Pac Man: Algoritmos Genéticos

Jenny King Bustamante
2023029525
jking@estudiantec.cr

Camila Thompson Sancho
2023083194
cthompson@estudiantec.cr

***Abstract*—En este trabajo se aplican los conceptos fundamentales de los algoritmos genéticos para resolver el juego de Pac-Man. El objetivo principal es desarrollar una estrategia para que el personaje logre ganar el juego, por medio de operadores genéticos como selección, cruce y mutación para mejorar su desempeño a lo largo de múltiples generaciones. Posteriormente se realiza una simulación del mejor individuo encontrado.**

Palabras clave—algoritmos genéticos, pac-man, optimización

I. Introducción

Los algoritmos genéticos (AG) son una técnica evolutiva muy efectiva de implementar en procesos de optimización en entornos observables. Su naturaleza basada en poblaciones permite aplicar un conjunto de operadores (selección, cruzamiento y mutación) que son adecuados para explorar soluciones amplias. En el caso de estudio se comprende el juego clásico de arcade Pac-Man, debido a que combina la navegación de laberintos, recolección de comida, interacción con enemigos y toma de decisiones, lo cual propicia una serie de objetivos múltiples perfectos para aplicar optimización evolutiva, donde algoritmos deterministas y de búsqueda exhaustiva no son adecuados para solucionar.

La contribución principal de este proyecto es el desarrollo de un agente basado en algoritmos genéticos que aprende una política sin supervisión directa, utilizando de referencia retroalimentación basada en parámetros ambientales como el puntaje, las posiciones en el mapa y la cercanía y colisión con enemigos. Se plantea un fitness que incorpora métodos de penalización y recompensa de acuerdo con las condiciones actuales del individuo, permitiendo evaluar su desempeño sin depender de heurísticas prediseñadas o modelos explícitos del entorno.

Además, se implementa un motor de juego completo en JavaScript que integra mecánicas esenciales. Este marco experimental facilita la realización de múltiples episodios y permite estudiar el comportamiento emergente del algoritmo genético bajo distintos hiperparámetros. Los resultados obtenidos muestran que el agente es capaz de mejorar progresivamente su desempeño, aprendiendo patrones de navegación útiles y evitando enemigos con una política derivada exclusivamente de la presión selectiva del fitness.

II. Metodología

A. Codificación de Agentes

La generación de agentes Pac-Man se codifican mediante un cromosoma de longitud L el cual está conformado por un patrón de movimientos a ejecutar desde la posición inicial. Cada gen representará una dirección a la que el agente podría desplazarse (arriba, abajo, izquierda, derecha).

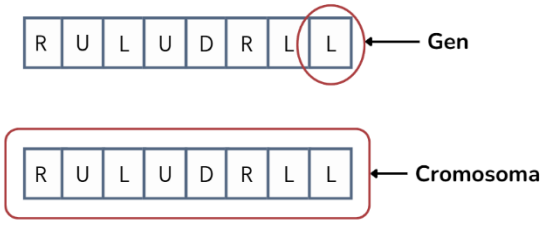

Figura 1. Genes y Cromosomas

El cromosoma completo constituye una política open-loop (predefinida), es decir, el individuo no toma decisiones adaptativas durante la ejecución, sino que sigue la secuencia codificada. Donde se define un array_size de 100 genes aleatorios al iniciar la población.

B. Operadores Genéticos

Se utilizan los operadores estándar, cada uno adaptado para conseguir los resultados deseados. Estos se detallan a continuación.

Selección

Se realiza un torneo para la selección de los padres con el objetivo de evitar la dominancia absoluta. Este torneo consiste en escoger una cantidad k de individuos al azar para competir, aquel que posea un mayor fitness será el ganador y pasarán sus genes a las próximas generaciones. Realizamos este torneo dos veces para conseguir a ambos padres.

```
FUNCTION selectParent()

  tournamentSize ← 30
  selected ← empty list

  FOR i from 1 to tournamentSize
      r ← random integer from 0 to population.length - 1
      add population[r] to selected
  END FOR

  sort selected by fitness in descending order

  RETURN selected[0]    // best individual in the tournament

END FUNCTION
```

Figura 2. Pseudocódigo de selección

CRUCE

Una vez se han seleccionado los individuos se cortan sus cromosomas en un punto c ∈ [1, L-1] seleccionado aleatoriamente para generar dos segmentos, una cabeza y una cola. Las colas se intercambian entre los dos padres para generar a los hijos. De este modo los descendientes heredan información genética de cada uno de los padres.

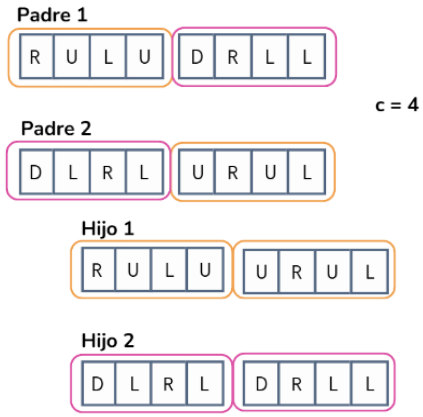

Figura 3. Cruce de 1 Punto

MUTACIÓN

La mutación se realiza con una tasa de mutación de 0.1 que remplaza el gen por un movimiento aleatorio del conjunto $G = \{U, D, L, R\}$.

Figura 4. Mutación

C. Fitness

El fitness evalúa varios objetivos: comer los puntos, explorar (visitar celdas nuevas), evitar fantasmas y completar el nivel. Se propone una función compuesta (suma ponderada), con penalizaciones fuertes para muertes y repeticiones.

A continuación, se definen unas variables que representan las recompensas y penalizaciones que se asignan dentro del proyecto, las cuales cumplen la función de incentivar el comportamiento deseado y reforzar el progreso dentro del sistema.

- C: número total de puntos comidos
- DC: número de movimientos acercándose a comida
- V: número de celdas nuevas visitadas
- R: número de celdas revisitadas
- MP: número de colisiones con pared
- E: número de movimientos a espacios vacíos
- F: número de choques con fantasmas

$$\boldsymbol{Fitness} = (\mathbf{30}\boldsymbol{C}) + (\mathbf{20}\boldsymbol{V}) + (\mathbf{10}\boldsymbol{DC}) - (\mathbf{50}\boldsymbol{R}) - (\mathbf{10}\boldsymbol{MP}) - (\mathbf{100}\boldsymbol{F}) - (\mathbf{15}\boldsymbol{E})$$

Se penaliza fuertemente la repetición de este modo se busca evitar bucles y fomentar la exploración del agente para no quedarse en una sola área del tablero. Asimismo, se utiliza un sensor de comida, en caso tener comida cerca se da una bonificación ya que el objetivo principal es conseguir estos puntos por lo que hay que buscarlos conforme el agente se movilice.

El agente debe recorrer todo el tablero, por lo que se premia en caso de llegar a celdas que no habían sido visitadas antes, de otro modo, se penalizará por cada celda repetida a la que llegué, ya que no estaría buscando completar el tablero y darías pasos repetidos.

Las colisiones con paredes también se castigan, esto para evitar movimientos inválidos que provoquen que el agente se quede pegado en un solo lugar y haciendo movimientos que no lo llevan a ningún lugar.

```
Algorithm: EvaluateFitness(individual)

Set (x, y) to Pac-Man's start position
Set score = 0
Create empty set visited

For each move in individual:

    Compute (newX, newY) based on move

    If new position is wall or out of bounds:
        score -= 10
        Continue

    If Pac-Man is near food:
        score += 10

    Update (x, y) to new position

    If collides with ghost:
        score -= 200

    If ghost is nearby:
        score -= 20

    If (x, y) was visited before:
        score -= 50
    Else:
        Add (x, y) to visited
        score += 20

        If tile has food:
            score += 30
        Else:
            score -= 15

Return score
```

Figura 5. Pseudocódigo Fitness

En caso de que el agente colisione con un fantasma, se le deducirá una cantidad significativa para que el agente no muera. Si el agente no logra superar el tablero se le asignará un 0, pero si es capaz de completarlo obtendrá un 1 y el premio máximo por ganar el juego.

D. Protocolo experimental

Para garantizar consistencia y reproducibilidad, los experimentos se ejecutaron con los siguientes parámetros:

PARÁMETRO	VALOR
TAMAÑO DE POBLACIÓN	20
GENERACIONES	50
LONGITUD CROMOSOMA	100
SELECCIÓN	Torneo (k)
CRUCE	1 Punto
MUTACIÓN	0.10
EPISODIOS EVALUADOS POR INDIVIDUO	1
SEMILLAS	123456789, 987654321, 12121212

Durante cada generación, todos los individuos se evalúan desde la misma posición inicial. Los mejores dos individuos se conservan mediante elitismo y el resto de la población se rellena con descendientes producidos por cruce y mutación. Además, se registró información del entorno:

- Versión del navegador: **Chrome 142**
- Sistema operativo: **Windows 10**
- Mapa utilizado: archivo tileMap del motor JS
- Archivos entregados: código fuente, configuraciones, logs por semilla

Para garantizar la reproducibilidad se utilizó el generador pseudoaleatorio Mulberry32. Este permite definir una semilla explícita (en este caso 123456789, 121212, 987654321) y garantiza que todas las operaciones dependientes del azar produzcan exactamente la misma secuencia de valores en ejecuciones repetidas.

III. RESULTADOS

Los experimentos realizados permitieron evaluar el desempeño del agente evolutivo a lo largo de 50 generaciones para cada una de las semillas. En la mayoría de los casos el algoritmo logró mejorar el comportamiento del agente, evidenciado por el incremento en el valor del fitness obtenido y el recorrido más eficiente a través del tablero.

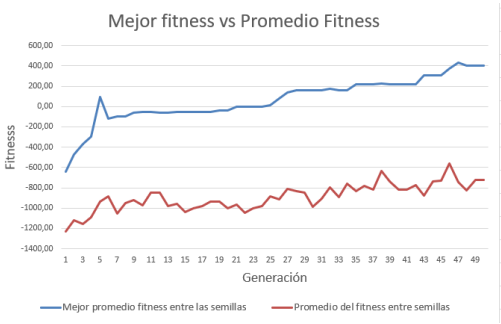

Figura 6. Gráfico comparativo del mejor fitness vs el fitness promedio

A lo largo del proceso evolutivo, los operadores genéticos, especialmente el cruce de un punto y la mutación con probabilidad 0.10, promovieron la aparición de secuencias de movimiento más estables. Esto puede observarse en la disminución gradual de la desviación estándar dentro de cada generación, lo cual indica que la población converge hacia comportamientos óptimos.

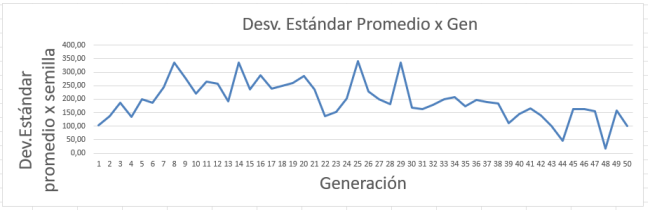

Figura 7. Gráfico comparativo de desviación estándar por generación

Un hallazgo adicional es que, aunque los individuos muestran una mejora clara generación tras generación, ninguno logró completar el tablero dentro del límite estructural de 100 acciones definido por el cromosoma. Aun así, los individuos evolucionados lograron recorrer una mayor cantidad de celdas y evitar colisiones tempranas, alcanzando niveles de rendimiento notablemente superiores a los iniciales. Los patrones convergentes evidencian que el agente aprende a priorizar rutas más despejadas, recolectar comida consistentemente y evitar regiones peligrosas del mapa.

Finalmente, se observó que una vez agotada la secuencia de 100 acciones, el agente deja de moverse. Aunque este comportamiento no afecta la evaluación del fitness (ya que el episodio finaliza al consumir todos los movimientos), sí se manifiesta como un estado inmóvil y no interactivo al finalizar la simulación visual.

A. Registro de datos

Para analizar rigurosamente el comportamiento del algoritmo, cada ejecución generó un archivo JSON mediante

un módulo denominado seed logger, el cual registró el fitness de cada individuo en cada generación, junto con estadísticas adicionales como promedio y desviación estándar. Estos registros fueron producidos de forma independiente para cada semilla utilizada y posteriormente procesados en una hoja de cálculo para obtener métricas agregadas. A partir de estos datos se calcularon, para cada generación, la media entre semillas, la desviación estándar entre ejecuciones y los valores máximos de fitness alcanzados. Este procedimiento permitió generar las gráficas comparativas utilizadas en este estudio y garantizó la consistencia del análisis.

IV. Discusión

Los resultados experimentales demuestran que el algoritmo genético es capaz de mejorar sustancialmente el desempeño del agente, incluso bajo una política open-loop en la que el individuo ejecuta una secuencia fija de acciones sin retroalimentación del entorno durante la ejecución. Sin embargo, esta misma estructura limita la capacidad del agente para reaccionar ante eventos dinámicos, como el movimiento de fantasmas o situaciones no anticipadas. Esto hace que algunos individuos, a pesar de tener secuencias eficientes en términos locales, colisionen de manera inevitable debido a la falta de adaptación en tiempo real.

Una limitación importante es la longitud fija del cromosoma (100 acciones). Dado el tamaño del tablero y la presencia de obstáculos y persecuciones, este número resulta insuficiente para permitir una exploración completa o la recolección total de pellets. En otras palabras, la política nunca podrá completar el juego en su estado actual, independientemente de qué tan óptimo sea el individuo resultante. A pesar de ello, el algoritmo sí logra generar rutas eficientes dentro de ese presupuesto limitado de acciones, lo que sugiere que un cromosoma más largo o variable podría incrementar significativamente la capacidad de exploración.

Otro factor relevante es el número de generaciones (50). Aunque las curvas de fitness muestran una clara tendencia a la mejora, el ritmo de crecimiento sugiere que un mayor número de generaciones podría permitir que el agente alcanzara comportamientos más robustos y estratégicos. No obstante, las restricciones computacionales y de tiempo impiden ejecutar experimentos de mayor escala para confirmar esta hipótesis.

Finalmente, se identificó un efecto secundario en el motor de simulación: una vez agotadas las acciones disponibles del cromosoma, el agente permanece inmóvil e inmune a los fantasmas durante el resto del episodio visual. Aunque esto no interfiere con la evaluación numérica del fitness, constituye una consideración relevante para futuras mejoras del entorno de simulación, especialmente si se desea comparar agentes basados en políticas reactivas o con retroalimentación en línea.

V. Conclusiones y trabajo futuro

El desarrollo del juego e incorporación de algoritmos genéticos permitió demostrar que se pueden conseguir resultados útiles incluso en entornos dinámicos. Por medio de los cromosomas definidos y la función del fitness deseñada, el agente fue capaz de aprender cuales rutas lo llevaban a maximizar la recolección de comida, favorecer la exploración y entender las consecuencias de diferentes conductas.

Al presentar un entorno cambiante debido a los movimientos independientes de los enemigos y el uso de un open-loop provocó que el agente no logre reaccionar a los cambios en el entorno. Lo que puede impedir encontrar soluciones optimas. A pesar de estas limitaciones se puede evidenciar que los algoritmos genéticos son un método valido y funcional, que sienta las bases para explorar implementaciones más complejas para optimizar el comportamiento y los resultados.

En trabajos futuros, con el objetivo de encontrar un agente completamente inteligente y capaz de aprender de su entorno, esquivar fantasmas y planear rutas más efectivas se podría implementar agentes reactivos o redes neuronales para el aprendizaje automático y la resolución de problemas. Asimismo, podría optarse por Imitation Learning para que el agente aprenda por demostración con jugadas ya existentes.

Integrar la simulación con fantasmas dentro del AG representa un primer paso crítico para reemplazar secuencias estáticas por agentes más adaptativos y “sensibles” al entorno, acercándonos a un comportamiento de Pac-Man verdaderamente inteligente. En referencia a este objetivo futuro se plantea un pseudocodigo.

```
Eval(cromosoma):
  inicializar estados
  para acción en cromosoma:
    actualizar fantasmas (perseguir/huir)
    mover PacMan
    comer comida/píldora
    si colisión:
       si weak → comer fantasma
       else → muerte
  retornar fitness
```

Figura 8. Pseudocódigo de simular fantasmas en AG

VI. Referencias

[1] C. Doty‐Humphrey, "Small Fast Random Number Generators," 2017.
Available:
https://github.com/bryc/code/blob/master/jshash/PRNGs.md

VII. Apéndice A - Datos experimentales y hojas de cálculo

Los resultados completos de las corridas del algoritmo genético, junto con las tablas de estadísticas (media, desviación estándar y curvas de evolución del fitness), se encuentran disponibles en el archivo Excel:

Disponible en:
https://github.com/cthompson23/P2_PacmanGame/tree/main/documents