Trabajo Corto: Algoritmos Genéticos

Informe de resultados

Inteligencia Artificial, GR 1

Prof. Ing. Kenneth Obando R.

Instituto Tecnológico de Costa Rica

Julián Rodríguez Sarmiento, c.2019047635

Samuel Valverde Arguedas, c.2022090162

II Semestre 2025

Índice

1. Resumen ejecutivo.

Car", un problema clásico de control con recompensas escasas. El objetivo fue eEste trabajo aplica algoritmos genéticos para aprender una política que resuelva el problema de "Mountain volucionar una política con una capa oculta que, a parte del estado del carro este seleccione acciones efectivas para llegar a los más alto de la montaña en la menor cantidad de pasos.

Se implementó un AG con población de 30 individuos, 50 generaciones, elitismo proporcional, "crossover" de punto, la mutación gaussiana por generación y un "matin pool". El fitness se definió como el retorno promedio en 3 episodios por individuo.

2. Introducción (motivación, fundamentos de AG).

MountainCar-v0 es un problema clásico de algoritmos genéticos de recompensas escasas y dinámica no lineal, el carro debe retroceder para ganar impulso y luego llegar a la cima de la montaña. Se exploró la búsqueda evolutiva sobre el espacio de políticas esto porque entrenar por gradiente puede ser sensible a la forma de recompensa del problema.

Un Algoritmo Genético mantiene una población de soluciones (cromosomas), evalúa su fitness, y aplica selección, combinación y mutación para generar nuevas soluciones, preservando diversidad mediante mutación y presión selectiva mediante elitismo.

3. Descripción de la modalidad escogida (juego o arte).

"MountainCar-v0" es un entorno clásico de control: un carro con motor débil debe subir una colina, pero para lograrlo primero tiene que retroceder y tomar inercia. El estado es continuo y bidimensional la posición y velocidad del carro; en Gymnasium suele estar acotado. El espacio de acciones es discreto con tres opciones: empujar izquierda (0), nada (1) o derecha (2). El episodio termina al alcanzar la meta o al agotar el límite de pasos.

La recompensa es −1 por paso hasta terminar el episodio, lo que incentiva llegar a la meta en el menor número de pasos posible. La dinámica obedece a una física simple: aplicar fuerzas pequeñas sobre el carro influye en su velocidad y, por la pendiente, el carro puede aprovechar la gravedad para impulsarse. En variantes del entorno puede exigirse una velocidad objetivo mínima al llegar a la meta "goal_velocity, lo que

incrementa la dificultad. En resumen, resolver “MountainCar-v0” requiere descubrir una estrategia no lineal (oscilar hacia atrás y adelante) más que simplemente “avanzar hacia la cima”.

4. Diseño del cromosoma y función de fitness.

4.1 Diseño del Cromosoma

Cada individuo del algoritmo genético está representado por un cromosoma codificado como un vector real. Dicho vector almacena los pesos de una red neuronal simple que actúa como la política del agente en el entorno MountainCar-v0.

- **Entrada:** el estado del entorno compuesto por posición y velocidad (obs_size = 2).
- **Capa oculta:** 8 neuronas, con activación ReLU.
- **Salida:** tres acciones posibles (action_size = 3): acelerar a la izquierda, no hacer nada o acelerar a la derecha.

El cromosoma concatena todos los pesos de la red en un único vector:

1. **Pesos de la capa entrada → oculta:** matriz de tamaño 2 × 8 = 16 genes.
2. **Pesos de la capa oculta → salida:** matriz de tamaño 8 × 3 = 24 genes.
3. **Total:** 40 genes por individuo.

De este modo, cada cromosoma puede reconstruirse en dos matrices de pesos (**W1** y **W2**) que definen la política del agente.

4.2 Función de Fitness

La función de fitness mide la calidad de un cromosoma en el entorno. En MountainCar, la recompensa del entorno es:

- -1 por cada paso que el agente ejecuta.
- Si nunca alcanza la cima en 200 pasos, recibe el mínimo total de -200.
- Si llega antes, la recompensa es mayor (por ejemplo, llegar en 86 pasos otorga -114).

Para calcular el fitness de un cromosoma:

1. Se ejecuta el agente en el entorno durante *N* episodios (en nuestro caso, 3).
2. Se acumulan las recompensas obtenidas en cada episodio.
3. El valor final del fitness es el **promedio de recompensas**:

$$fitness(cromosoma) = \frac{1}{N}\sum_{i=1}^{N} recompensa_i$$

Esta estrategia reduce la varianza y evita que un individuo parezca bueno por pura suerte en un episodio aislado. El objetivo del GA es maximizar este fitness, lo cual equivale a encontrar políticas que alcancen la cima en el menor número de pasos.

5. Parámetros y configuración experimental.

El entorno en el que se ejecuta es “MountainCar-v0” de Gymnasium con “goal_velocity=0.0”. A continuación la configuración del algoritmo genético implementado.

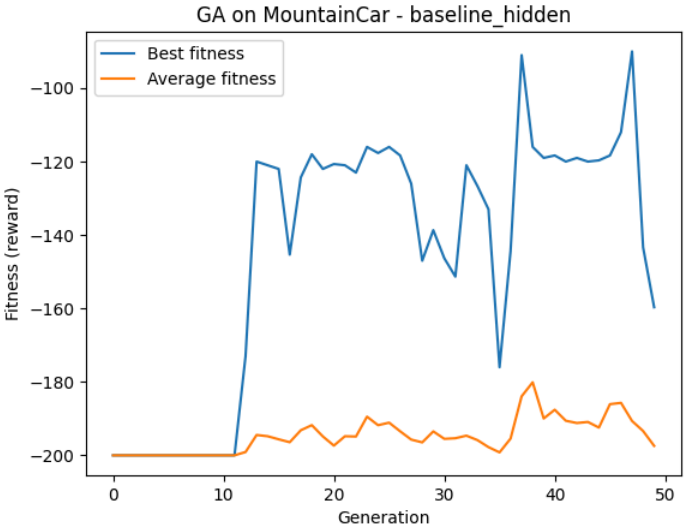
- **Población:** 30 individuos.
- **Generaciones:** 50.
- **Selección:** Torneo binario sobre el mating pool.
- **Mating pool:** 25% mejores por fitness.
- **Crossover:** 1 punto, prob. 0.7.
- **Mutación:** gaussiana N(0,0.1) por gen con prob. 0.1.
- **Elitismo:** 5% de mejores pasan intactos.
- **Evaluación por individuo:** 3 episodios (promedio).

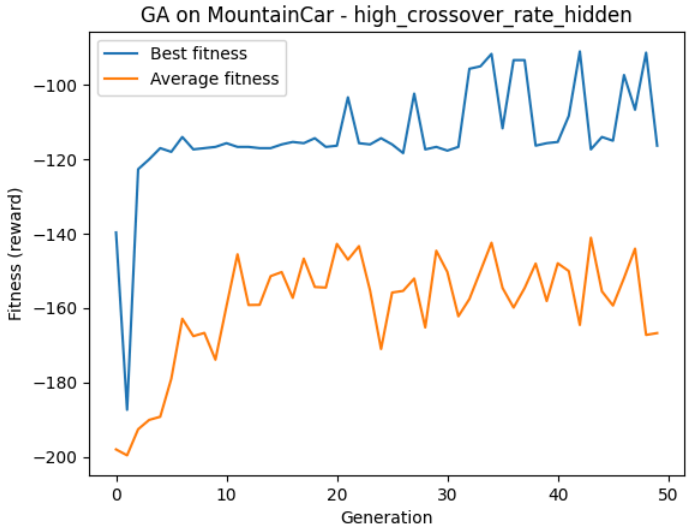
Para la configuración experimental se utilizó una corrida base para ver las diferencias al cambiar ciertos parámetros, los parámetros que se cambiaron fueron, el tamaño de la población, la mutación y el “crossover rate”. Para el experimento base se utilizó el algoritmo propuesto implementado, entonces para el primer experimento se cambió la población a 15 individuos por generación, el segundo va a variar con una mutación más alta de 0.3 y para el último experimento se va a cambiar el cruce por uno mayor de 0.9.

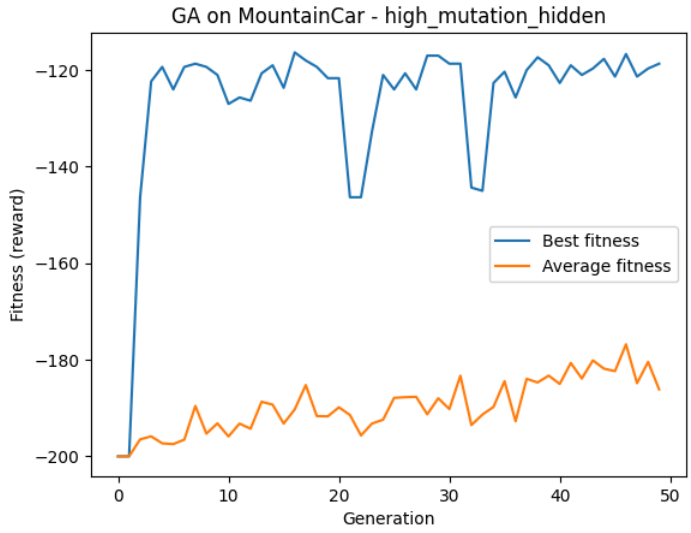
6. Resultados y visualizaciones.

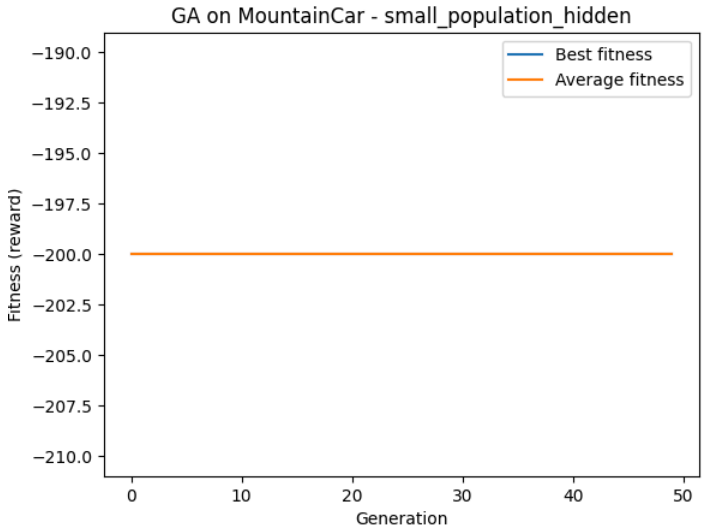
A continuación la tabla de resultados que resume los experimentos y sus respectivas gráficas.

Configuración	Best global	Gen	AVG (mejor)	Gen	Best(Final)	AVG (Final)
Base	-90.00	48	-180.11	39	-159.67	-197.42
Baja población	-200.0	--	-200.00	--	-200.00	-200.00
Alta mutación	-166.33	17	-176.80	47	-118.67	-186.13
Alto cruce	-91.00	43	-141.09	44	-116.33	-166.71






GA on MountainCar - high_mutation_hidden
Fitness (reward)
Generation
Best fitness
Average fitness
−120
−140
−160
−180
−200
0
10
20
30
40
50


GA on MountainCar - small_population_hidden
Fitness (reward)
Generation
Best fitness
Average fitness
−190.0
−192.5
−195.0
−197.5
−200.0
−202.5
−205.0
−207.5
−210.0
0
10
20
30
40
50

7. Discusión crítica y análisis.

Ejecutamos cuatro configuraciones variando población, tasa de mutación y tasa de cruce. En la línea base, muestra una dinámica típica en tareas con recompensa escasa, un largo estancamiento inicial en -200 seguido por saltos puntuales cuando aparece una política que logra acercarse a la conducta correcta. tras varias generaciones emergió un individuo destacado (best = −90 en la Gen 48), aunque el promedio poblacional se mantuvo alrededor de −180 y finalizó cerca de −197, reflejando baja consistencia, la propagación de combinaciones favorables es irregular, aparecen picos sin arrastre del promedio. El rol de elitismo fue clave, sin elitismo se observó un estancamiento de -200, con este aspecto (≥1 o 5%) el progreso se conserva y acelera la convergencia.

La configuración con población pequeña no mostró progreso (best y avg en −200 durante toda la corrida), evidenciando que la falta de diversidad impide la exploración efectiva del espacio de políticas. Con mutación alta (p=0.3) observamos descubrimiento temprano de buenos individuos (best = −116.33 en Gen 17) y un promedio máximo de −176.80, lo que confirma mayor exploración a costa de mayor varianza. Con tan pocos individuos por generación las trayectorias evolutivas colapsan rápido, el torneo tiende a usar las soluciones mediocres y con la mutación que se utilizó no alcanza para llegar a esas políticas prometedoras.

Al triplicar la probabilidad de mutación por gen, aumenta la exploración: aparecen buenos individuos antes (*Best* = −116.33 ya en la gen 17) y se observan mejoras persistentes del promedio (mejor *Avg* ≈ −176.80 en la gen 47; cierre en −186.13). La lectura es consistente con la teoría: más mutación inyecta variación que ayuda a escapar del estancamiento y a cubrir mejor el espacio continuo de pesos. La contracara es que la mutación alta también rompe combinaciones previamente útiles, por lo que el *Avg* no sube tanto como en la configuración de cruce alto

Incrementar la tasa de cruce a 0.9 elevó notablemente la calidad promedio de la población (mejor Avg = −141.09, Gen 44) superior al promedio máximo del experimento base y al que tenía una mutación alta, mantuvo un best competitivo (−91.00, Gen 43), cercano al récord del base(−90.00). Esto sugiere que la recombinación más frecuente disemina combinaciones útiles de genes por la población, mejorando la robustez global aunque no siempre supere el mejor valor individual absoluto. En conjunto, la evidencia empírica indica que la combinación de tasa de cruce alta con mutación moderada ofrece un buen compromiso entre exploración poblacional y desempeño final.

Comparando los resultados de los experimentos, tomados en conjunto dan lugar al equilibrio entre exploración y explotación en los algoritmos genéticos.

- **Población**: muy pequeña da lugar a poca diversidad genética que deriva en que no haya progreso. Con 30 individuos se alcanza un tamaño que permite explorar y seleccionar logrando resolver el problema.

- **Mutación**: alta da lugar a más exploración y descubrimiento temprano, es decir, mejores *Best* relativamente pronto, pero menor estabilidad poblacional, es decir, promedios más modestos.

- **Crossover**: alto da lugar a diseminación eficaz de bloques buenos, conjuntos de genes con promedio mucho mejor, con *Best* competitivo.

- **Baseline**: suficiente para encontrar outliers muy buenos, pero con difusión irregular teniendo un promedio pobre al final, lo que evidencia que la tasa de cruce por sí sola (0.7) no maximizó la transferencia de información genética útil.

8. Conclusiones y trabajo futuro.

Un AG simple, con elitismo proporcional y política 2→8→3, mejora de forma consistente el retorno en “MountainCar-v0” y supera el estancamiento de la variante lineal. El enfoque demuestra que, aun sin gradientes, la búsqueda evolutiva puede encontrar políticas efectivas en control discreto.

Para trabajos futuros se propone dificultar el “goal_velocity>0” aplicando límites de tiempo más estrictos, también realizar evaluaciones comparando variantes sin elitismo y con diferentes tamaños del “mating pool”, tasas de mutación y esquemas de “crossover”. También se puede incorporar normalización de observaciones junto con escalado o *clipping* de pesos; aumentar los episodios por individuo (p. ej., 5–10) para reducir la varianza del fitness.