TEC | Tecnológico de Costa Rica

Inteligencia Artificial - IC6200

TC3 - Algoritmos Genéticos - Snake

Profesor:

Kenneth Obando Rodríguez

Integrantes:

Leonardo Céspedes Tenorio - 2022080602

Kevin Chang Chang - 2022039050

Fecha de Entrega: 28 de Abril

I Semestre 2025

Resumen Ejecutivo

El proyecto tiene como propósito aplicar un Algoritmo Genético (AG) para entrenar una inteligencia artificial capaz de jugar el clásico juego Snake de forma autónoma, demostrando cómo los AG pueden evolucionar estrategias eficientes sin intervención humana directa. Se diseñó un agente basado en una tabla de decisión que asocia estados del tablero con movimientos, utilizando un sistema de recompensas para evaluar su desempeño mediante un valor de fitness. A través de tres configuraciones experimentales variando la población y la tasa de mutación, se encontró que una mayor tasa de mutación (15%) y una población de 300 individuos lograron mejores resultados, alcanzando un fitness máximo superior a 400 y un promedio más elevado. Se recomienda priorizar tasas de mutación altas y poblaciones grandes para maximizar la exploración del espacio de soluciones, así como extender los períodos de entrenamiento para mejorar aún más el rendimiento. En conclusión, los resultados confirman que en problemas como Snake, donde existen múltiples óptimos locales, favorecer la exploración agresiva y mantener la diversidad genética son estrategias clave para el éxito.

Introducción

Los Algoritmos Genéticos (AG) son técnicas de optimización inspiradas en los principios de la evolución natural, como la selección, la reproducción y la mutación. Se utilizan para encontrar soluciones aproximadas a problemas complejos donde métodos tradicionales pueden ser insuficientes o inadecuados.

En este proyecto, aplicamos un AG para enseñar a jugar Snake de manera autónoma. Snake es un juego clásico en el que una serpiente se desplaza por un tablero buscando comida, creciendo con cada bocado y evitando chocar contra sí misma o contra los bordes. Programar una IA que juegue Snake plantea desafíos interesantes, como la toma de decisiones en entornos dinámicos, la planificación de movimientos y el balance entre exploración y supervivencia. La motivación detrás de este trabajo es demostrar cómo los AG pueden evolucionar estrategias de comportamiento eficientes a partir de acciones inicialmente aleatorias, mejorando progresivamente su desempeño sin intervención humana directa.

Diseño del Agente

Cada individuo o agente contiene varios parámetros y sensores que utiliza para entender su entorno al jugar el juego. Cada agente individual se reduce a una tabla de decisión (Class DescicionTable) que tiene como objetivo contener el mejor movimiento a ejecutar dado un estado del tablero. Cada tabla se conforma de la posición de la cabeza de la serpiente, la posición de la comida, la dirección a tomar desde ese estado, posición de la comida relativa a la cabeza, peligro cercano (si está cerca de chocar) y el tamaño de la serpiente. Con esto, el agente tiene información general del tablero, la posición de los objetos y la cercanía que tiene la cabeza de la serpiente a los objetos.

La idea es asignarle un "reward" value a la una partida completa que el agente ejecute y en base a la suma de "rewards" que obtenga, calcular el fitness de la tabla, consecuentemente obteniendo que tan buenos son los movimientos presentados en la tabla en cada dado estado. Con la ayuda del algoritmo genético, se desea obtener los mejores movimientos en cada estado posible para que la serpiente aprenda a ir por la comida y sobrevivir la mayor cantidad de tiempo posible.

Diseño del AG

Nuestro algoritmo genético consta de un modelo muy básico. Se crea una población con varios agentes. Cada agente genera una tabla, agregando una fila nueva por cada estado nuevo que encuentre. Para cada fila nueva que se genera, se le asigna una dirección a tomar aleatoria. La serpiente toma esa dirección mientras juega la partida y se va generando la tabla y sumando el" rewards" de dicha partida.

El "reward" castiga colisiones (-25), donde se castiga más si colisiona sin haber ganado muchos puntos (-(10/(score+1))), por cada paso que toma (-0.05), si toma muchos pasos sin haber encontrado comida (-5) y si su cabeza se aleja de la comida (-0.2). Por otro lado el "reward" incrementa si la cabeza toma un paso que se acerca a la comida (+0.3), por cada paso que toma con un puntaje alto (+(0.01 * score)), por encontrar nuevos estados (+0,2), por encontrar comida sin haber tomado muchos pasos (+máx(15 - 0.1 * stepsSinceLastFood, 5)) y por encontrar comida (+25). La suma de estos valores genera el "fitness" de un individuo el cual se utiliza para determinar cuáles individuos pasan a la fase de cruce. El "fitness" desea

determinar qué tan óptimos son los pasos tomados dado un estado específico que contienen las tablas de cada individuo en términos de ganar el juego.

Para el cruce se toma un porcentaje específico de los individuos en la población que hayan obtenido el mejor "fitness" utilizando sus tablas de decisión. El cruce combina las tablas de 2 o más individuos creando una tabla más grande con la suma de los estados encontrados por cada individuo. Para los estados que estén repetidas en ambas tablas se escoge aleatoriamente la dirección tomada entre las 2 tablas. Adicionalmente, después del cruce, se mutan a los individuos. En este proceso se recorre cada fila de la tabla, y a cada estado se le da la posibilidad de cambiar la dirección que posee a una aleatoria. Finalmente se crea una nueva población tomando estos pasos, cruzando y mutando, y se repite constantemente hasta que se comienzan a generar tablas con un "fitness" alto y capaz de jugar "Snake" de forma más óptima y adecuada.

El algoritmo genético se repite hasta encontrar una tabla que tenga los movimientos más óptimos dado un estado específico del tablero.

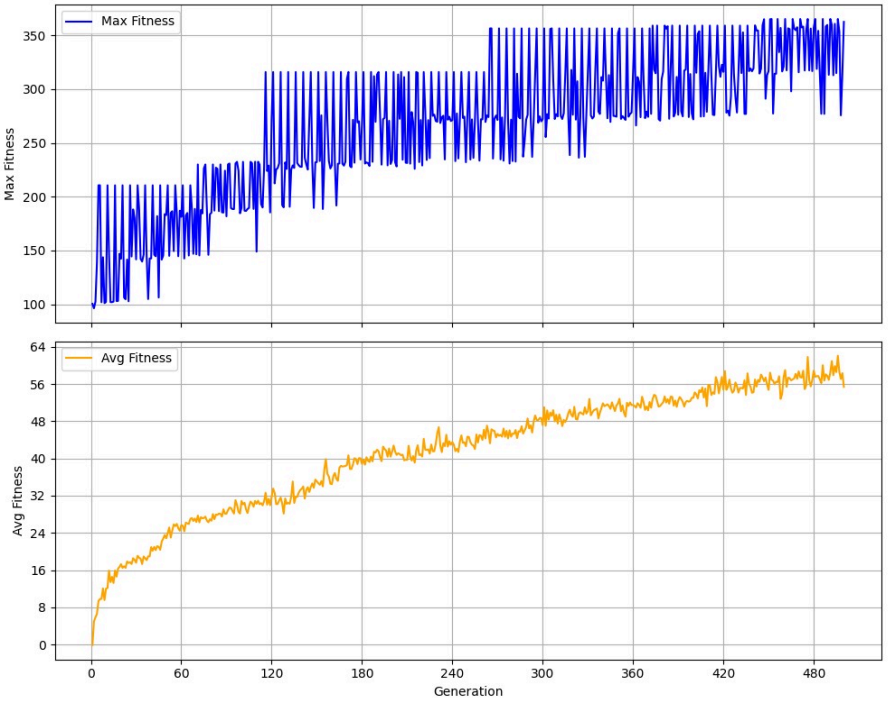
Configuración experimental

En cuanto a nuestra configuración experimental, creamos 3 ambientes diferentes donde corrimos el algoritmo con parámetros distintos y registramos los resultados de aprendizaje. Para nuestra prueba base, utilizamos una población de 300 individuos por generación, corrimos el algoritmo por 500 generaciones y utilizamos una tasa de mutación del 4% **(Prueba 1)**. En otra prueba, utilizamos una población de 150 individuos por generación, corrimos el algoritmo por 500 generaciones y utilizamos una tasa de mutación del 4% **(Prueba 2)** y finalmente en la prueba final, utilizamos una población de 300 individuos por generación, corrimos el algoritmo por 500 generaciones y utilizamos una tasa de mutación del 15% **(Prueba 3).**

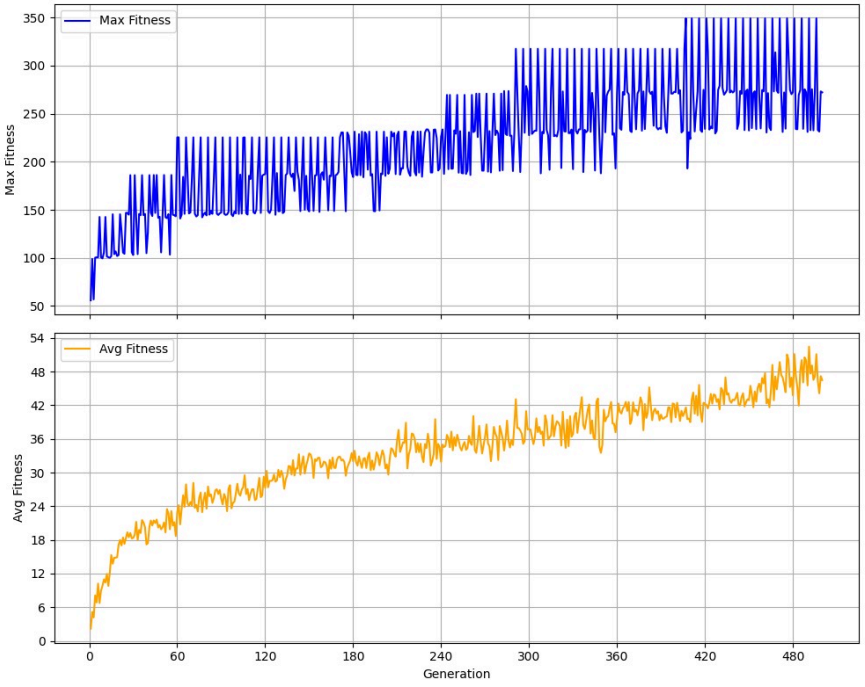
Con esto entrenamos cada agente con sus parámetros respectivos de mantenimiento el tablero y la forma en la que aparecen las frutas constante. Finalmente registramos en dos gráficos el "fitness" promedio por generación y el mejor "fitness" por generación

Opcionalmente, podemos utilizar la tabla del mejor individuo al final del entrenamiento para visualizar su aprendizaje al jugar el juego.

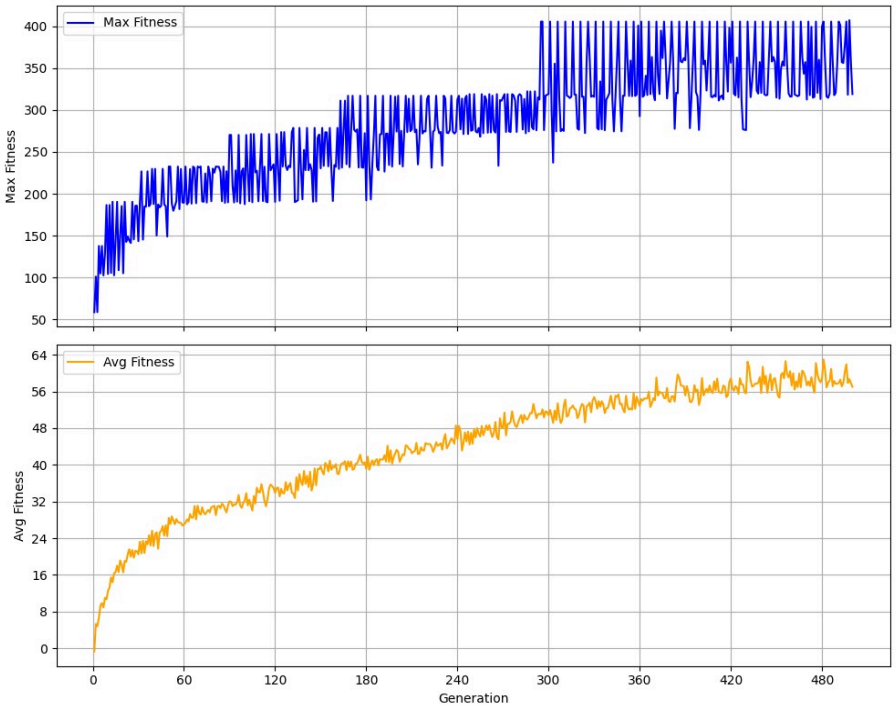
Resultados y visualizaciones



Prueba 1



Prueba 2



Prueba 3

Discusión y análisis

Como se puede observar en las tres figuras anteriores, los tres experimentos muestran un patrón de evolución similar, pero se pueden apreciar ciertas diferencias en cuanto a velocidad de aprendizaje y valores finales de fitness alcanzados. A continuación se describen los resultados obtenidos en cada una de las pruebas:

Prueba 1: En esta prueba se utiliza una población de 300 individuos y una tasa de mutación del 4%

- **Fitness máximo**: Evoluciona desde valores iniciales bajos hasta alcanzar picos de aproximadamente 350
- **Fitness promedio**: Progresa gradualmente desde valores cercanos a 0 hasta aproximadamente 60
- Se observa un salto significativo en el fitness máximo alrededor de la generación 150
- Las oscilaciones en el fitness máximo son constantes pero mantienen una tendencia creciente

Prueba 2: En esta prueba se utiliza una población de 150 individuos y una tasa de mutación del 4%

- **Fitness máximo**: Alcanza valores máximos de aproximadamente 350
- **Fitness promedio**: Alcanza aproximadamente 50 en las últimas generaciones
- Muestra un progreso más lento y menos pronunciado que la Prueba 1
- Presenta oscilaciones menos extremas, pero aún significativas

Prueba 3: En esta prueba se utiliza una población de 300 individuos y una tasa de mutación del 15%

- **Fitness máximo**: Logra los valores más altos entre todas las pruebas, superando 400
- **Fitness promedio**: Llega hasta aproximadamente 60-64, superior a las otras pruebas
- Exhibe un salto notable en el fitness máximo cerca de la generación 300
- La alta tasa de mutación parece haber proporcionado beneficios sustanciales en etapas avanzadas

Al realizar una comparación directa entre las tres pruebas se puede determinar que las poblaciones más grandes muestran un aprendizaje más acelerado en etapas iniciales, lo que sugiere una mayor exploración del espacio de soluciones. Así mismo, la prueba 1 logró un fitness promedio final superior a la prueba 2, lo que indica que la mayor diversidad genética aportada por el aspecto de una población más grande fue beneficiosa a largo plazo. La población mayor mostró oscilaciones más pronunciadas en el fitness máximo, lo que podría indicar mayor diversidad genética mantenida a lo largo de las generaciones.

Al comparar la prueba 1 y 3 se puede observar que la tasa de mutación más alta en la Prueba 3 permitió descubrir soluciones de mayor calidad, evidenciado por un fitness máximo que alcanza 400, significativamente superior a las otras pruebas. El incremento en la tasa de mutación parece haber sido particularmente efectivo después de la generación 300, donde se observa un salto sustancial en el fitness máximo. La alta tasa de mutación permitió mantener un equilibrio favorable entre exploración de nuevas estrategias y explotación de las soluciones ya encontradas, lo que se refleja en el fitness promedio más alto.
En general, se pueden identificar patrones comunes y diferencias significativas entre las tres pruebas como las siguientes: todas las pruebas muestran fases distinguibles de aprendizaje, con períodos de rápida mejora seguidos por mesetas de estabilización, las oscilaciones en el fitness máximo son una característica constante en todas las pruebas, y la pendiente de mejora del fitness promedio es más pronunciada en las etapas iniciales (primeras 100 generaciones) y luego se vuelve más gradual.

Conclusiones y trabajo futuro

- La tasa de mutación demostró ser el factor más determinante para el rendimiento final del algoritmo. El salto cualitativo observado con la tasa del 15% (Prueba 3) sugiere que en este problema específico, una exploración más agresiva del espacio de soluciones resulta beneficiosa.
- Un tamaño de población mayor (300 individuos) proporcionó beneficios tanto en la velocidad inicial de aprendizaje como en el rendimiento final, comparado con una población más pequeña (150 individuos).
- Las tres configuraciones demostraron mejora continua a lo largo de las 500 generaciones, sin evidencia de estancamiento completo, lo que sugiere que períodos de entrenamiento aún más largos podrían seguir mejorando el rendimiento.
- La Prueba 3 (población de 300 individuos con tasa de mutación del 15%) mostró los mejores resultados generales, alcanzando tanto el fitness máximo más alto como el fitness promedio más elevado, lo que la identifica como la configuración óptima entre las probadas.
- Los resultados sugieren que en problemas como Snake, donde el espacio de soluciones es complejo y contiene múltiples óptimos locales, favorecer la exploración (mayor mutación) sobre la explotación resulta ventajoso.

Enlace al Colab:

https://colab.research.google.com/drive/1oxhiSQUDJJvVWI1jhy89FfHeij2q2lwr?usp=sharing