Proyecto Final de IA y Simulación
El Juego de Ander

Javier Rodríguez Sanchez C-411
Alejandro Camacho Pérez C-412
Luis Alejandro Rodríguez Otero C-411

April 16, 2024

Contents

1 Introducción

1.1 Estrategias para guerras medievales: El juego de Ander

En un contexto medieval, donde hay n reinos en guerra, se busca saber cuál es la mejor estrategia a seguir por los reyes para salir victoriosos. En este contexto, para evitar conflictos históricos o culturales, lo desarrollaremos en una tierra lejana llamada Ander, donde existen n reinos anónimos.

Un reino consta de un rey, el pueblo al que gobierna, su castillo/fortaleza y un ejército. Todos los reinos inician en guerra por conveniencia. Un reino puede mejorar sus murallas o aumentar su ejército y atacar o aliarse a otros reinos. El objetivo de cada uno es salir airoso de la pelea, es decir, ser el único que sobreviva.

2 Modelo utilizado

2.1 Resumen del Modelo

El modelo usado para representar el sistema consiste en un accionar por turnos de cada reino. La población estará clasificada según que tan grande es (dado por niveles del 0 al 10) y su castillo/muralla tendrá un índice de fortaleza. El ejército estará dividido en tropas cada una con un tamaño. La población de un reino asumimos que está en constante cambio, y todos los turnos variara en tamaño, con mayor probabilidad de permanecer cerca del valor anterior. Durante un turno los reyes pueden ordenar a su población a reforzar sus murallas o a reclutar tropas, según permita el tamaño del pueblo; y ordenar a sus tropas a atacar otros reinos. Para destruir un reino debes matar a su rey. Para esto primero hay que destruir su ejército, luego sus murallas, luego su pueblo y finalmente al rey. Si dos tropas pelean, la más fuerte gana, pero pierde en fuerza igual al nivel de la tropa destruida. Si una tropa ataca una muralla, la muralla resiste, pero pierde resistencia igual a la mitad del nivel de la tropa que la asedia. Si un pueblo es atacado por una tropa, este quedará diezmado en una cantidad igual al nivel de la tropa. Una simulación termina cuando solo queda un rey en pié o cuando esta alcanza un número de rondas predefinido, en este caso el ganador es el reino con mejor puntaje, definiendo puntaje como la suma entre los tamaños de todas sus tropas y su muralla.

Los reyes consistirán en agentes, los cuales contienen conocimiento básico de como manejar a su reino y el estado actual de este y de los reinos enemigos en todo momento. Ademas, cada rey tiene niveles de relación con el resto de los reinos, los cuales inician siendo bajos, pero dichas relaciones pueden mejorar. Un rey solo conoce la relación que tienen otros con él y las relaciones no son simétricas. Entre las cosas que afectan una relación están: entrar en alianza(aumento grande), que ataquen a un enemigo(aumento mínimo), que ataquen a un aliado(disminución mínima) recibir un ataque a una tropa(disminución estándar), recibir un ataque a la muralla(disminución alta), recibir un ataque al pueblo(disminución enorme), ser traicionado(disminución maxima). Dos reinos pueden entrar en una alianza, la cual queda como un pacto de no agresión por 3 turnos. Una traición ocurre cuando un aliado ataca a otro.

Las decisiones sobre la alianza y que acciones serán tomadas en un turno, conforman la estrategia que sigue un rey. El objetivo de nuestro modelo es

encontrar la mejor estrategia.

2.2 Ventajas

Modelo simple y escalable a todos los niveles. Permite dar gran diversidad de estrategias a probar en un gran número de condiciones iniciales.

2.3 Desventajas y Limitaciones

Las decisiones de los reinos no se realizan en tiempo real, lo cual afecta el realismo del modelo y no toma en cuenta la velocidad de la toma de decisiones, factor crucial en una guerra. La ausencia de mapa y de recursos también hace que no se tomen en cuenta costos de ataque a un reino o fortalecimiento de las murallas. El conocimiento de los estados de los reinos ajenos no es realista, ya que en el sistema el conocimiento de los reyes suele ser limitado. Una tropa numéricamente superior no siempre es la que sale victoriosa de un combate, quitándole lugar a la incertidumbre. El carácter centralizado de los reinos también es una suavización del problema, haciendo que todas las órdenes de este se ejecuten, y no hayan controversias internas.

2.4 Aspectos Mejorables

Los siguientes aspectos no fueron incluidos en el modelo por falta de tiempo, pero son perfectamente integrables:

- La adición de capacidades especiales: cada reino tendría una capacidad extra y distintiva que le permitiera distinguirse del resto de los reinos (se pudiera ver como cierta tecnología que contengan, una especie de arte marcial, un don divino o una raza), que le permita extender las reglas establecidas y otorgarles cierta ventaja.

- La comunicación entre aliados: en una alianza, un rey puede recomendarle a un aliado suyo que acciones realizar en su turno. De esta manera algunas estrategias podrían basar sus jugadas en las recomendaciones de sus aliados

3 Implementación

El código de la aplicación que permite la simulación, y que será relatado a continuación, se encuentra en la carpeta **src**.

3.1 Modelo de la Simulación

Todas las descripciones y la lógica de un reino se encuentran en la clase **Kingdom** en el archivo **Simulation_Model/Reigns.py**. De aquí vale la pena destacar el método 'actions', el cual devuelve una lista con todas las órdenes inmediatas que puede dar un rey, y el método 'act', que dado una acción la ejecuta. La clase está diseñada de tal manera que pueda ser extensible para agregar nuevas funcionalidades al modelo en caso de ser necesario.

3.2 Agente

Los agentes del modelo serían los reyes, y están representados en la clase **Agent** del archivo **Agent/Agent.py**. Estos contienen como propiedad una base de conocimientos simple, representada en **Agent/Knowledge_Base.py**.

3.3 Base de Conocimiento de los agentes

Esta cuenta con dos métodos fundamentales: 'Learn' y 'Think'. El primero permite al agente guardar toda aquella información que sea útil para este, como la estrategia que seguirán, el número de reinos y la información de estos, el estado actual de la geopolítica y las acciones realizadas por otros reinos. El segundo permite a un agente tomar sus decisiones dado su conocimiento, como las acciones que realizara durante su turno, sus consecuencias inmediatas y con que reinos aliarse.

3.4 Toma de decisiones

Las decisiones de un rey consistirán en una búsqueda informada de cuál sería el mejor camino para su reino, donde buscaría entre todos los posibles finales cuáles son los que más le conviene seguir, de esta forma, este elige cuáles serían las decisiones óptimas en función de su estrategia.

3.5 Estrategia de un agente

Un agente tiene una estrategia, y nuestro objetivo es encontrar la más conveniente. Todas estas estrategias se encuentran en la carpeta **Strate-**

gies/Implementations y heredan de la clase **Strategy** en **Strategies/Strategy.py**. Para implementar nuevas estrategias solo deben crearse más clases que hereden de **Strategy** y añadirlas a esta carpeta. Cada una representa una forma distinta de tomar decisiones para un agente en una simulación.

3.6 Simulador

El proceso de simular consta de dos componentes principales:

- Una interfaz que sirve de intermediario entre el usuario y la simulación.
- Un simulador que permite correr varios juegos y obtener estadísticas de estos.

3.7 Interfaz

La interfaz de la simulación está creada para que pueda ser extensible a cualquier tipo de visualización que ofrezca python. Esto se logra con una clase abstracta **SimulationInterface** la cuál contiene tanto los métodos necesarios para poder interactuar con el usuario, cómo los métodos para poder correr una simulación.

En este trabajo, la interfaz está a cargo de la clase **SimulationInterfaceConsole**, la cuál permite correr la simulación en la consola de python. Esta clase se encarga de mostrar los resultados de la simulación, así como de pedir los parámetros necesarios para correr una simulación. Ambas clases se encuentran en **Simulation_gestor.simulation_interface.py**

3.8 Llm

Para un mejor análisis de los resultados, utilizamos **LLM** para resumir el historial del juego y crear una historia más amena y legible en vista a la interpretación de las decisiones tomadas por los agentes, y entender qué decidieron hacer y en qué contexto. Dando la facilidad de tener una idea del desarrollo de la simulación.

Con la finalidad de lograr esto, se crea un proceso que va a estar corriendo en paralelo, el cual es el encargado tanto de ir gestionando los logs que generó un juego, como de comunicarse con el proveedor de LLM para poder ir generando la historia del juego.

Para lograr lo más cercano posible a lo deseado, a la hora de hacer la petición al servidor LLM, como parámetro del sistema se pasa un resumen de lo que ha acontecido de historia hasta el momento, así como cosas que son invariantes en la historia que se está generando, y que de perderse, puede carecer de sentido lo generado. Luego, se pasa como petición del cliente un prompt el cual es creado a partir de la información extraida del log del juego que se desea decorar.

Para establecer la comunicación entre el proceso del simulador y el proceso creador de la historia, ambos procesos comparten una misma cola de logs, que utiliza la clase **Queue**, de la biblioteca **multiprocessing**, ya que esta garantiza seguridad en la comunicación entre procesos. Además, el proceso de la historia recibe del proceso del simulador, un delegado que le va a permitir la gestión del log en la interfaz de la historia.

4 Análisis de las simulaciones realizadas

4.1 Estrategias implementadas

Volviendo a la pregunta inicial: ¿cuál es la mejor estrategia?. Para saber esto, buscamos resolverlo inicialmente para el caso n=2 y n=3 (n es la cantidad de jugadores) en igualdad de condiciones iniciales. Para cada simulación se estableció un límite de 50 rondas y se anotó el ganador de cada una. Las estrategias implementadas actualmente son las siguientes:

- **Allies Strategy:** Toma Decisiones de acuerdo al nivel de relación que tiene con el resto de reinos. De esta forma, tiene una alta probabilidad de atacar a los reinos con los que se lleva mal y propone alianzas a aquellos con los que tiene buenas relaciones.

- **Always Attack Strategy:** Ataca a la mayor cantidad posible de objetivos, entiéndase tropas o murallas enemigas. No propone alianzas y solo acepta proposiciones de esta a reinos con los que tiene buenas relaciones.

- **Attack Weak Strategy:** Ataca al reino con el ejército más débil en ese momento. Aunque por momentos su comportamiento es equivalente al de **Always Attack**, la principal diferencia entre estas dos estrategias está en la manera que elijen aliados. La anterior era bastante básica pero esta solo se alía con el reino del ejército más fuerte del momento. Por lo que, resumiendo, esta es una estrategia que ataca al más débil y se alía con el más fuerte.

- **Current Situation Strategy:** Crea una tabla en donde ubica de forma ordenada a los reinos respecto a su poder (teniendo en cuenta tropas y murallas) y actúa dependiendo de su puesto en dicha tabla. Por ejemplo si es el primero de la tabla, es decir, si es el reino más poderoso de ese momento, tiene un 70% de posibilidad de solo defenderse y un 30% de posibilidad de atacar al que esté en 2do puesto (Este 30% disminuye si el objetivo tiene buenas relaciones con él o son aliados), si por el contrario es el último de la tabla entonces intentará atacar al penúltimo para mejorar su posición, y si se encuentra en medio de la tabla puede atacar al que está por delante o al que está por detrás. Cada vez que elije atacar a alguien primero comprueba su relación y estado de alianza hacia este, lo cual puede hacerlo no atacar y optar por un final defensivo. Respecto a las alianzas, este solo propone alianzas cuando es el reino más poderoso, y solo acepta

proposiciones de reinos que no están en una posición adyacente a la suya en la tabla, ya que estos son usualmente sus objetivos a atacar.

- **Defend Strategy:** Sube de nivel su muralla y su ejército en cada turno y opta por no atacar casi nunca. Tiene un 20% de posibilidades de atacar un reino con el que tenga muy malas relaciones, lo cual es la única situación en que esta estrategia es ofensiva. No propone alianzas y acepta solicitudes solo de reinos con los que tiene buenas relaciones.

- **Do Nothing Strategy:** Pasa cada turno sin hacer ninguna acción, no propone, ni acepta alianzas. Es normal preguntarse qué hace esta estrategia aquí, pero fue bastante útil a la hora de probar las simulaciones y corregir bugs, asi que se quedó e incluso va a formar parte del análisis a continuación.

- **Focus Strategy:** Fija desde el inicio a un reino enemigo y dirige sus ataques a este, turno tras turno hasta que muera. Cuando su objetivo muere procede a buscar un nuevo objetivo, intentando que no sea un aliado, aunque si no queda de otra no tendrá problemas en romper una alianza. Solo crea alianzas con reinos que no son el objetivo al que está atacando en ese momento.

- **Multiple Strategy:** Recibe en su constructor una lista de estrategias y un nivel de importancia por cada una, en cada turno elije la acción que elegiría una de esas estrategias dando más posibilidad a las estrategias más importantes pero sin descartar ninguna. Esta estrategia es mas bien una familia de estrategias. Evidentemente esta estrategia depende de otras para funcionar y hay infinitas formas de meterla en una simulación (jugando con las estrategias y las importancias) pero al final su efectividad va a depender de las estrategias que la compongan. Por esto, y para ahorrar tiempo, elejimos dejar esta estrategia fuera del análisis.

- **Random Strategy:** Como su nombre indica, juega de manera aleatoria. Esto incluye proponer alianzas a reinos aleatorios y aceptar una solicitud de quien sea con un 50% de probabilidad.

4.2 Resultados para n=2

Se realizaron simulaciones haciendo todos los posibles emparejamientos entre pares de estrategias. Cada par se enfrentará 20 veces, 10 veces saliendo uno de primero y las otras 10 se invierte el orden. En este caso es importante

analizar ambos resultados ya que al ser solo 2 reinos, el primero que mueve tiene una ligera ventaja sobre el otro. Estos fueron los resultados de las simulaciones:

Reino 1	**Reino 2**	**Resultado**
Allies	Always Attack	4 - 6
Always Attack	Allies	8 - 2
Allies	Attack Weak	3 - 7
Attack Weak	Allies	6 - 4
Allies	Current Situation	9 - 1
Current Situation	Allies	2 - 8
Allies	Defend	10 - 0
Defend	Allies	3 - 7
Allies	Do Nothing	10 - 0
Do Nothing	Allies	0 - 10
Allies	Focus	6 - 4
Focus	Allies	7 - 3
Allies	Random	8 - 2
Random	Allies	3 - 7
Always Attack	Attack Weak	7 - 3
Attack Weak	Always Attack	7 - 3
Always Attack	Current Situation	7 - 3
Current Situation	Always Attack	5 - 5
Always Attack	Defend	9 - 1
Defend	Always Attack	4 - 6
Always Attack	Do Nothing	10 - 0
Do Nothing	Always Attack	0 - 10
Always Attack	Focus	8 - 2
Focus	Always Attack	7 - 3
Always Attack	Random	10 - 0
Random	Always Attack	1 - 9
Attack Weak	Current Situation	8 - 2
Current Situation	Attack Weak	3 - 7
Attack Weak	Defend	9 - 1
Defend	Attack Weak	3 - 7
Attack Weak	Do Nothing	10 - 0
Do Nothing	Attack Weak	0 - 10
Attack Weak	Focus	5 - 5
Focus	Attack Weak	6 - 4

Reino 1	Reino 2	Resultado
Attack Weak	Random	10 - 0
Random	Attack Weak	2 - 8
Current Situation	Defend	9 - 1
Defend	Current Situation	1 - 9
Current Situation	Do Nothing	10 - 0
Do Nothing	Current Situation	0 - 10
Current Situation	Focus	3 - 7
Focus	Current Situation	5 - 5
Current Situation	Random	8 - 2
Random	Current Situation	5 - 5
Defend	Do Nothing	10 - 0
Do Nothing	Defend	0 - 10
Defend	Focus	2 - 8
Focus	Defend	9 - 1
Defend	Random	5 - 5
Random	Defend	6 - 4
Do Nothing	Focus	0 - 10
Focus	Do Nothing	10 - 0
Do Nothing	Random	0 - 10
Random	Do Nothing	10 - 0
Focus	Random	9 - 1
Random	Focus	3 - 7

Resumen:

En total se realizaron 560 simulaciones. Cada estrategia participa en 140, de ellas, 70 saliendo de 1ro y 70 de 2do.

#	Estrategia	Ganadas como 1ro	Ganadas como 2do	Total
1	Attack Weak	55	46	101
2	Always Attack	59	42	101
3	Focus	53	43	96
4	Allies	50	41	91
5	Current Situation	40	35	75
6	Random	30	20	50
7	Defend	28	18	46
8	Do Nothing	0	0	0

4.3 Resultados para n=3

De forma similar al caso anterior se hicieron simulaciones con cada posible trío de estrategias (20 simulaciones con cada uno) para probar su efectividad en todos los escenarios posibles. La diferencia es que esta vez no consideramos todos los órdenes posibles de salida principalmente para no tener un número excesivo de simulaciones, para esto hicimos que antes de cada simulación se eligiera el orden de los turnos de manera aleatoria, es decir, en las 20 simulaciones de un trio de estrategias estas tienen un orden aleatorio, lo que nos permite hacer un análisis más general. Estos fueron los resultados:

Reino 1	**Reino 2**	**Reino 3**	**Resultado**
Allies	Always Attack	Attack Weak	7 - 6 - 7
Allies	Always Attack	Current Situation	10 - 5 - 5
Allies	Always Attack	Defend	8 - 7 - 5
Allies	Always Attack	Do Nothing	6 - 14 - 0
Allies	Always Attack	Focus	6 - 7 - 7
Allies	Always Attack	Random	6 - 11 - 3
Allies	Attack Weak	Current Situation	11 - 6 - 3
Allies	Attack Weak	Defend	6 - 9 - 5
Allies	Attack Weak	Do Nothing	8 - 12 - 0
Allies	Attack Weak	Focus	5 - 11 - 4
Allies	Attack Weak	Random	7 - 8 - 5
Allies	Current Situation	Defend	7 - 6 - 7
Allies	Current Situation	Do Nothing	13 - 7 - 0
Allies	Current Situation	Focus	5 - 3 - 12
Allies	Current Situation	Random	11 - 6 - 3
Allies	Defend	Do Nothing	12 - 8 - 0
Allies	Defend	Focus	6 - 6 - 8
Allies	Defend	Random	7 - 11 - 2
Allies	Do Nothing	Focus	10 - 0 - 10
Allies	Do Nothing	Random	16 - 0 - 4
Allies	Focus	Random	6 - 9 - 5
Always Attack	Attack Weak	Current Situation	9 - 8 - 3
Always Attack	Attack Weak	Defend	9 - 8 - 3
Always Attack	Attack Weak	Do Nothing	15 - 5 - 0
Always Attack	Attack Weak	Focus	8 - 6 - 6
Always Attack	Attack Weak	Random	7 - 10 - 3
Always Attack	Current Situation	Defend	9 - 7 - 4

Reino 1	**Reino 2**	**Reino 3**	**Resultado**
Always Attack	Current Situation	Do Nothing	14 - 6 - 0
Always Attack	Current Situation	Focus	9 - 4 - 7
Always Attack	Current Situation	Random	13 - 7 - 0
Always Attack	Defend	Do Nothing	13 - 7 - 0
Always Attack	Defend	Focus	5 - 8 - 7
Always Attack	Defend	Random	11 - 7 - 2
Always Attack	Do Nothing	Focus	9 - 0 - 11
Always Attack	Do Nothing	Random	13 - 0 - 7
Always Attack	Focus	Random	10 - 6 - 4
Attack Weak	Current Situation	Defend	10 - 7 - 3
Attack Weak	Current Situation	Do Nothing	17 - 3 - 0
Attack Weak	Current Situation	Focus	7 - 6 - 7
Attack Weak	Current Situation	Random	11 - 4 - 5
Attack Weak	Defend	Do Nothing	12 - 8 - 0
Attack Weak	Defend	Focus	10 - 4 - 6
Attack Weak	Defend	Random	15 - 1 - 4
Attack Weak	Do Nothing	Focus	7 - 0 - 13
Attack Weak	Do Nothing	Random	16 - 0 - 4
Attack Weak	Focus	Random	13 - 5 - 2
Current Situation	Defend	Do Nothing	12 - 8 - 0
Current Situation	Defend	Focus	6 - 5 - 9
Current Situation	Defend	Random	5 - 11 - 4
Current Situation	Do Nothing	Focus	7 - 0 - 13
Current Situation	Do Nothing	Random	12 - 0 - 8
Current Situation	Focus	Random	7 - 11 - 2
Defend	Do Nothing	Focus	6 - 0 - 14
Defend	Do Nothing	Random	12 - 0 - 8
Defend	Focus	Random	8 - 9 - 3
Do Nothing	Focus	Random	0 - 17 - 3

Resumen:

Se realizaron en total 1120 simulaciones, cada estrategia participa en 420.

#	**Estrategia**	**Victorias**
1	Attack Weak	208
2	Always Attack	204
3	Focus	191
4	Allies	173
5	Defend	137
6	Current Situation	126
7	Random	81
8	Do Nothing	0

4.4 Análisis de los resultados

A simple vista se aprecia que no hacer nada es una terrible estrategia, y la estrategia random no es para nada viable. Luego podemos ver que las estrategias más ofensivas tienen un mejor rendimiento para un número bajo de reinos, y tomar iniciativa primero suele dar una ligera ventaja. Los reinos mas pasivos y las estrategias más elaboradas suelen tener mejor rendimiento a medida que el número de reinos aumentan. Otro dato curioso es que la mayoria de victorias de las estrategias centradas en el aguante (dígase Current Situation y Defend) fueron por score, o sea, por límite de rondas, lo que indica que el límite de rondas que se defina puede afectar el porcentaje de victorias de estas 2 estrategias.

5 Conclusiones

Nuestro trabajo consistió en averiguar el rendimiendo de diversas estrategias a seguir en conflictos medievales. Para esto diseñamos un modelo simple y escalable con el propósito de simular gran variedad de escenarios para poner en práctica diferentes acciones a escoger por sus respectivos reyes, representados como agentes que emplean conocimiento en la búsqueda del mejor final de turno de acuerdo a su propia estrategia. Para el mejor estudio de los resultados de las simulaciones, se utilizo LLM en el procesamiento de los logs de la simulación para hacerlos mas legibles mediante una historia creada a partir de estos. Luego, como resultados de las simulaciones hechas concluimos que es más recomendable ser más agresivo cuando el número de reinos es bajo, una actitud más prudente parecida a la de estrategia Allies si el número es elevado, o una actitud de más aguante si el límite de rondas es bajo. Aun así, el equipo considera que se deben probar mayor número de escenarios con diferentes estrategias a las probadas.