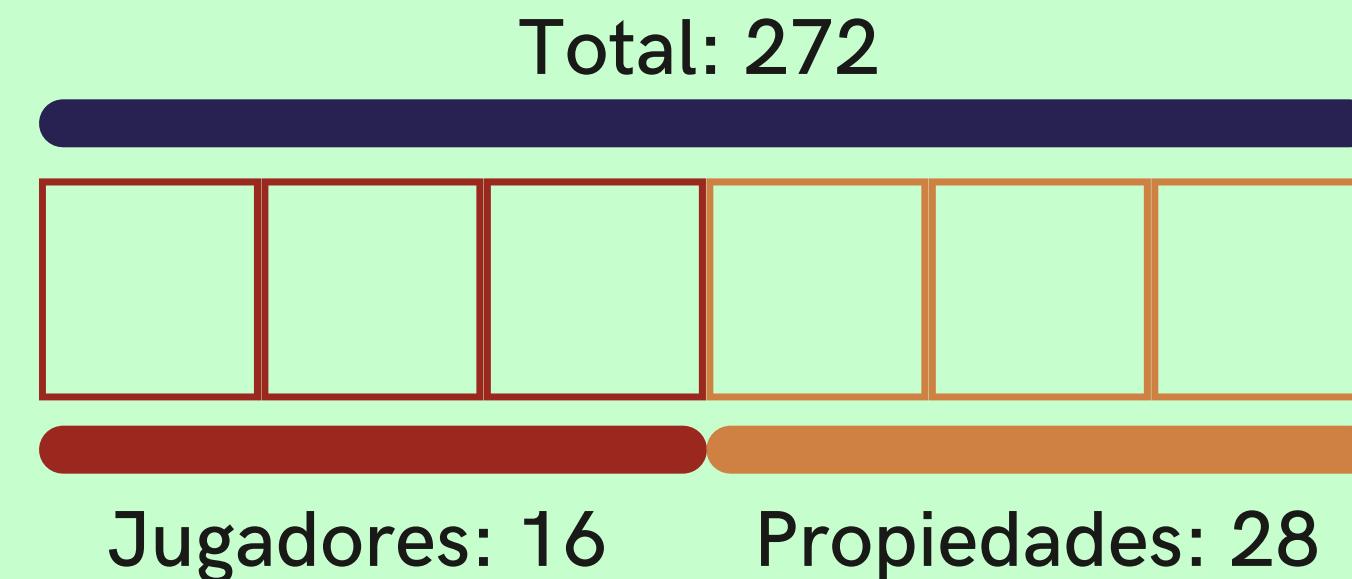
- Propietario
- Estado de Hipoteca
- Renta
- Color de Avenida
- Casas construidas
- Set de color completo

# Espacio de Estado

Representación Matemática

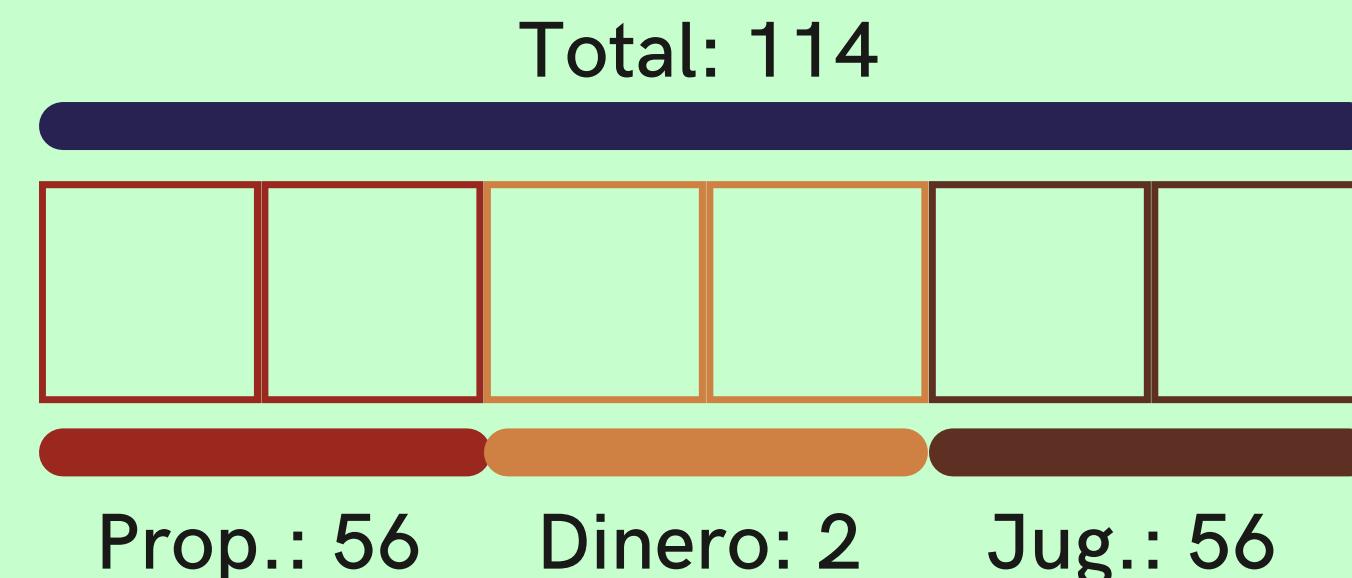
## Espacio de Estado Regular:

- 4 Jugadores
- 28 Propiedades (Completa)



## Espacio de Estado de Comercio:

- Propiedades Ofrecidas (Binaria)
- Propiedades Pedidas (Binaria)
- Dinero Ofrecido
- Dinero Pedido
- Propiedades Jugador Ofertante (Binaria)
- Propiedades Jugador Objetivo (Binaria)

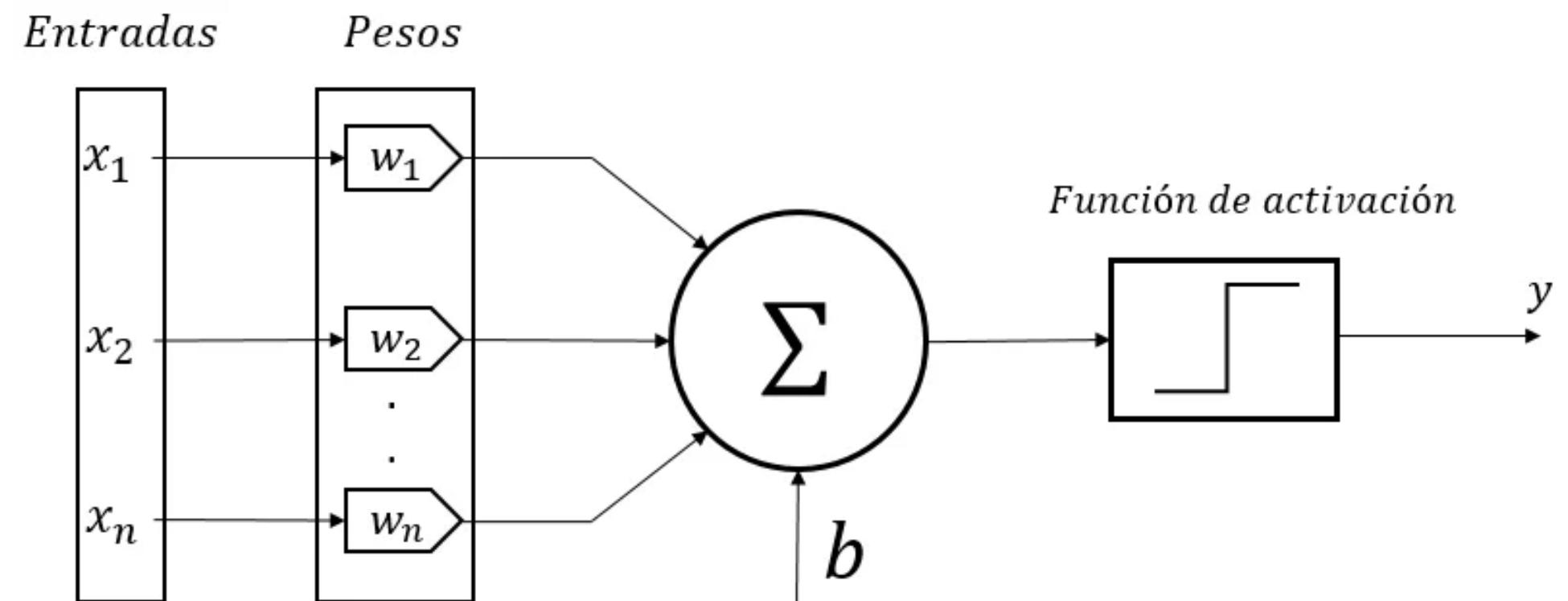
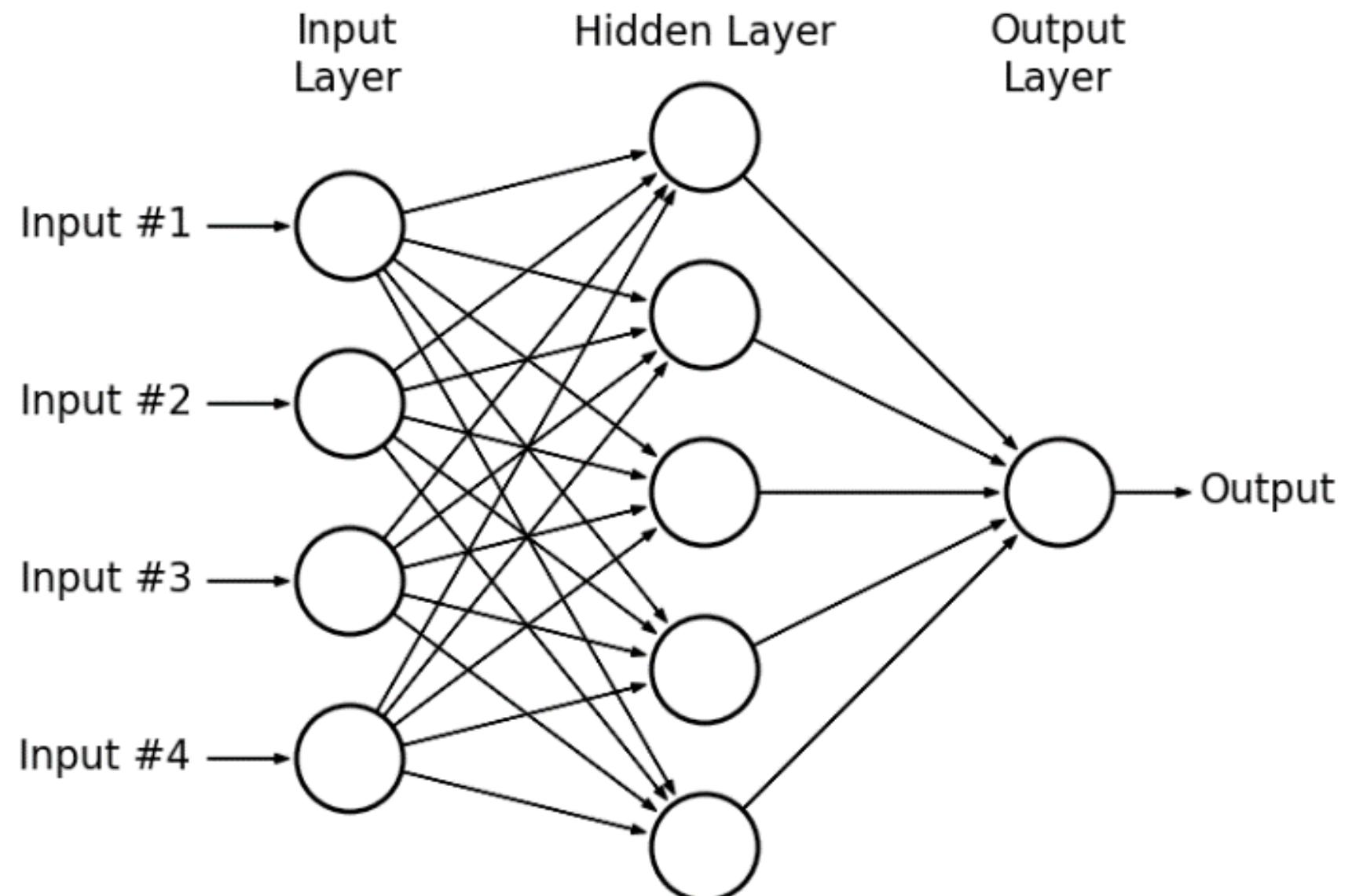


# Red Neuronal

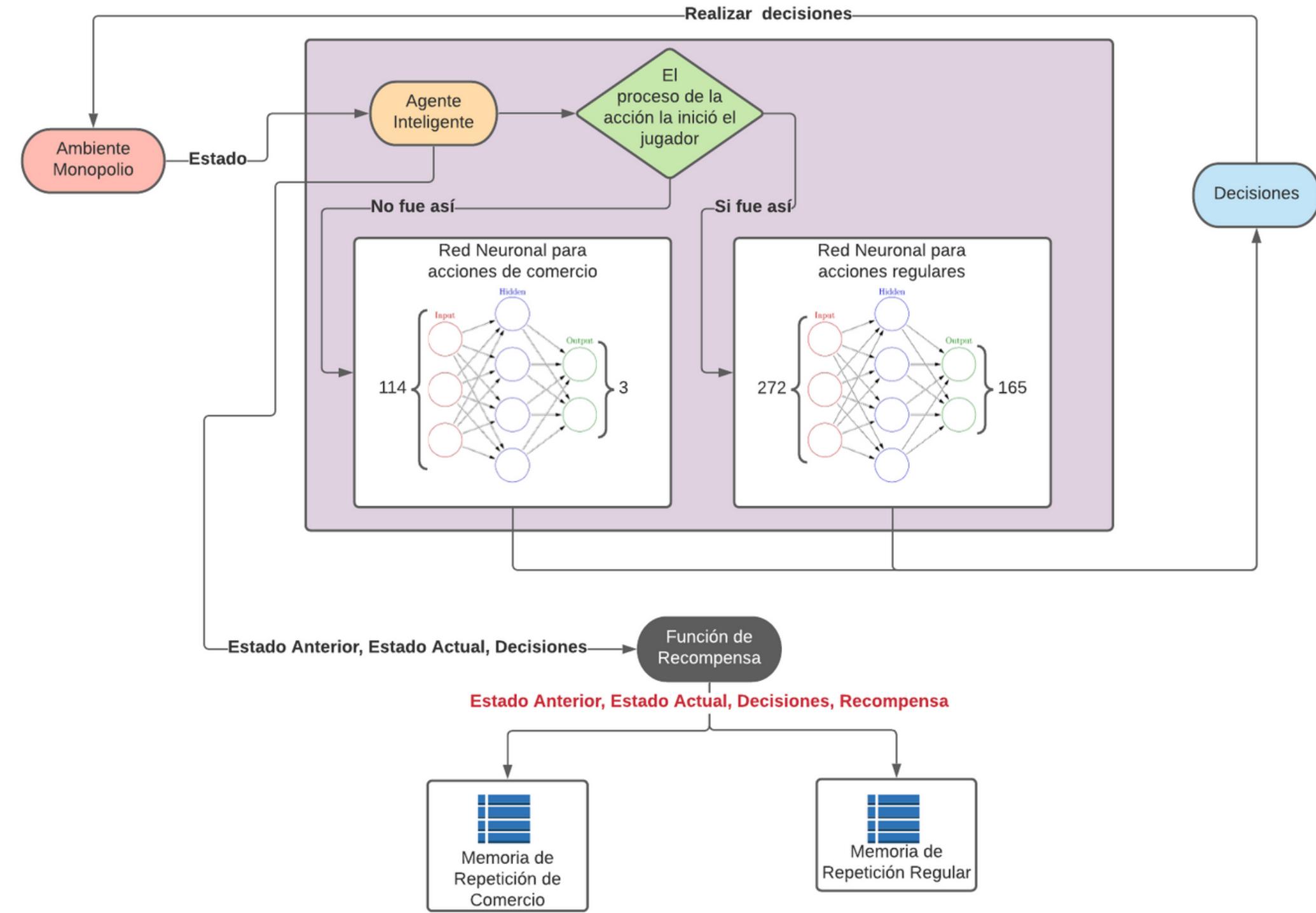
Intenta de emular al cerebro humano

Modelo estadístico

Posee un conjunto de perceptrones (representaciones de las neuronas) que se activan y devuelven un valor en función a lo se ingrese y a su modo de activación



# Estructura interna del Agente



Diferenciación entre contextos



Iniciada por el jugador



Iniciada por oponente

# Función de Recompensa

DETERMINA EL COMPORTAMIENTO DEL AGENTE

$$r(x) = \frac{x}{1 + |x|}$$

$$x = asset_{factor} + finance_{factor}$$

$j \rightarrow \#$  de propiedades del jugador  
 $\mu_p \rightarrow$  media del  $\#$  de prop de los oponentes  
 $c \rightarrow \#$  de colores completados  
 $h \rightarrow \#$  de casas construidas  
 $d_j \rightarrow$  dinero del jugador  
 $\mu_{op} \rightarrow$  media del dinero de los oponentes

Lo que se desea alcanzar

Premiar el que el agente se encuentre por encima de la media

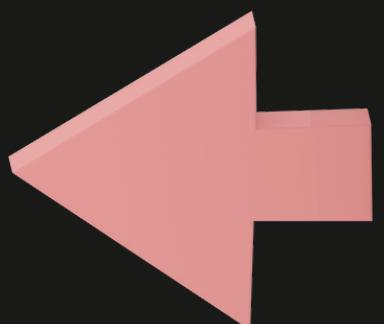
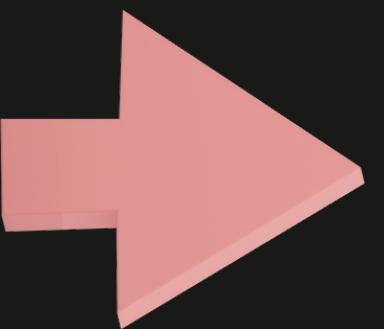
Factor finacienro:

$$finance_{factor} = d_j - \mu_{op}$$

Factor de bienes (premiando además el procentaje de casas construidas):

$$asset_{factor} = \begin{cases} j - \mu_p ; & si \ c < 1 \\ (j - \mu_p) + \left(\frac{h}{5*c}\right) ; & si \ c \geq 1 \end{cases}$$

# SELECT



# ALGORITMO GENETICO

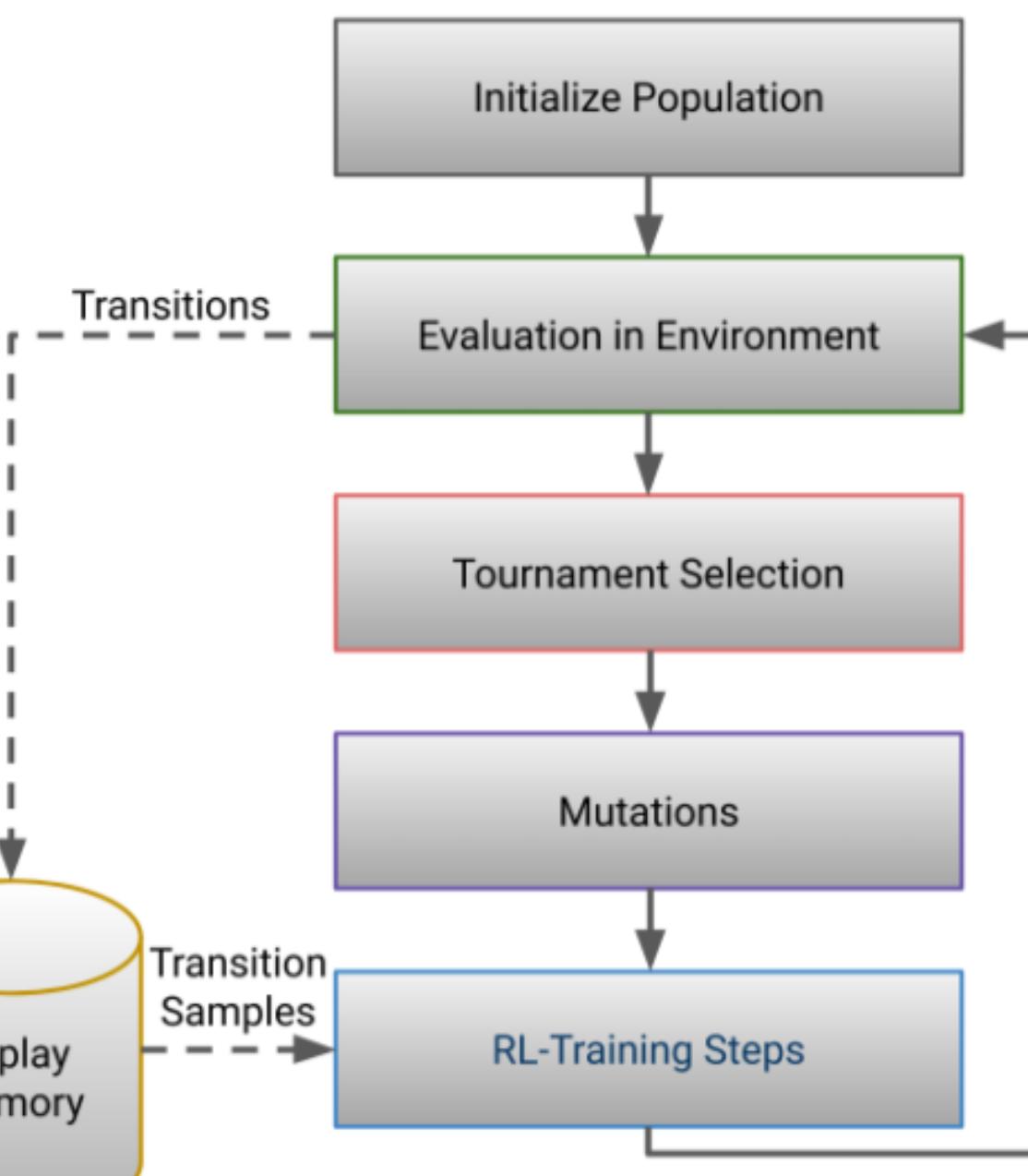
Emula el proceso evolutivo

De una población inicial se realiza un proceso de selección para escoger a la siguiente generación la cual mutará y se volverá a repetir el proceso

EFICIENTE OPTIMIZACION Y  
CONFIGURACION DINAMICA

# SEARL

Algoritmo genético



## Mutaciones:

- Ruido Gaussiano a los pesos de la red
- Cambio en la función de activación
- Cambio en la arquitectura interna
- Cambio en los hiperparámetros
- Ninguna

## Selección Elitista:

El que mayor valor de ajuste tenga ya tiene un puesto reservado en la próxima generación

## Valor de Ajuste:

Media de las recompensas obtenidas

Permite encontrar la configuración óptima

# Planteamiento de SEARL

---

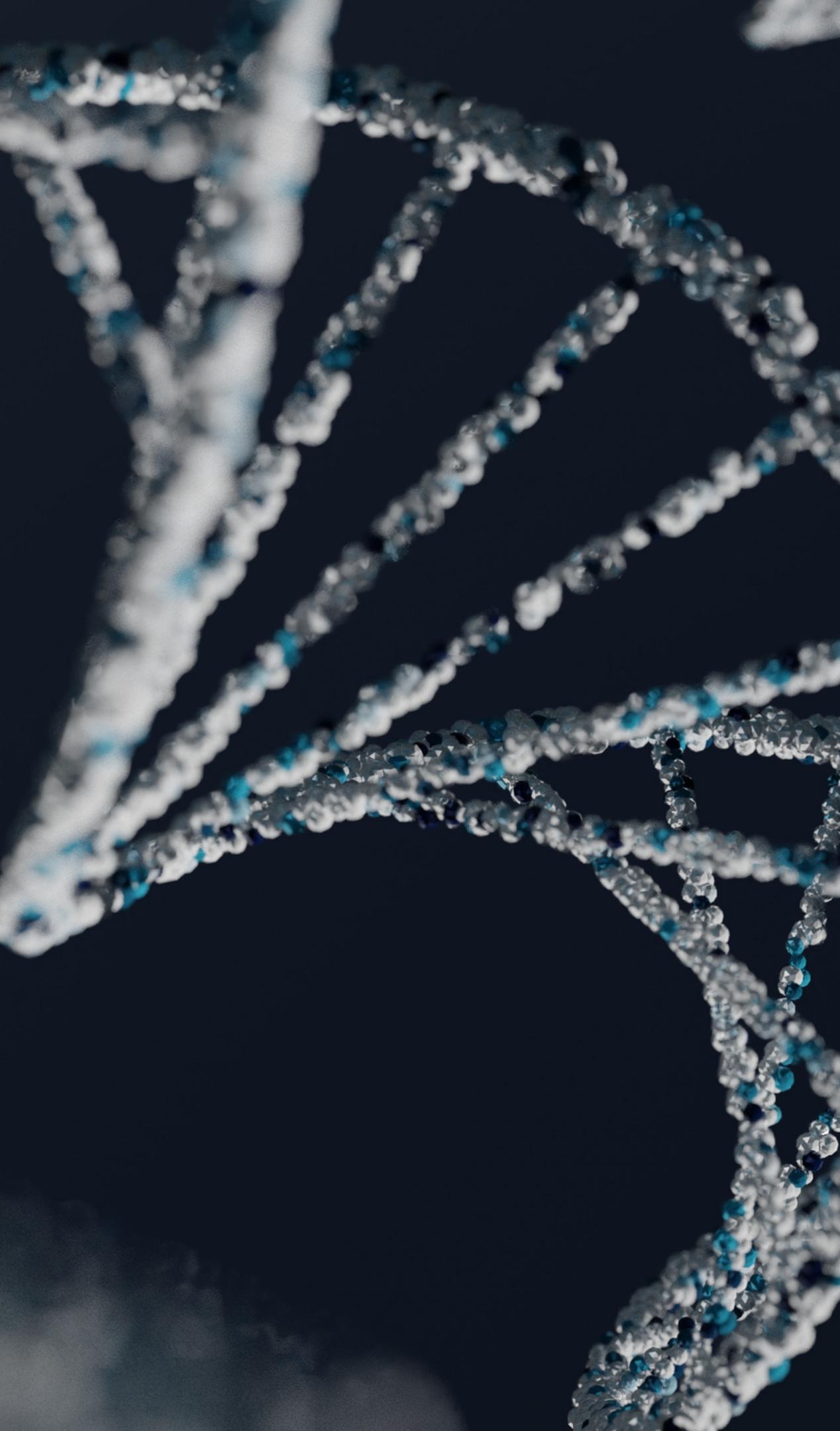
## Configuración inicial

Arquitectura interna de la red neuronal:

- 2 capas internas
- 64 nodos de profundidad
- Activación capas internas y capa de entrada: ReLU
- Activación capa de salida: Sigmoide

Población:

- 12 agentes en total
- 10 Agentes Inteligentes
- 2 Agentes Fijos
  - Uno prioriza a los ferrocarriles y al set de color azul oscuro
  - El otro prioriza a los set de color celeste y anaranjado



# Herramientas





# Herramientas

## Lenguaje de Programación: Python

Favorecedor debido:

- Gran comunidad y documentación completa
- Versatilidad
- La gran cantidad de librerías disponibles

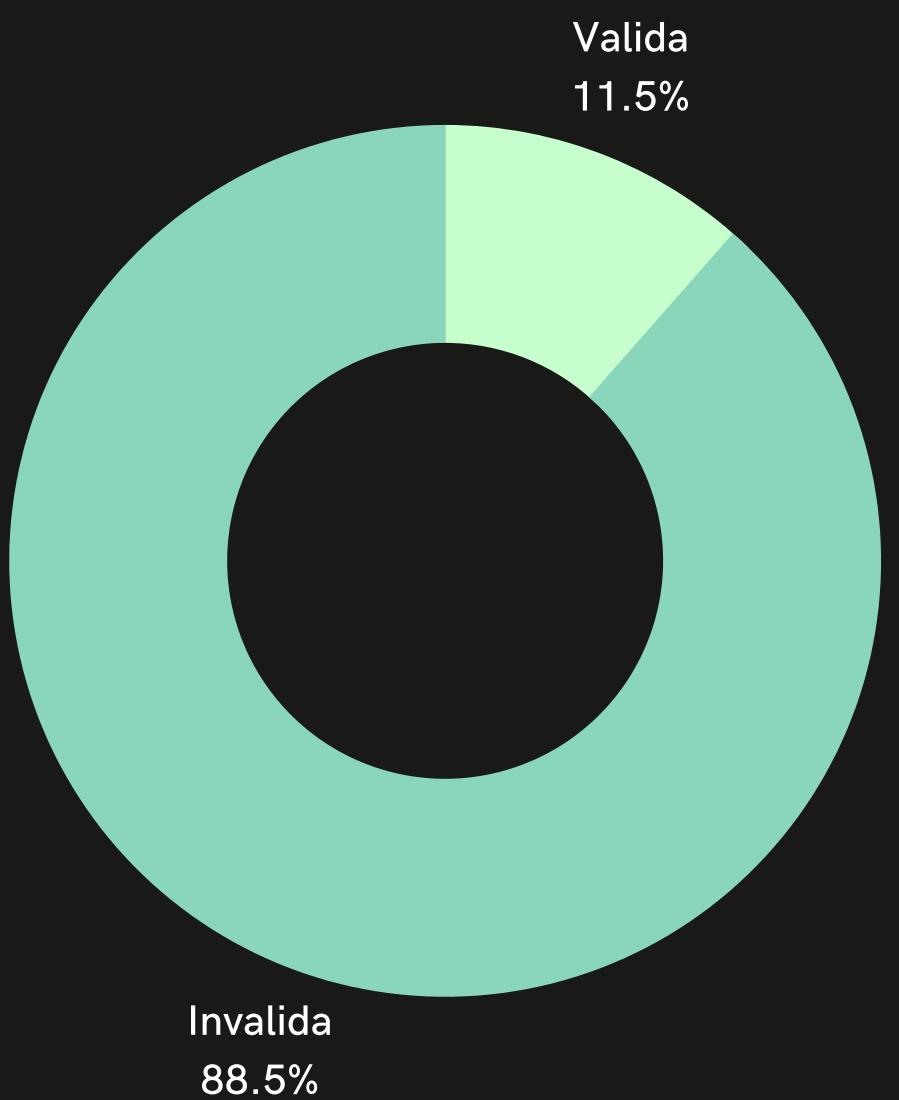
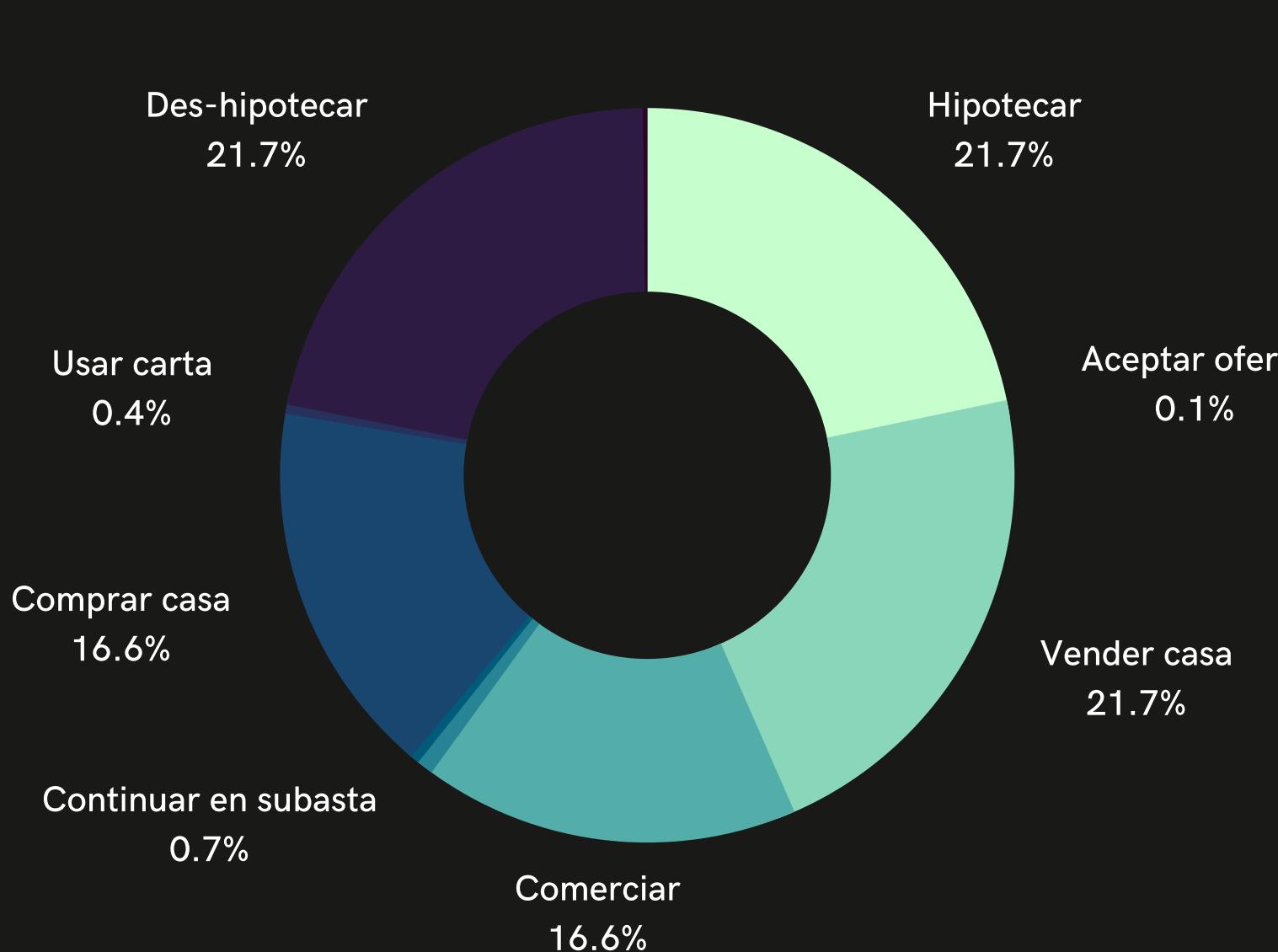
## Librerías más importantes

- Numpy:
  - Para trabajar con álgebra lineal
- Dataclasses, ABC, Random
- torch (PyTorch):
  - Tensores
  - Optimizadores
  - Red
- collections
  - OrderedDict
  - deque

# Resultados



# Resultados del Entrenamiento



## Datos importantes

Total de ofertas realizadas: 244982

Total de ofertas válidas: 1108

Duración de los agentes fijos en la población: 5 generaciones

Total de ofertas realizadas en las primeras dos generaciones: 16718

Total de ofertas válidas en las primeras dos generaciones: 261

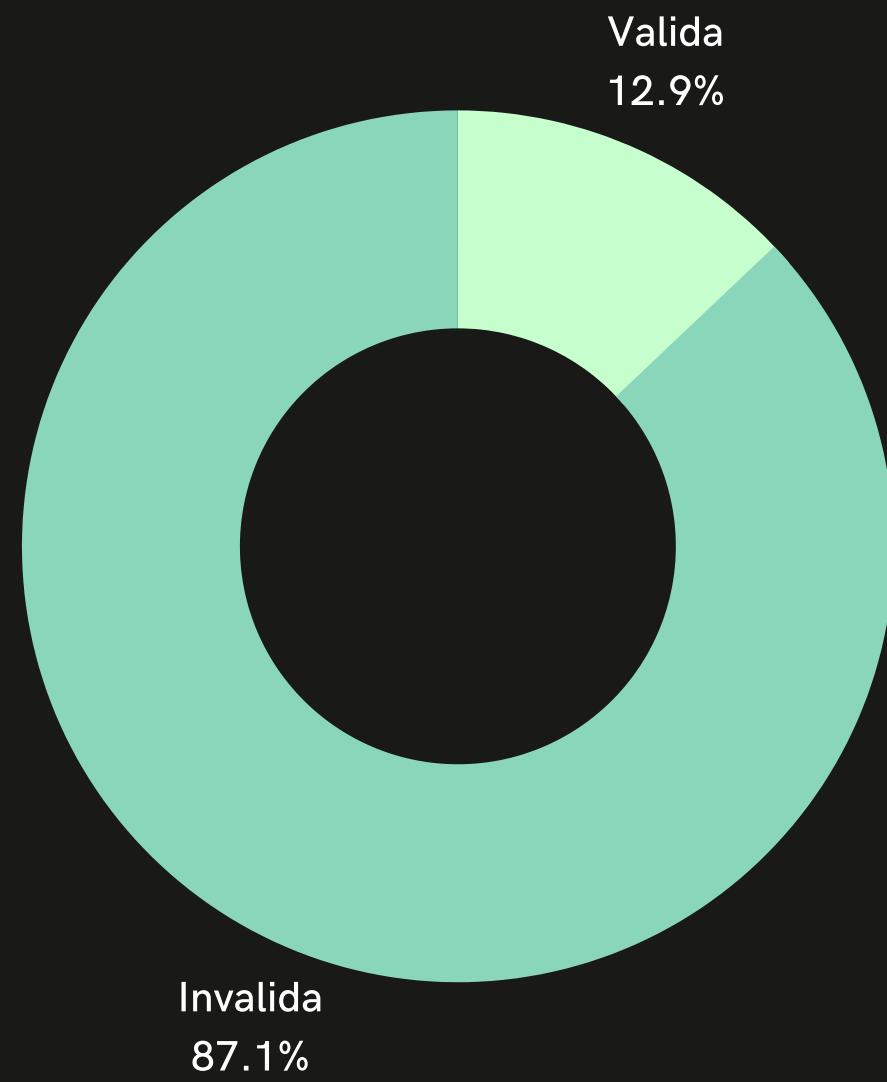
Total de ofertas realizadas en las últimas dos generaciones: 16878

Total de ofertas válidas en las últimas dos generaciones: 0

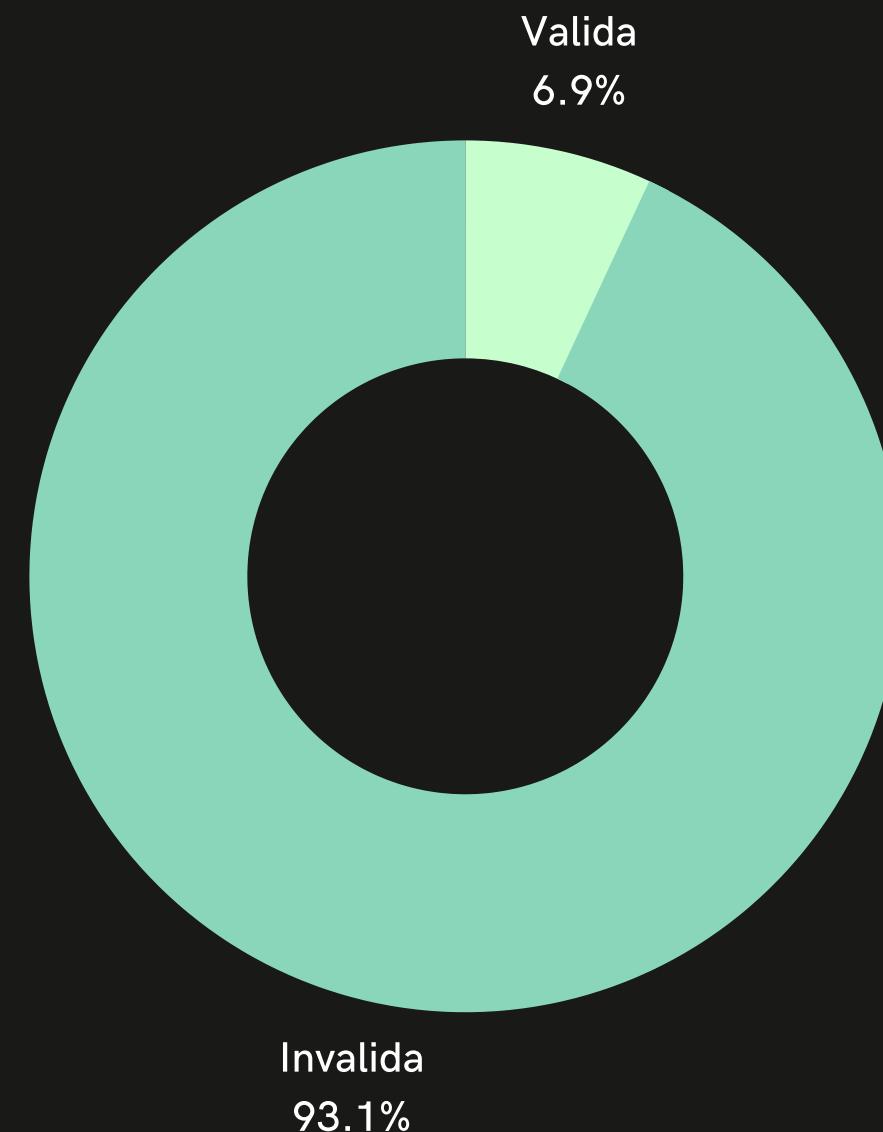
# Resultados del Entrenamiento

---

Acciones de las generaciones 19-20



Acciones de las generaciones 0-1



# Aspectos Importantes del Entrenamiento

## Los Agentes Inteligentes si fueron capaces de ganar

```
CONFIGURING GAME 0 EVALUATION
    Action logger Gen_searl_3_Game_0_Actions.log LOADED
    Trade logger Gen_searl_3_Game_0_Trade.log LOADED
    Players to take part in game 0 : [(44, 'smart'), (41, 'fixed'), (36, 'smart'), (43, 'smart')]
STARTING GAME 0
WINNER GAME 0 : 43 : Complete_winner
Fitness values : [0.30205990895038953, -0.8645171797547172, -0.3491680371404425, 0.9896565530444219]
```

## El ganador no siempre es élite

```
CONFIGURING GAME 1 EVALUATION
    Action logger Gen_searl_1_Game_1_Actions.log LOADED
    Trade logger Gen_searl_1_Game_1_Trade.log LOADED
    Players to take part in game 1 : [(20, 'smart'), (21, 'smart'), (14, 'fixed'), (12, 'fixed')]
STARTING GAME 1
WINNER GAME 1 : 12 : Complete_winner
Fitness values : [-0.33684058173582443, 0.8424813309332535, 0.2134957517687192, 0.4424835453586708]
```

```
CONFIGURING GAME 0 EVALUATION
    Action logger Gen_searl_15_Game_0_Actions.log LOADED
    Trade logger Gen_searl_15_Game_0_Trade.log LOADED
    Players to take part in game 0 : [(174, 'smart'), (170, 'smart'), (164, 'smart'), (172, 'smart')]
STARTING GAME 0
WINNER GAME 0 : 174 : Complete_winner
Fitness values : [0.022536426828798212, -0.8893244651275316, 0.24731928286538396, -0.023420290437567753]
```



# Población para los Experimentos

## Agente élite

Agente élite con mayor resultado de ajuste de la última generación

## Agente aleatorio

Agente escogido de manera aleatoria de la última generación

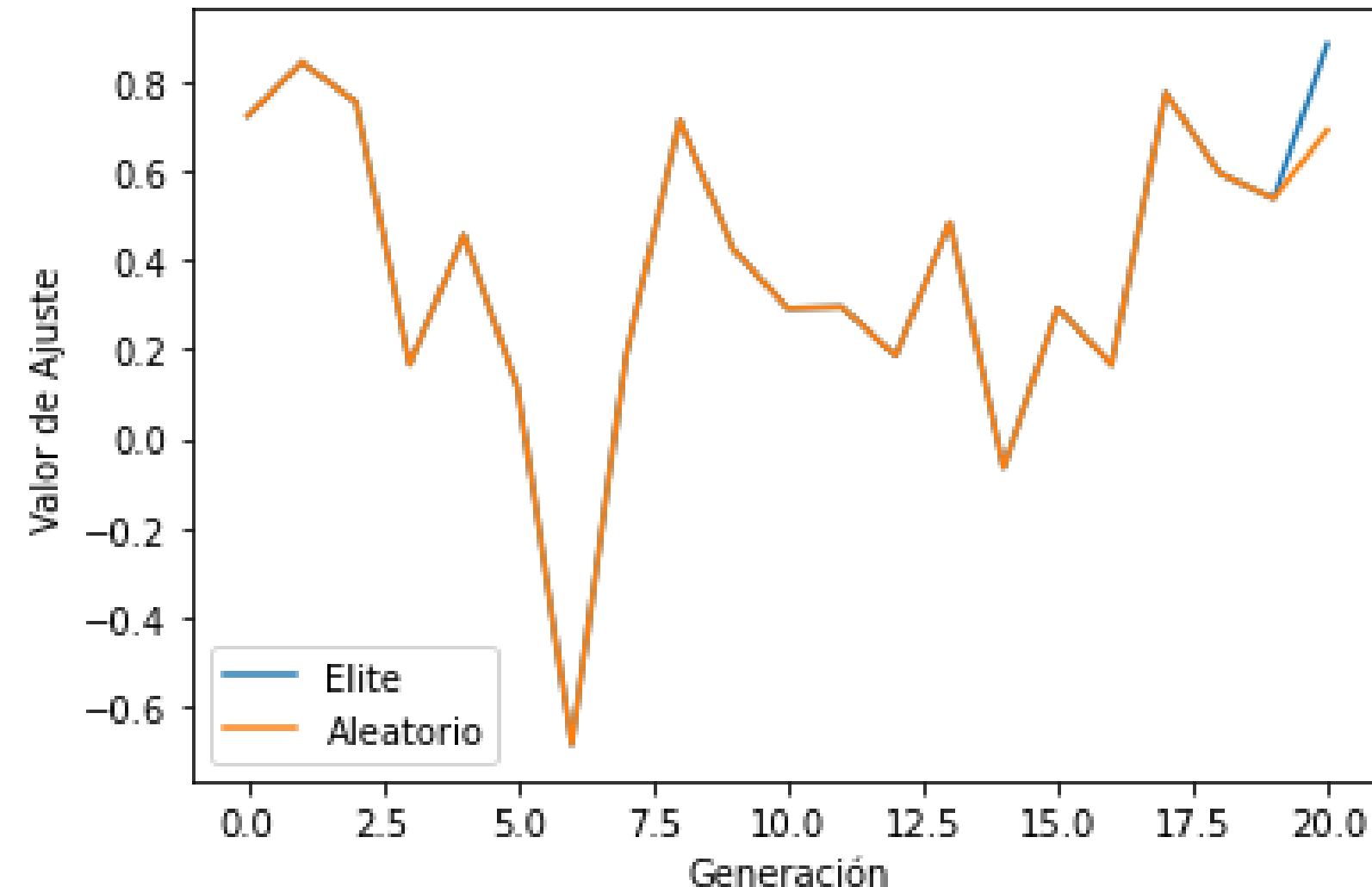
## Agente Fijo 0

Agente de política fija que prioriza los ferrocarriles y el set de color azul oscuro

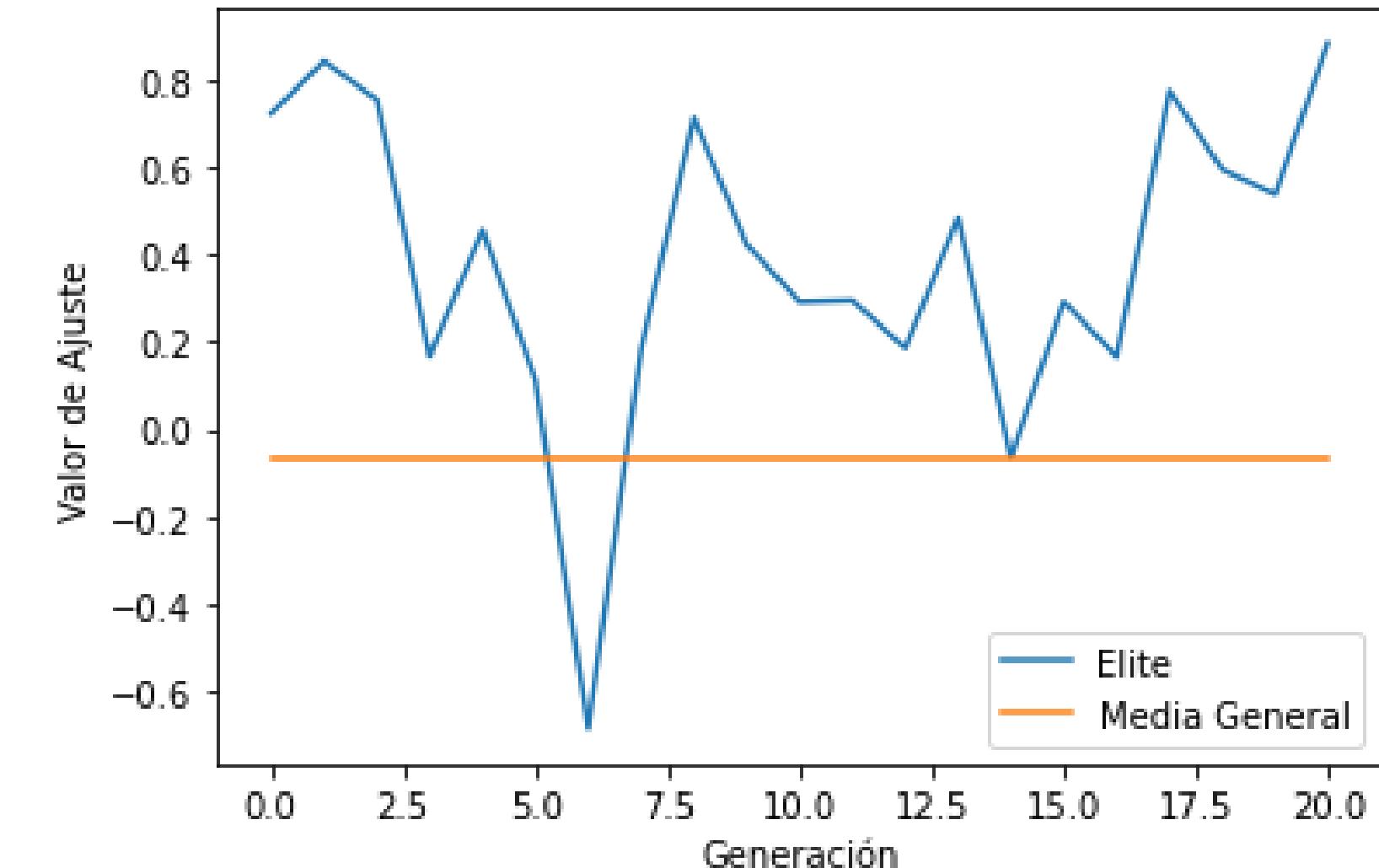
## Agente Fijo 1

Agente de política fija que prioriza los sets de color celeste y naranja

# Analizando la historia evolutiva: Agente élite vs Agente aleatorio



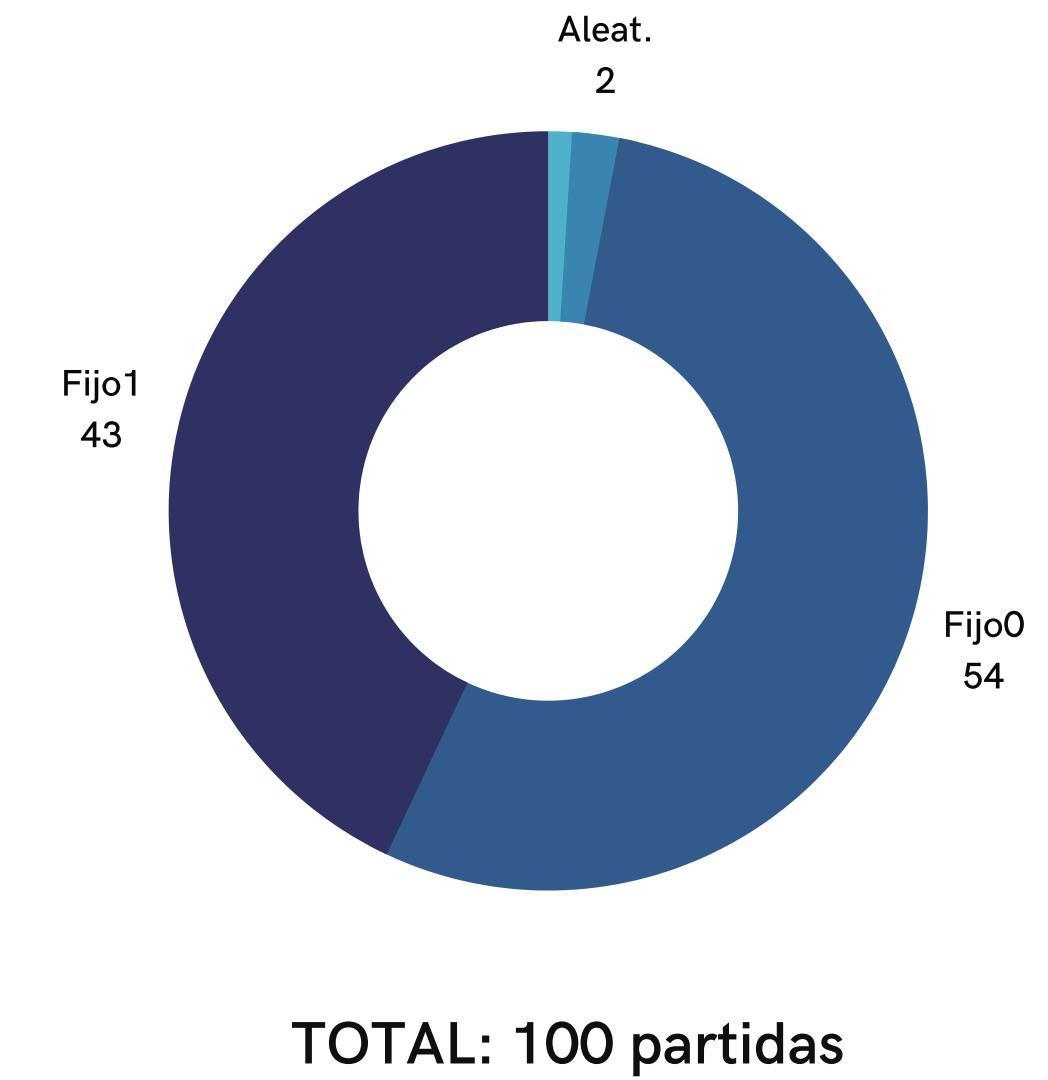
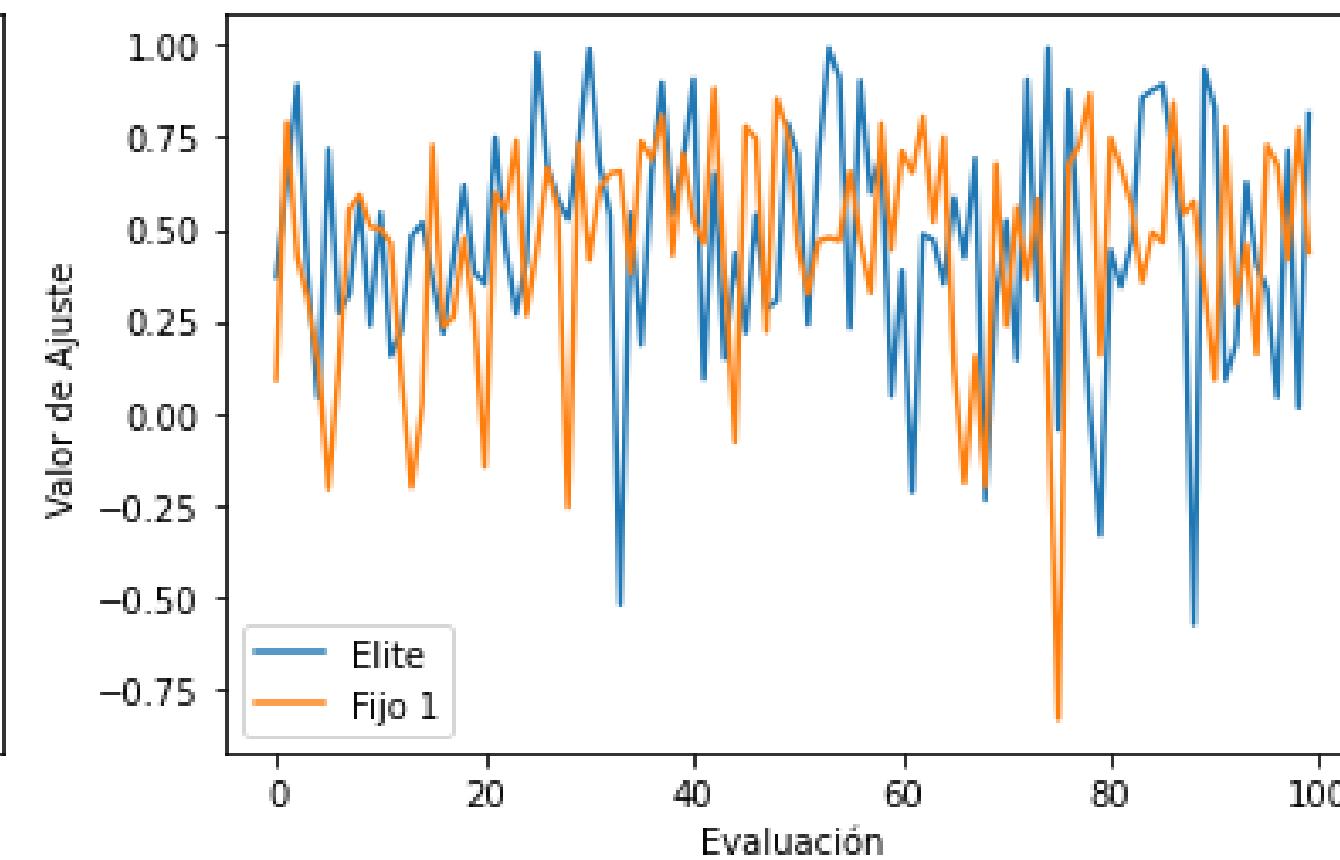
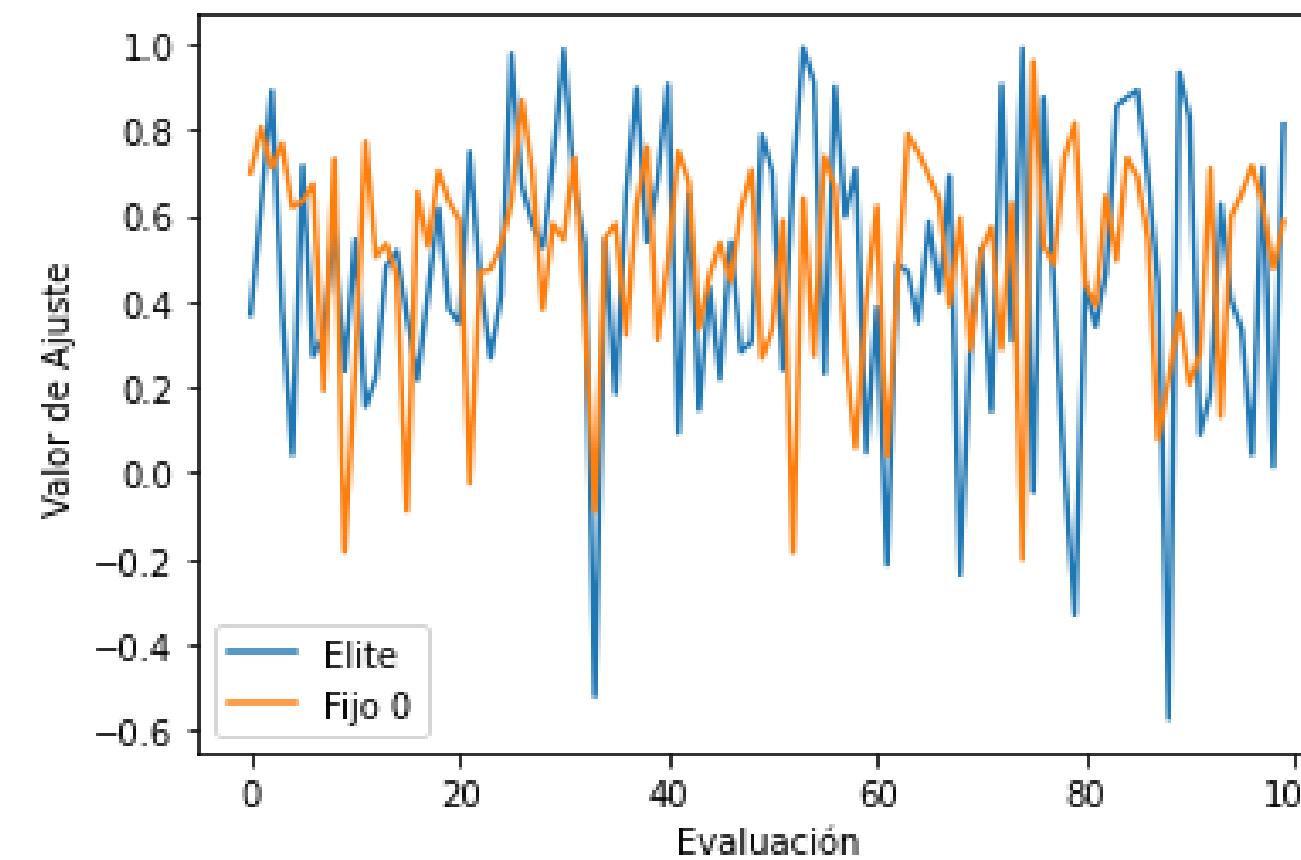
Historial evolutivo de los Valores de  
Ajuste: Elite vs Aleatorio



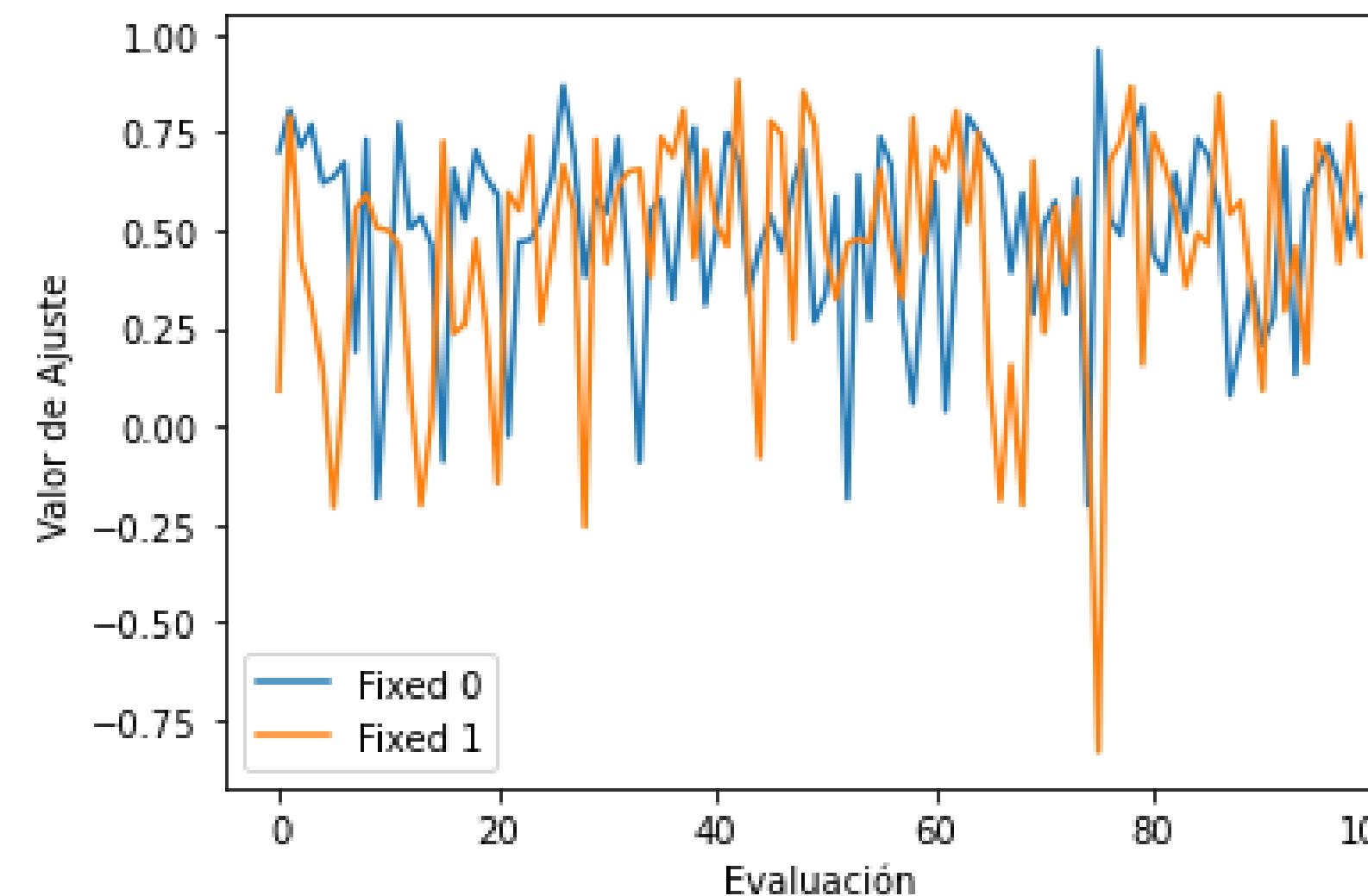
Historial evolutivo de los Valores de  
Ajuste: Elite vs Media

Media de victoria: 0.476

# Resultados de los Experimentos



# Resultados de los Experimentos

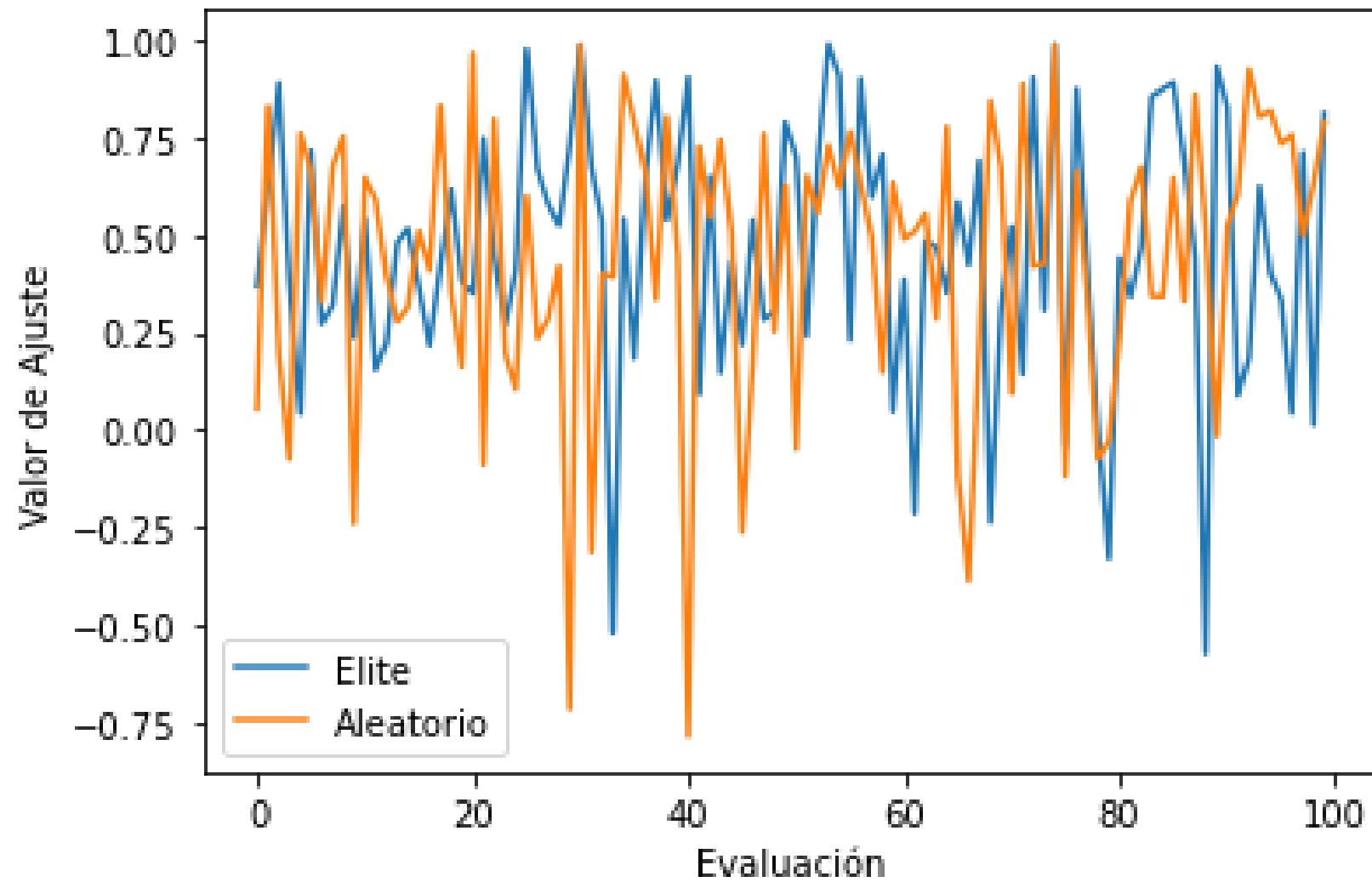


Valores de Ajuste: Fijo 0 vs Fijo1

Media de Valor de Ajuste:

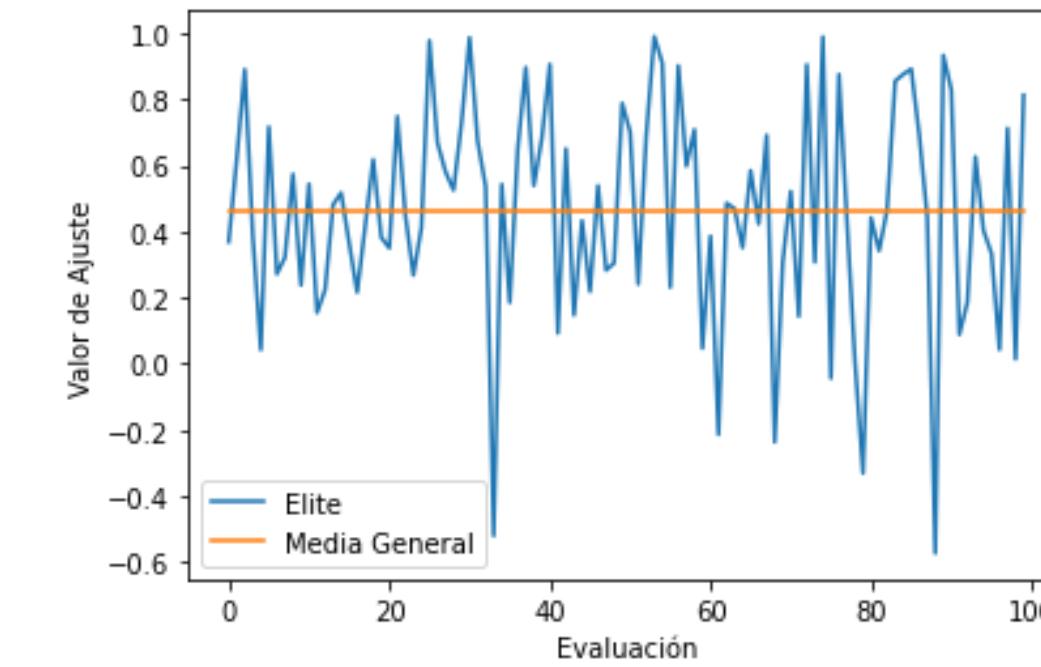
- Fijo 0: 0.4979
- Fijo 1: 0.4458

# Resultados de los Experimentos

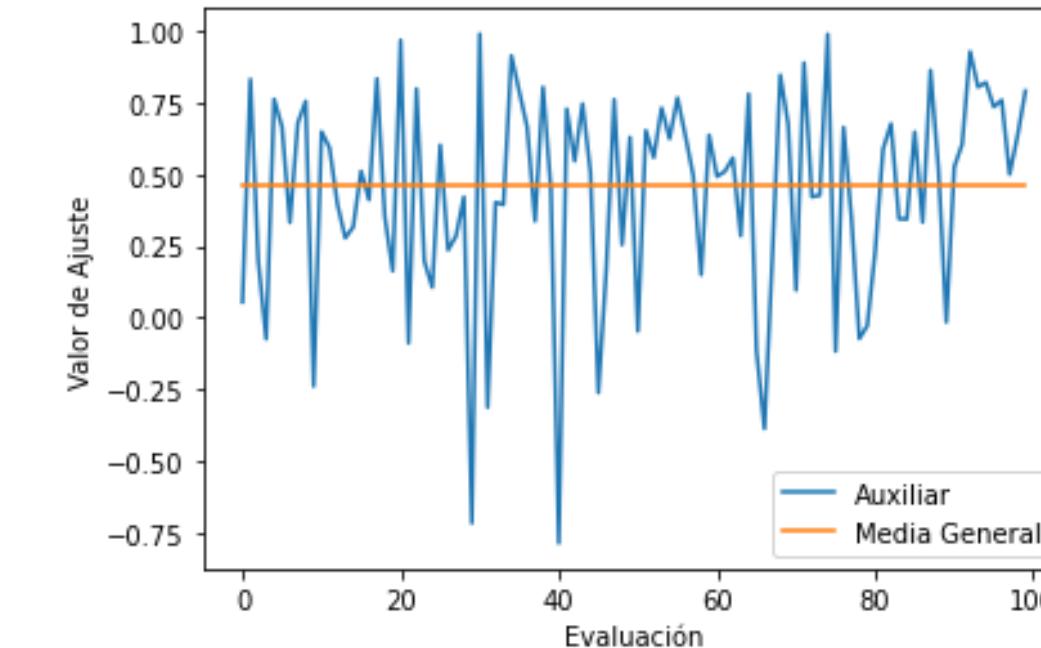


Media de Valor de Ajuste:

- Elite: 0.4611
- Aleatorio: 0.4424



Valores de Ajuste: Elite vs Media General



Valores de Ajuste: Aleatorio vs Media General

# Conclusión y Recomendación

Una Función de Recompensa General puede no resultar eficiente en casos con decisiones compuestas.

Asilar las decisiones del agente en dos redes neuronales diferentes, en la medida de lo posible, dió buenos resultados.

Puede que el modelo eficiente se encuentre en un planteamiento híbrido, pero con un planteamiento diferente.

Una solución puede ser Funciones de Recompensa específicas para cada acción.