Proyecto de Simulación e IA

Integrantes:
Hivan Cañizares Diaz
Franco Hernández Piloto
Carlos Mauricio Reyes Escudero

Universidad de La Habana
Curso 2024

Resumen

Este proyecto se centra en la simulación de combates Pokémon utilizando inteligencia artificial (IA) para evolucionar Pokémon y crear agentes de entrenadores. La primera fase utiliza algoritmos genéticos para evolucionar a los Pokémon a través de múltiples generaciones, seleccionando y reproduciendo los más exitosos. En la segunda fase, se desarrollan agentes de entrenadores basados en dos enfoques: el modelo BDI (Creencias, Deseos, Intenciones) y Modelos de Lenguaje de Gran Tamaño (LLMs). Estos agentes son evaluados en un entorno simulado para analizar su rendimiento en combate.

1. Introducción

Este proyecto tiene como objetivo simular combates Pokémon, utilizando técnicas de inteligencia artificial para analizar el rendimiento de diferentes estrategias y agentes en este entorno.

Para ello, desarrollamos dos partes:

1.1. Parte 1: Evolución mediante Algoritmos Genéticos

Utilizamos algoritmos genéticos para evolucionar a los diferentes Pokémon. Este proceso genera múltiples generaciones de Pokémon, seleccionando y reproduciendo los más exitosos. A través de este enfoque, se espera que los Pokémon evolucionados presenten características más fuertes y efectivas en combate.

1.2. Parte 2: Agentes de Entrenadores

En la segunda fase del proyecto, se desarrollan agentes de entrenadores utilizando dos enfoques distintos: el modelo BDI (Creencias, Deseos, Intenciones) y Modelos de Lenguaje de Gran Tamaño (LLMs).

1.2.1. Agentes BDI

Los agentes BDI toman decisiones basadas en un conjunto de creencias sobre el estado del combate, deseos que reflejan sus objetivos y intenciones que guían sus acciones. Estos agentes utilizan un motor de inferencia que les permite razonar sobre las acciones disponibles y seleccionar la mejor estrategia en función del contexto actual.

1.2.2. Agentes LLM

Por otro lado, los agentes LLM utilizan modelos de lenguaje generativo para tomar decisiones durante el combate. Estos agentes son capaces de interpretar el estado del juego y generar respuestas coherentes basadas en la información proporcionada. Al interactuar con el entorno, el agente LLM puede formular estrategias complejas y adaptarse a las acciones del oponente.

2. Simulación de Combates

Para la simulación de combates, nos apoyamos en una implementación existente de mecánicas de batalla Pokémon [1], que proporciona una base sólida y precisa para las interacciones entre Pokémon durante el combate.

3. Evolución mediante Algoritmos Genéticos

En esta parte, se lleva a cabo un proceso de evolución en paralelo para los Pokémon y sus cerebros, que son redes neuronales evolucionadas utilizando ES-HyperNEAT (Evolvable-Substrate HyperNEAT). Para la evolución de las redes neuronales, se utiliza la biblioteca PUREPLES [2], que proporciona una implementación eficiente del algoritmo ES-HyperNEAT.

Un aspecto clave de este enfoque es que el fitness (o aptitud) se comparte entre un individuo

Pokémon y su cerebro (red neuronal). La fórmula utilizada para calcular el fitness es:

$$\text{fitness} = k_1 \left(\frac{\text{lvl}}{100} \right) + k_2 \left(\frac{\text{won\_battles}}{\text{total\_battles}} \right) \quad (1)$$

Donde:

- **lvl** es el nivel del Pokémon.
- **won_battles** es el número de batallas ganadas por el Pokémon.
- **total_battles** es el número total de batallas en las que ha participado.
- $k_i \in [0, 1]$, $\sum(k_i) = 1$

Al optimizar el fitness compartido, se asegura que tanto el Pokémon como su cerebro evolucionan de manera coordinada, lo que lleva a la formación de individuos altamente adaptados y capaces de tomar decisiones estratégicas en los combates.

3.1. Evolución de Especies Pokémon

Los Pokémon se representan como entidades que evolucionan a lo largo de generaciones. Este proceso implica:

- **Reproducción**: Se utiliza el entrecruzamiento uniforme (uniform crossover) para combinar los atributos de dos Pokémon progenitores.
- **Mutación**: Los movimientos, habilidades y naturalezas de los Pokémon pueden mutar. La tasa de mutación está controlada por un índice de temperatura que disminuye a medida que aumenta el fitness del Pokémon, siguiendo la fórmula:

$$\text{temperatura} = (1 - \text{fitness})^{\text{COOLER}}$$

donde COOLER es una constante que controla la velocidad de enfriamiento.

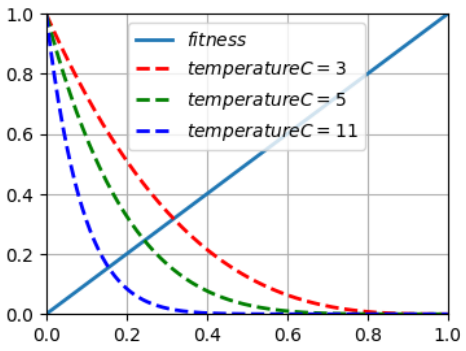

Figura 1: Relación entre el fitness y la temperatura para distintos valores de COOLER.

3.2. Inteligencia Artificial y ES-HyperNEAT

La IA que guía a cada Pokémon se implementa como una red neuronal evolucionada usando ES-HyperNEAT (Evolvable-Substrate HyperNEAT):

- **Entrada**: Estado del terreno y características de los Pokémon en combate.
- **Salida**: Movimiento a efectuar en el turno actual.
- **Evolución**: La topología de la red evoluciona a lo largo de las generaciones, adaptándose para mejorar la toma de decisiones en combate.

3.3. Manejo de Experiencia y Niveles

El sistema de experiencia y niveles se basa en datos actualizados de fuentes especializadas en Pokémon[3] [4] . La fórmula para el cálculo de experiencia es:

$$\Delta EXP = \left(\frac{b \times L}{5} \times \frac{1}{s} \times \left(\frac{2L+10}{L+L_p+10} \right)^{2.5} + 1 \right) \times t \times e \times v \times f \times p$$

Figura 2: Fórmula de cálculo de experiencia en Pokémon

3.4. Analizando monopolio de ataques

Para verificar la hipótesis de que los Pokémon tienden a especializarse en un único ataque poderoso, se implementa el siguiente proceso:

1. Recopilación de datos de uso de ataques para cada Pokémon evolucionado.
2. Cálculo de la distribución acumulativa empírica de uso de ataques.
3. Definición de una distribución teórica de uso equitativo de ataques.
4. Aplicación de la prueba de Kolmogorov-Smirnov de una muestra para comparar las distribuciones empírica y teórica.
5. Análisis de los resultados para determinar si hay una desviación significativa hacia el uso predominante de un solo ataque.

Este enfoque permite cuantificar objetivamente la tendencia de los Pokémon evolucionados a especializarse en estrategias de ataque único o diversificadas.

3.5. Experimentos sobre variabilidad de movimientos

Se realizaron dos experimentos distintos para evaluar la hipótesis principal:

1. **Experimento sin énfasis en la variedad de movimientos:** En este primer experimento, la función de fitness no considera la variedad de movimientos utilizados por el Pokémon, lo que permite que la evolución se centre únicamente en la efectividad de los Pokémon en términos de nivel alcanzado y tasa de victorias.

2. **Experimento con énfasis decreciente en la variedad de movimientos:** En este segundo experimento, se amplia la función de fitness completa descrita anteriormente:

   $fitness_1 = p \cdot \left(\frac{\text{moves\_used}}{4}\right)^{2,9} + (1-p) \cdot fitness$

   donde:

   - $p = \frac{3}{\text{age}+2,7}$ es un factor que disminuye con la edad del Pokémon.
   - moves_used es el número de movimientos diferentes utilizados por el Pokémon.

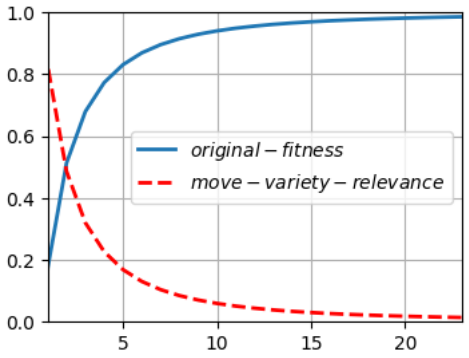

Figura 3: Relevancia de la variedad de movimientos y el fitness usual respecto a la edad.

Esta fórmula está diseñada para:

- Fomentar inicialmente la variedad de ataques usados.
- Reducir gradualmente la importancia de la variedad a medida que el Pokémon envejece.
- Balancear la importancia entre el nivel alcanzado y la tasa de victorias.

Estos dos experimentos nos permitirán comparar cómo la consideración de la variedad de movimientos afecta la evolución de las estrategias de los Pokémon. El primer experimento sirve como línea base, mientras que el segundo nos ayudará a entender si la presión inicial para usar una variedad de movimientos conduce eventualmente a estrategias más especializadas o si mantiene un conjunto más diverso de tácticas.

Los resultados de ambos experimentos se analizarán utilizando la prueba de Kolmogorov-Smirnov descrita anteriormente, lo que nos permitirá cuantificar y comparar la tendencia hacia la especialización en un único ataque poderoso en ambos escenarios.

3.6. Resultados

stat	exp. 1	exp. 2
dks	0.8200	0.8467
p-val	0.0004	0.0002
entropy	0.3251	0.8412
mean	1.10	1.33
variance	0.09	0.69

Cuadro 1: Resultados de los Experimentos de Uso de Movimientos

Los resultados obtenidos de ambos experimentos proporcionan información valiosa sobre el comportamiento de los Pokémon en combate. A continuación, se detalla el significado de cada estadística y su relevancia en el contexto del estudio:

- D de Kolmogorov-Smirnov: Los valores obtenidos (0.8200 para el Experimento 1 y 0.8467 para el Experimento 2) indican una diferencia significativa entre la distribución observada de uso de movimientos y una distribución uniforme. Esto sugiere que los Pokémon no utilizan sus movimientos de manera equitativa en ambos experimentos.

- p-valor: Los p-valores muy bajos (0.0004 y 0.0002) proporcionan evidencia estadística fuerte para rechazar la hipótesis nula de que los movimientos se utilizan de manera uniforme. Esto indica que los Pokémon tienden a especializarse en ciertos movimientos.

- Entropía de Shannon: La entropía de Shannon aumentó de 0.3251 en el Experimento 1 a 0.8412 en el Experimento 2, lo que sugiere una mayor diversidad en el uso de movimientos en el segundo experimento. Sin embargo, el valor de entropía sigue siendo relativamente bajo, indicando que, en general, los Pokémon aún utilizan un número limitado de movimientos.

- Promedio de movimientos usados: El promedio de movimientos utilizados aumentó de 1.10 a 1.33, lo que sugiere que al introducir un énfasis decreciente en la variedad de movimientos, los Pokémon comenzaron a usar un poco más

de movimientos diferentes, aunque todavía se especializan en un solo ataque.

- Varianza de movimientos usados: La varianza también aumentó de 0.09 a 0.69, lo que indica una mayor dispersión en el uso de movimientos. Esto sugiere que algunos Pokémon están utilizando más movimientos que otros, lo que podría ser un resultado de la estrategia de combate más flexible implementada en el segundo experimento.

Estos hallazgos refuerzan la idea de que los Pokémon tienden a especializarse en ciertos movimientos, pero también indican que al permitir una mayor variedad de movimientos, existe la posibilidad de diversificación en sus estrategias de combate.

4. Implementación de Agentes para Entrenadores

En esta parte del proyecto, se implementan agentes para entrenadores que gestionan múltiples Pokémon, permitiendo tanto el uso de ataques como el cambio entre ellos. Estos agentes están diseñados para tomar decisiones estratégicas durante los combates, optimizando su rendimiento en función de las condiciones del entorno.

4.1. Implementación del Agente BDI

El agente BDI se implementa utilizando un motor de inferencia lógica llamado PyReason [6]. Este motor permite razonar sobre las creencias, deseos e intenciones del agente, facilitando la toma de decisiones informadas en tiempo real.

4.1.1. Modelado de Creencias

Las creencias del agente se modelan como reglas dentro de PyReason. Estas reglas permiten al agente evaluar el estado actual del combate y tomar decisiones basadas en su conocimiento. Por ejemplo:

$$des(M,T) \Leftarrow can(M,T), typ_m(M,t_1), typ_p(T,t_2), eff(t_1,t_2) \quad (2)$$

Esto indica que si el agente puede realizar un movimiento sobre un objetivo y el tipo del movimiento es efectivo contra el tipo del objetivo, entonces el agente tiene el deseo de llevar a cabo esa acción.

4.1.2. Modelado de Deseos

Los deseos se interpretan a partir del conjunto de predicados en PyReason que forman las creencias. En la figura se puede observar una arista que se convierte en un deseo, y surge de aplicar una creencia:

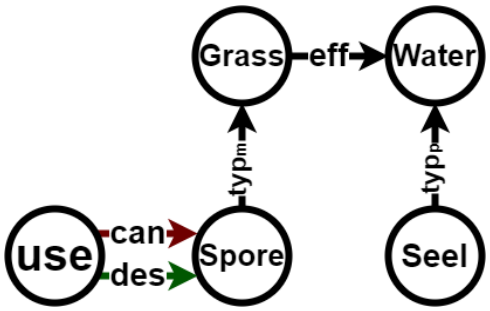

Figura 4: En rojo un hecho agregado al grafo de conocimiento y en verde un deseo inferido usando la regla (2).

4.1.3. Intenciones

Las intenciones no están expresadas explícitamente; en su lugar, se utiliza un índice obtenido a partir de las reglas para determinar cuánto se desea ejecutar una acción. Este índice se calcula utilizando la función softmax, que permite seleccionar acciones en función de la intensidad del deseo. El índice se obtiene declarando diversos predicados para diferentes intensidades de deseo y aprovechando las anotaciones de PyReason.

4.1.4. Acciones y Restricciones

Las acciones se toman a partir de un plan predefinido, y se imponen restricciones en el grafo de conocimiento para evitar acciones inválidas. Esto asegura que el agente tome decisiones coherentes y efectivas durante el combate.

4.1.5. Conocimiento inicial

Se probaron dos alternativas: pasar todo el conocimiento al agente o solo el conocimiento necesario para el combate. Se observó una mejora en la eficiencia de hasta diez veces al optar por la segunda opción, lo que demuestra la importancia de optimizar la información disponible para el agente.

4.1.6. Conocimiento Actual

Cada turno, el agente recibe información detallada sobre todo su equipo y el estado del campo de batalla. Sin embargo, el agente no tiene acceso al estado privado del equipo o del agente rival. Esta limitación en la información disponible es crucial

para la toma de decisiones, ya que obliga al agente a basar sus estrategias únicamente en los datos que puede observar y procesar en tiempo real.

4.2. Implementación del Agente Basado en LLM

Los agentes basados en LLM se desarrollaron utilizando de la API de Google [5], el modelo Gemini Flash 1.5. Este enfoque permite a los agentes tomar decisiones informadas durante los combates Pokémon, aprovechando el procesamiento del lenguaje natural para interpretar el estado del juego y generar respuestas adecuadas.

4.2.1. Instrucción de Sistema

Se crea una instrucción de sistema a través de la cual se le pasa al modelo tanto el conocimiento relevante como las instrucciones de formato para las órdenes. Esta instrucción establece el contexto en el que el agente debe operar y define cómo debe estructurar sus respuestas. Entre las instrucciones mas relevantes estan:

- Razonamiento Paso a Paso: Se le indicó al agente que razonara paso a paso sobre su posible decisión.
- Categorización de Jugadas: Cada posible jugada se categoriza desde la A hasta la F. Este enfoque estructurado permite al agente evaluar diferentes estrategias.

4.2.2. Información del Estado del Campo

Cada turno, se proporciona al agente información sobre el estado del campo y de su equipo. Algunos valores numéricos, como la vida de los Pokémon, se convierten a texto categórico para facilitar la comprensión del estado actual. Por ejemplo, la vida puede ser descrita como 'muy baja', 'baja', 'relativamente alta' o 'perfecta'.

4.2.3. Obtención de Movimientos Válidos

Para extraer el movimiento válido retornado por el modelo, se utiliza un conjunto de expresiones regulares. Estas expresiones permiten identificar rápidamente las acciones que el agente puede ejecutar en función de la respuesta generada por el modelo.

4.2.4. Historial de Turnos

Se le pasa al agente el historial de turnos y sus consecuencias para evitar que cambie drásticamente su estrategia en situaciones de desventaja total. Esto ayuda a mantener una coherencia en la toma de decisiones y evita reacciones impulsivas que podrían resultar en una pérdida inmediata.

4.3. Sobre la flexibilidad de los agentes racionales

El agente BDI no se limita a ser una simple entidad con una capacidad fija para la tarea para la que ha sido diseñado; en realidad, representa un framework completo que permite la construcción de diversos modelos de combate. Esta flexibilidad es fundamental, ya que los modelos pueden desempeñarse de manera variable, adaptándose a diferentes situaciones y contextos en el juego.

4.4. Análisis de Sensibilidad de los Modelos BDI

Se realizó un análisis de sensibilidad en el que se variaron las reglas que los modelos BDI tenían como creencias. Este análisis permitió evaluar cómo estos cambios afectaban el rendimiento de los agentes en combate, considerando la relevancia asignada a diferentes acciones.

El modelo BASELINE da alta relevancia a ataques efectivos y media relevancia a intercambios de Pokémon. En contraste, el modelo LESS_SWITCH reduce la prioridad de las reglas para intercambiar Pokémon, lo que puede afectar su capacidad para adaptarse a situaciones cambiantes. Por otro lado, en el modelo NO_SWITCH, se incluyen reglas para cambiar de Pokémon solo cuando el actual está debilitado, lo que limita las oportunidades de intercambio a situaciones críticas.

A continuación se presentan las comparaciones entre diferentes configuraciones de los modelos.

4.4.1. Resultados del Análisis

Name	Won	Percentage (%)
BASELINE	16	53.33
LESS_SWITCH	14	46.67

Cuadro 2: Resultados de la comparación entre BASELINE y LESS_SWITCH

Name	Won	Percentage (%)
NO_SWITCH	15	50.0
BASELINE	15	50.0

Cuadro 3: Resultados de la comparación entre BASELINE y NO_SWITCH

Name	Won	Percentage (%)
LESS_SWITCH	14	46.67
NO_SWITCH	16	53.33

Cuadro 4: Resultados de la comparación entre LESS_SWITCH y NO_SWITCH

Referencias

[1] **pokemon-python**: Simulador de batallas que proporcionó una base sólida para las mecánicas de combate y las interacciones entre Pokémon (Repositorio de GitHub).

[2] **PUREPLES**: Proporcionó la implementación de ES-HyperNEAT (Evolvable-Substrate HyperNEAT) utilizada para evolucionar las topologías de redes neuronales (Repositorio de GitHub).

[3] **Bulbapedia**, the community-driven Pokémon encyclopedia (ver sitio web).

[4] **Pokémon Database** (ver sitio web).

[5] **Google API for devs** (ver sitio web).

[6] **PyReason** (Repositorio de GitHub).