Algoritmos Genéticos en Juegos y Arte

Valeria Gómez
Escuela de Ingeniería
en Computación
Tecnológico de Costa Rica
Puriscal, San José
vagomez@estudiantec.cr

Carlos Venegas
Escuela de Ingeniería
en Computación
Tecnológico de Costa Rica
San Rafael, Cartago
cavenegas@estudiantec.cr

I. Resumen Ejecutivo

El presente informe describe la implementación de un algoritmo genético para optimizar el control de un carro en el entorno simulado MountainCar-v0 de Gymnasium. Este entorno presenta el desafío de controlar el movimiento de un carro que debe ser impulsado para ascender una colina y alcanzar una meta situada en la cima. El objetivo del trabajo es encontrar una política de control que permita al carro alcanzar la meta de la manera más eficiente posible utilizando un algoritmo genético.

El algoritmo genético evoluciona una población de cromosomas que representan diferentes políticas de control, ajustando sus parámetros a lo largo de varias generaciones mediante operadores de selección, cruce y mutación. La función de fitness se utiliza para evaluar el rendimiento de cada cromosoma en base a la recompensa acumulada y la posición máxima alcanzada por el carro durante la simulación.

Se realizaron tres configuraciones experimentales, cada una con diferentes valores para los parámetros del algoritmo genético, tales como el tamaño de la población, la tasa de mutación y el número de generaciones. Las configuraciones fueron diseñadas para explorar el balance entre exploración y explotación en la búsqueda de la política óptima. En la configuración 1, se priorizó la explotación de las soluciones con una población pequeña y evaluaciones extensivas; la configuración 2 favoreció la exploración agresiva con una población más grande y una tasa de mutación más alta; y la configuración 3 representó un balance entre los recursos computacionales y el rendimiento del algoritmo.

Los resultados obtenidos muestran que el algoritmo genético es capaz de encontrar políticas de control que permiten al carro avanzar más rápidamente hacia la meta, demostrando la efectividad del algoritmo genético para resolver problemas de control en entornos dinámicos y complejos. Este trabajo destaca la aplicabilidad de los algoritmos genéticos en tareas de optimización en tiempo real, como el control en entornos simulados, y proporciona un análisis comparativo de cómo distintos parámetros afectan el rendimiento del algoritmo en la tarea de control del carro.

II. Introducción

Los algoritmos genéticos son una clase de técnicas de optimización basadas en los principios de la evolución biológica. Estos algoritmos pertenecen al campo de la inteligencia computacional y se inspiran en los procesos naturales de selección, cruce y mutación para encontrar soluciones a problemas complejos. En un algoritmo genético, una población de soluciones potenciales se evoluciona iterativamente mediante la aplicación de operadores genéticos, como la selección de individuos, el cruce de cromosomas y la mutación, para mejorar las soluciones a lo largo de varias generaciones. Este enfoque es especialmente útil para resolver problemas donde las soluciones exactas son difíciles de obtener mediante métodos convencionales, como en entornos con gran cantidad de variables o condicionales no lineales.

En el contexto de este trabajo, se aplican algoritmos genéticos en un entorno simulado, específicamente en el problema de control de un carro en el entorno MountainCar-v0 proporcionado por Gymnasium. Este entorno presenta el desafío de controlar el movimiento de un carro que debe ser impulsado para ascender una colina y alcanzar una meta ubicada en la cima. El problema se complica por el hecho de que el carro comienza en el fondo de un valle, lo que requiere que la política de control encuentre una forma eficiente de acelerar el carro y superar la montaña.

El objetivo de este trabajo es implementar un algoritmo genético para optimizar la política de control de este carro. El algoritmo genético se utiliza para encontrar los parámetros que mejor definan la política de control, representada por un cromosoma que determina las acciones que debe tomar el carro en función de su posición y velocidad en cada paso. Para evaluar el rendimiento de la política, se define una función de fitness que mide la recompensa acumulada en varios episodios de simulación.

Este enfoque no solo proporciona una solución eficiente al problema del control en un entorno simulado, sino que también ofrece una demostración práctica de la aplicabilidad de los algoritmos genéticos en tareas de optimización en tiempo real. A través de experimentos con diferentes configuraciones de parámetros, se busca explorar cómo los distintos factores, como el tamaño de la población, la tasa de mutación y el número de generaciones, afectan al rendimiento del algoritmo en la tarea de control del carro.

III. Modalidad escogida

Para este trabajo, se eligió el entorno MountainCar-v0 de la librería Gymnasium. Este es un problema clásico en el campo del control y optimización, donde un carro deber impulsarse para lograr salir de un valle y llegar a la meta que esta ubicada en una colina. El objetivo principal es controlar el movimiento del carro para lograr que suba hasta la cima de la montaña

El algoritmo genético que se emplea para optimizar una política de control que se representa mediante un cromosoma que determina las acciones debe tomar el carro en función de su posición y velocidad. Cada acción tomada afecta directamente la recompensa acumulada en un episodio, y se busca maximizar dicha recompensa a través de la evolución de una población de soluciones posibles.

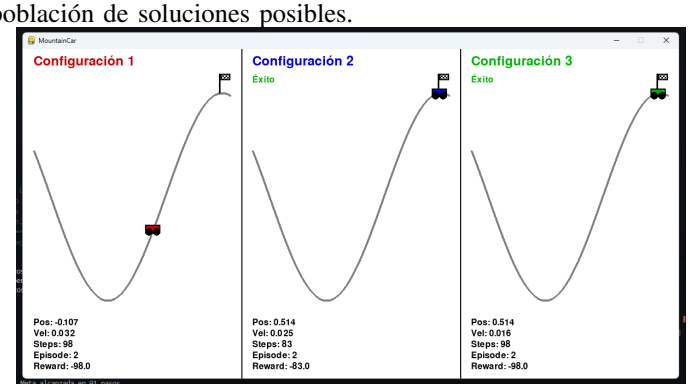

figure 1: Mountain Car desarrollado

IV. Diseño del cromosoma y función de fitness

A. Estructura del Cromosoma

El algoritmo genético se aplica para optimizar la política de control del carro. El cromosoma, que representa esta política, se compone de un vector de 18 genes, que corresponden a tres posibles acciones que el carro puede tomar en cada paso (moverse a la izquierda, mantenerse quieto o moverse a la derecha). Cada acción se evalúa en función de seis características del entorno:

- **Posición (x)**: Qué tan lejos está el carro de la meta.
- **Velocidad (v)**: La velocidad con la que el carro se mueve.
- **Posición al cuadrado (x^2)**: Para capturar la no linealidad de la dinámica.
- **Velocidad al cuadrado (v^2)**: Similarmente, para capturar la influencia de la velocidad en el comportamiento del carro.
- **Producto entre posición y velocidad (x * v)**: Para representar interacciones entre la posición y la velocidad.
- **Valor constante (1.0)**: Para facilitar el cálculo de la función de control.

Estos parámetros, combinados, forman un vector de características para cada acción. La longitud total del cromosoma es de 18 genes, distribuidos en tres bloques de seis genes, cada uno asociado con una de las tres acciones posibles.

B. Función de Fitness

La función de fitness se utiliza para evaluar el rendimiento de cada cromosoma (política de control) en el entorno MountainCar-v0. El objetivo de esta función es maximizar tanto la recompensa acumulada como la posición máxima alcanzada por el carro durante varios episodios de simulación.

Es importante destacar que el valor del fitness es negativo, ya que la función está diseñada para ser maximizada a lo largo de las generaciones. La recompensa acumulada en cada episodio se suma y, además, se premia al carro por alcanzar posiciones más altas en la montaña. Cuanto mayor sea la posición alcanzada, mayor será el valor del fitness. Así, el algoritmo genético busca maximizar el fitness, lo que se traduce en una política de control que permite al carro llegar lo más lejos posible, minimizando el número de pasos y maximizando su recompensa acumulada.

En este proceso, se evalúa el cromosoma a través de varios episodios, registrando la recompensa obtenida y la posición máxima alcanzada. La recompensa acumulada en cada episodio se suma, y la posición máxima alcanzada se actualiza si el carro avanza más lejos en cada paso. Al final de los episodios, se calcula el promedio de las recompensas y de las posiciones máximas alcanzadas, y el fitness total se determina como la suma de estos dos valores, con un factor de ponderación para la posición máxima alcanzada.

Este enfoque permite que el algoritmo genético optimice no solo la cantidad de recompensas obtenidas, sino también la habilidad del carro para avanzar lo más lejos posible, lo cual es fundamental para resolver el problema de control de manera eficiente.

V. Parámetros y configuración experimental

Los parámetros en un modelo son los valores que el modelo aprende de los datos durante el entrenamiento mediante el algoritmo genético. En el caso del juego MountainCar-v0, en la implementación realizada por el grupo se tiene el parámetro theta, este parámetro es un vector de 18 elementos que se reorganiza como una matriz de 3 filas × 6 columnas, donde cada fila corresponde a los pesos de una acción,las 3 posibles acciones que se pueden realizar son moverse a la izquierda, neutral y moverse a la derecha; las 6 características son las siguientes:

- Información básica de estado: posición que se representa con x y velocidad que se representa con v.
- Término de bias: 1.0
- Para capturar no linealidades se usan términos cuadráticos: x^2, v^2, $x * v$.

Theta define la política de control óptima mediante el siguiente proceso, para cada estado [x, v], calcula tres puntuaciones (una por acción) y selecciona la acción con mayor valor. El algoritmo genético ajusta estos pesos premiando las combinaciones que maximizan el fitness.

El objetivo del aprendizaje es que theta adquiera la estrategia de balanceo que consiste en que el carro debe balancearse hacia atrás cuando tiene poca velocidad y hacia adelante cuando tiene momentum favorable para así conseguir llegar a la meta, para que logre esto debe aprender:

- Cuando el carro está a la izquierda debe acelerar a la derecha.

- Cuando se encuentra a la derecha pero con poca velocidad debe ir a la izquierda para poder ganar momentum.
- Si tiene buena velocidad cuando va hacia la derecha debe mantenerla o acelerar para tratar de llegar a la cima de la colina.
- Debe detectar cuando está en la parte más baja del terreno o el pie de la colina para que se balancee más fuerte.

Aquí es donde los términos cuadráticos antes mencionados son fundamentales:

- x^2 ayuda a detectar qué tan lejos está el pie de la colina, es decir, el punto mas bajo en el terreno.
- El valor v^2 detecta si el carro tiene suficiente velocidad para subir la colina.
- $x * v$ captura la relación entre posición y velocidad en otras palabras, el momentum que es lo que necesita para que el carro suba la colina.

Una parte muy importante de realizar este trabajo es la experimentación con el modelo, esto con el objetivo de observar el comportamiento y desempeño del mismo ante distintas situaciones, para ello se realizaron tres configuraciones experimentales, cada una con un enfoque diferente en la búsqueda de la política óptima. Antes de explicar las configuraciones realizadas es importante aclarar cuales son los hiperparámetros que se utilizan en las configuraciones:

1) POBLACION: determina el tamaño de la población, es decir, cuantos individuos compiten en cada generación, lo individuos son vectores theta. Si hay una cantidad de baja de individuos converge más rápido y hay menos diversidad, mientras que si hay mayor población hay mas exploración pero es más lento por generación.
2) GENERACIONES: es el número de iteraciones, controla la cantidad de tiempo que corre el algoritmo genético. Si este hiperparámetro tiene un valor alto como, por ejemplo, 200 generaciones entonces, se tarda más tiempo en encontrar una solución óptima. Por otro lado, si se tiene un valor bajo (100) el algoritmo dura menos tiemmpo en encontrar una posible solución subóptima.
3) TASA_MUTACION: es la pobabilidad de que cada gen (número en el vector theta) sufra una mutación, controla cuán frecuente se introducen cambios aleatorios permitiendo introducir novedad genética y mantener la diversidad. Si este valor es bajo (0.10) se dan pequeños ajustes y con valores altos (0.35) se explora más buscando nuevas soluciones.
4) DESV_MUTACION: Es la intensidad de la mutación, indica qué tan grandes son los cambios cuando ocurre una mutación. Con valores bajos se dan cambios pequeños (0.15) y con valores altos hay cambios grandes (0.30), se dan saltos en el espacio de búsqueda.
5) ELITE: Es la cantidad de élites, esto indica cuántos mejores individuos pasan directamente a la siguiente generación. Con valores bajos (2) hay mayor presión evolutiva provocando menor preservación de las soluciones, mientras que con valores altos (8) se conservan más las soluciones buenas.
6) EPISODIOS_EVAL: son los episodios de evaluación, controla cuántas veces se evalúa cada theta para calcular su fitness. Con un valor alto (8) se obtiene una evaluación mas precisa pero más lenta, con valores bajos (3) es rápida pero con más ruido.
7) PASOS_MAX: es el límite de pasos por cada episodio, indica cuál es el tiempo máximo que tiene el carro para intentar llegar a la meta en cada episodio. Este hiperparámetro es igual en todas las configuraciones con el objetivo de ver el comportamiento del modelo con diferentes configuraciones pero con la misma cantidad de tiempo para cumplir con el objetivo del juego.

Ya que se ha aclarado la función de cada hiperparámetro utilizado, se puede explicar cada una de las configuraciones utilizadas:

- **Configuración 1:** esta se puede considerar como conservadora en cuanto a población porque, se utiliza una población pequeña de solo 30 individuos pero cuenta con evaluaciones extensivas contando con 8 episodios de evaluación por cada uno de los individuos, también tiene una baja tasa de mutación con solo 10%. Esta aproximaciónprioriza la explotación cuidadosa de las soluciones con más potencial a lo largo de 200 generaciones.
- **Configuración 2:** esta configuración es más enfocada a la exploración, tiene una población grande de 100 individuos, con una tasa de mutación del 35% lo cual es alto y tiene evaluaciones rápidas porque cuenta solo con 3 episodios. Esta estrategia favorece ala exploración agresiva del espacio de búsqueda durante 150 generaciones.
- **Configuración 3:** es una configuración balanceada porque representa un punto intermedio en todos los hiperparámetros, tiene una población media de 60 individuos, con una mutación moderada del 20% y una avaluación equilibrada de 5 episodios. Este enfoque busca optimizar el uso de los recursos computacionales en 100 generaciones.

Este diseño experimental permite comparar como con diferentes balances entre exploración y explotación pueden afectar la capacidad del algoritmo para resolver el desafío de MountainCar, donde el agente debe aprender la estrategia de balanceo para alcanzar la cima de la colina y así llegar a la meta.

VI. Resultados y visualizaciones

Para evaluar el desempeño del algoritmo genético en el entorno MountainCar-v0, se analizaron las curvas de aprendizaje de tres configuraciones experimentales durante el proceso de entrenamiento. Las métricas principales incluyeron el fitness máximo, fitness promedio y la posición máxima alcanzada por generación.

A. Desempeño General de las Configuraciones

1) Evolución del Fitness Máximo: Las tres configuraciones mostraron mejoras significativas en el fitness máximo a lo largo de las generaciones, pero con patrones distintos:

TABLE I
FITNESS FINAL ALCANZADO

Configuración	Fitness Máximo Final	Generaciones	Eficiencia
Config2	-33.49	150	Mejor absoluto
Config3	-34.33	100	Mejor eficiencia
Config1	-142.71	200	Más lenta

- **Configuración 1 (Conservadora):** Partió de -223.57 y alcanzó -142.71 en la generación 200, mostrando una mejora constante pero gradual.
- **Configuración 2 (Exploratoria):** Demostró la mejora más dramática, pasando de -141.34 a -33.49 en 150 generaciones.
- **Configuración 3 (Balanceada):** Mostró rápida convergencia inicial, mejorando de -192.93 a -34.33 en solo 100 generaciones.

2) Posición Máxima Promedio ($xmax_prom$) : Este indicador muestra la capacidad real del modelo para de resolver el problema:

- **Config1:** Evolucionó de -0.429 a 0.427, alcanzando cerca de la meta (0.5) en generaciones finales.
- **Config2:** Mostró comportamiento errático, fluctuando entre -0.404 y 0.062 sin converger consistentemente.
- **Config3:** Progresó de -0.426 a 0.201, mostrando mejora estable pero sin alcanzar el desempeño óptimo.

B. Análisis Comparativo de Curvas de Aprendizaje

1) Estabilidad vs. Rendimiento:

- **Configuración 1:** Curva de fitness suave y estable, típica de estrategias conservadoras con baja tasa de mutación (0.10).
- **Configuración 2:** Comportamiento volátil con picos de alto rendimiento pero inconsistencia, reflejando la alta exploración (tasa de mutación 0.35).
- **Configuración 3:** Balance óptimo entre exploración y explotación, mostrando mejora rápida con reasonable estabilidad.

2) Eficiencia Computacional:

- Config3 logró resultados competitivos en 100 generaciones, demostrando mayor eficiencia que Config1 (200 generaciones).
- Config2, a pesar de su población grande (100 individuos), no capitalizó su potencial de exploración en una solución robusta.

C. Rendimiento por Métrica Clave

1) Fitness Final Alcanzado: El fitness final alcanzado se muestra en Table I.

2) Capacidad de Resolución del Problema:

- **Config1:** xmax_prom = 0.427. El modelo casi resuelve el problema (meta en x=0.5).
- **Config3:** xmax_prom = 0.201. Este resultado indica progreso significativo.
- **Config2:** xmax_prom = 0.062. El modelo obtuvo un limitado éxito a pesar de buen fitness.

D. Visualizaciones Destacadas

*1) **Evolución del Fitness Máximo**:* En esta gráfica se muestra la evolución que tuvo el fitness máximo a lo largo del entrenamiento del modelo, se hace una comparación del fitness máximo de las 3 configuraciones experimentales.

Como se muestra en el gráfico, la configuración 1 es la que tiene un peor desempeño con fitness final de -142.71, mientras que la configuración 3 y la 2 tienen un mejor desempeño siendo la configuración 2 la que tiene un mejor fitness máximo al final.

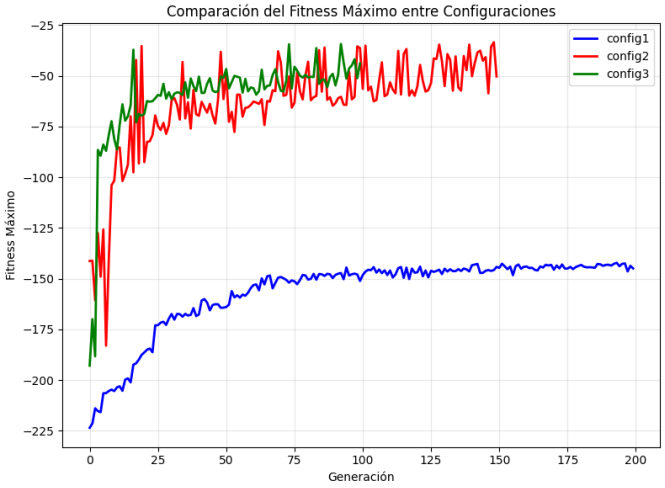

figure 2: Comparación del Fitness máximo entre configuraciones

*2) **Fitness Promedio**:* En esta gráfica se muestra la evolución que tuvo el fitness promedio a lo largo del entrenamiento del modelo, se hace una comparación del fitness promedio de las 3 configuraciones experimentales.

Como se observa en el gráfico la configuración 3 tienen mejor fitness promedio durante la mayoría de las generaciones tiene un curva más estable aunque con picos de valores pero con valores consistentemente más altos que las otras generaciones. La configuración 1 tiene el peor fitness promedio consistentemente porque tiene una curva con valores más bajos a lo largo de todas las generaciones. Por último la configuración 2 tiene un desempeño intermedio en fitness promedio y aunque cuenta con el mejor fitness máximo (-33.49), su promedio es menor que config3 esto indica mayor diversidad poblacional pero menor calidad promedio.

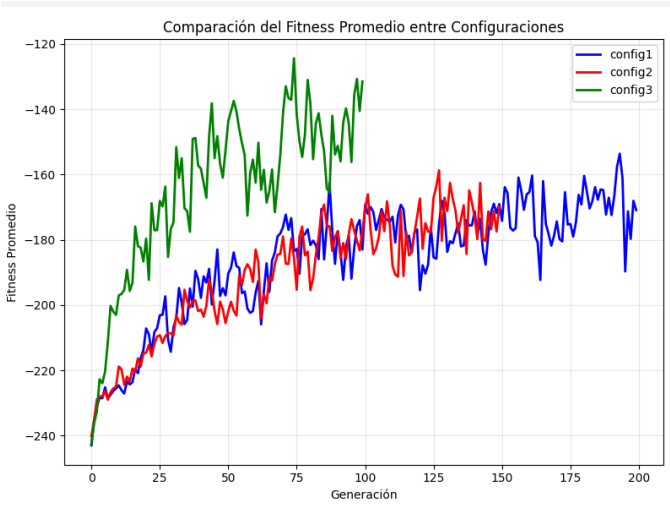

figure 3: Comparación del Fitness Promedio entre Configuraciones

*3) **Posición máxima promedio de la configuración 2**:* Esta gráfica muestra una gran desconexión entre la optimización del fitness y el desempeño práctico. Mientras Config2 logra el mejor fitness, su traducción a comportamiento efectivo es inconsistente, evidenciando sobreoptimización. Config3, aunque con fitness ligeramente inferior, demuestra mejor correlación entre métrica optimizada y desempeño real.

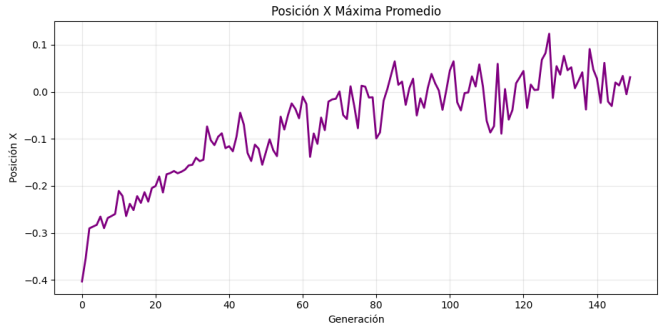

figure 4: Posición máxima promedio de la configuración 2

E. Hallazgos

1) La configuración 3 es balanceada y demostró tener la mejor relación rendimiento-eficiencia, alcanzando resultados competitivos en la mitad de generaciones que la Configuración 1.
2) Como se mostró con la configuración 2, la alta exploración genera soluciones optimistas pero inconsistentes, sugiriendo que puede beneficiarse de mayor tiempo de convergencia.
3) La evaluación robusta contribuye a la estabilidad pero a costo de velocidad computacional, esto se puede observar en la configuración 1 porque tiene 8 episodios y es más estable que las demás.
4) El indicador xmax_prom revela que Config1 fue la más efectiva en resolver el problema subyacente, a pesar de no tener el mejor fitness.

F. Conclusiones de Visualización

Las métricas complementarias como fitness máximo, promedio, y xmax_prom proporcionan una visión completa del desempeño del modelo. Mientras el fitness indica la calidad de la política aprendida, xmax_prom mide directamente la capacidad de resolver el problema de MountainCar.

VII. Discusión crítica y análisis

A. Análisis Comparativo de las Estrategias Implementadas

El estudio comparó tres configuraciones experimentales diseñadas para explorar diferentes filosofías en la optimización mediante algoritmos genéticos. La configuración 1 (POBLACION=30, TASA_MUTACION=0.10) representó un enfoque conservador, priorizando la explotación de soluciones existentes mediante una baja tasa de mutación y evaluaciones extensivas. La configuración 2 (POBLACION=100, TASA_MUTACION=0.35) mostró una estrategia exploratoria agresiva, favoreciendo la diversidad genética y la búsqueda de nuevas regiones prometedoras. La configuración 3 (POBLACION=60, TASA_MUTACION=0.20) buscó un punto intermedio, balanceando exploración y explotación.

Los resultados revelaron patrones distintivos que reflejan directamente las características de cada configuración. La configuración 1 demostró una mejora gradual y estable, pasando de un fitness de -223.57 a -142.71 en 200 generaciones. Esta trayectoria suave es característica de estrategias de baja mutación, donde la población converge lentamente pero de manera predecible. La consistencia observada se explica por la mínima interferencia de mutaciones aleatorias, permitiendo un refinamiento progresivo de las soluciones.

B. Alto Rendimiento vs Efectividad Real

La configuración 2 presentó el caso más interesante y poco intuitivo. Aunque alcanzó el mejor fitness final (-33.49), su desempeño práctico fue decepcionante, con una posición máxima promedio de apenas 0.062. Esta aparente contradicción se explica por el fenómeno de sobreoptimización del fitness: el algoritmo aprendió a maximizar la función objetivo sin necesariamente resolver el problema subyacente. La alta tasa de mutación (35%) generó una exploración intensiva pero superficial, produciendo soluciones que ”parecen” buenas en papel pero carecen de robustez en la práctica.

La volatilidad observada en configuración 2—con fitness fluctuando entre -141.34 y -33.49—refleja la naturaleza inherentemente errática de las estrategias exploratorias extremas. Cada mutación significativa puede llevar a mejoras dramáticas o retrocesos catastróficos, creando un landscape de aprendizaje impredecible. Esta inestabilidad resulta problemática para problemas como MountainCar, donde se requiere una estrategia coherente y repetible.

C. El Balance Óptimo

La configuración 3 mostró ser la estrategia más equilibrada, logrando un fitness de -34.33 en solo 100 generaciones. Este desempeño representa la mejor eficiencia computacional del estudio, demostrando que un balance adecuado entre exploración y explotación puede producir resultados competitivos con menor inversión de recursos. La tasa de mutación del

20% demostró ser suficiente para escapar de óptimos locales sin comprometer la estabilidad del aprendizaje.

La progresión de configuración 3 de -192.93 a -34.33 muestra una curva de aprendizaje ideal: rápida mejora inicial seguida de consolidación estable. Este patrón sugiere que el algoritmo encontró regiones prometedoras tempranamente y las refinó efectivamente, evitando tanto el estancamiento prematuro como la exploración excesiva.

D. Análisis de la Capacidad de Resolución del Problema

La métrica de posición máxima promedio (xmax_prom) reveló perspectivas críticas sobre la efectividad real de cada configuración. La configuración 1, a pesar de su fitness moderado, demostró la mayor capacidad práctica, alcanzando una posición de 0.427 (cerca de la meta en 0.5). Este resultado subraya la importancia de la consistencia, las soluciones estables y bien refinadas, aunque menos espectaculares en términos de fitness, pueden ser más efectivas en escenarios reales.

El pobre rendimiento de configuración 2 en xmax_prom (0.062) expone las limitaciones de las métricas proxy. El fitness function, aunque correlacionado con el éxito, no captura completamente la complejidad del problema. Esta discrepancia enfatiza la necesidad de métricas complementarias que midan diferentes dimensiones del desempeño.

E. Influencia de los Hiperparámetros en los Resultados

El tamaño poblacional demostró tener un impacto significativo en la dinámica de búsqueda. La configuración 2, con 100 individuos, teóricamente debería haber aprovechado su diversidad genética para una exploración más efectiva. Sin embargo, la combinación con alta mutación resultó contraproducente, creando demasiado ruido para una convergencia estable. Esto sugiere que la coordinación entre hiperparámetros es más importante que sus valores individuales.

La evaluación por episodios también mostró efectos notables. La configuración 1, con 8 episodios de evaluación, logró estimaciones más precisas del fitness pero a un costo computacional significativo. La configuración 2, con solo 3 episodios, sufrió de alta varianza en sus evaluaciones, posiblemente llevando a decisiones subóptimas en la selección.

F. En contexto del Problema MountainCar

La naturaleza específica de MountainCar ayuda a explicar los resultados observados. Este problema requiere una estrategia de balanceo: retroceder para ganar momentum antes de avanzar. Esta complejidad estratégica beneficia a aproximaciones que permiten un aprendizaje estable y refinado, favoreciendo a Configuraciones 1 y 3 sobre la aproximación más caótica de Configuración 2.

El panorama de fitness de MountainCar es particularmente engañoso, con recompensas inmediatas que no siempre se alinean con el objetivo a largo plazo. Esta característica penaliza a las estrategias que priorizan la optimización óptima del fitness sobre el desarrollo de políticas robustas.

VIII. Conclusiones y trabajo futuro

En este trabajo, se ha demostrado la efectividad que tienen los algoritmos genéticos en la optimización de políticas de control en entornos dinámicos como el de MountainCar-v0. A través de la experimentación con tres configuraciones distintas, se ha observado cómo el algoritmo genético puede encontrar soluciones eficientes al problema de control del carro, alcanzando la meta en diferentes grados de efectividad dependiendo de los parámetros experimentales que se utilicen.

La configuración 3, balancea exploración y explotación, demostrando ser la más eficiente, alcanzando resultados competitivos en solo 100 generaciones. Esta configuración optimizó el uso de los recursos computacionales y mostró una mejora significativa en el rendimiento del modelo.

La configuración 1, aunque fue la más lenta debido a su tamaño de población pequeño y evaluaciones extensas, fue la que logró obtener el mejor desempeño en términos de la capacidad de resolución del problema. Aunque su fitness final fue peor que el de las otras configuraciones, logró alcanzar una posición máxima promedio cercana a la meta, mostrando que una evaluación más robusta puede ser crucial para la estabilidad del aprendizaje.

La configuración 2, favoreció la exploración agresiva, pero no alcanzó el mismo nivel de estabilidad que las otras configuraciones. Aunque mostró una mejora rápida en las primeras generaciones, el modelo no logró converger de manera consistente hacia una solución óptima, sugiriendo que la alta tasa de mutación y la gran población pueden generar soluciones inestables.

La evaluación robusta con 8 episodios en la configuración 1 contribuyó significativamente a la estabilidad del modelo, aunque a costa de la eficiencia computacional. Este enfoque permite una mejor estimación del fitness, pero aumenta el tiempo requerido para encontrar una solución adecuada.

En cuanto al trabajo futuro, se sugiere experimentar con diferentes técnicas de selección y cruce en el algoritmo genético, como el cruce de un punto o el cruce uniforme, para explorar cómo estos afectan la diversidad genética y el rendimiento del modelo. También se recomienda incrementar el número de generaciones y probar configuraciones con poblaciones aún más grandes para observar cómo el algoritmo puede aprovechar la mayor diversidad genética a lo largo de un tiempo de entrenamiento mucho más largo.

Además, se podría incorporar técnicas de optimización híbrida, que combinen el algoritmo genético con otros métodos como el aprendizaje por refuerzo o redes neuronales, para mejorar la capacidad de generalización y acelerar la convergencia en problemas más complejos. También explorar la implementación del algoritmo en entornos más complejos y desafiantes, como entornos que tengan múltiples agentes o escenarios dinámicos más realistas, para validar su escalabilidad y adaptabilidad.

Finalmente, realizar un análisis más detallado del impacto de los hiperparámetros en la estabilidad y convergencia del algoritmo, especialmente la tasa de mutación, la población y el

número de episodios de evaluación, podría contribuir a mejorar el desempeño del algoritmo y ampliar su aplicabilidad a una gama más amplia de problemas de optimización y control en entornos dinámicos.