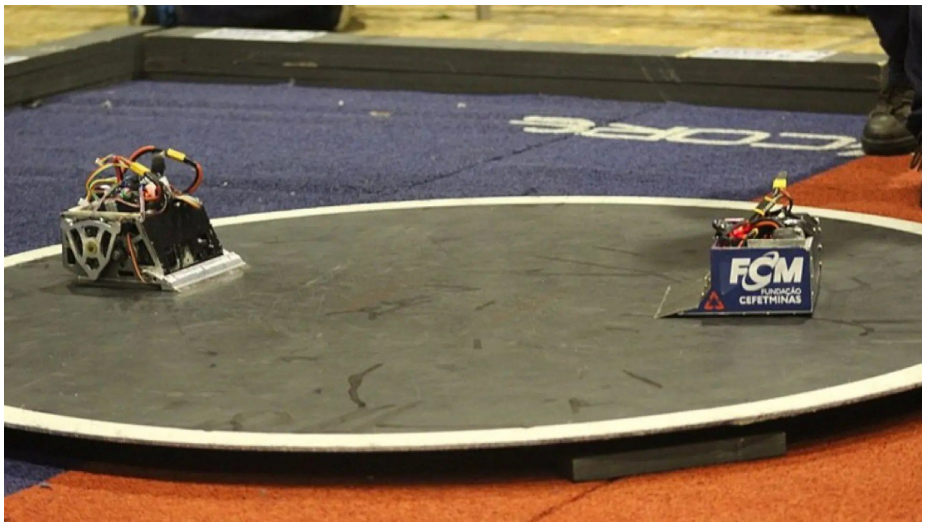
Informe Proyecto IA 2024

Ernesto Bárcena, Alejandro García, José A. Concepción

Septiembre 2024



1 Introducción

El "Robot Sumo" es un deporte real, que incluso se juega a nivel profesional. Análogo al Sumo entre humanos, este consiste en sacar al oponente del campo, solamente empujándolo. A veces se juega con robots autónomos, y a veces a control remoto. En este trabajo se presenta una simulación de lo que se podrıa considerar una generalización de este deporte en su modalidad autónoma. En particular: en lugar de haber solo dos robots, hay varios equipos de robots que se enfrentan todos contra todos, y la forma del campo puede variar durante la partida, lo que generalmente se traduce en que a partir de cierto momento, el campo comienza a reducirse, hasta que desaparece. En el presente informe se aborda principalmente como se atacaron cuatro cuestiones relacionadas con el comportamiento inteligente de los robots. Dichos problemas son: Búsqueda, Representación del Conocimiento, Toma de Decisiones y Procesamiento del Lenguaje

Natural. Para cada uno se explica como se introdujo el problema en el escenario que nos ocupa, así como las soluciones propuestas para cada uno, detalles de implementación y conclusiones observadas.

1.1 Detalles de la simulación

Por simplicidad, la simulación se realiza en 2D. Para simular la física (colisiones, fuerzas, etc.) se utiliza el motor de físicas Box2D. Para la visualización se usa la biblioteca Raylib. Toda acción que realiza un robot (atacar, posicionarse, rotar, etc.) se simula siempre aplicando fuerzas físicas sobre el mismo. Los robots se representan como conjuntos de polígonos, y pueden tener forma arbitraria. Estos poseen además otras propiedades físicas, entre ellas: masa, restitución, fuerza (separada en fuerza linear y rotacional), precisión, rango de visión y frecuencia de procesado (cuantas veces por segundo analiza la situación actual para decidir qué hacer). Hay distintas "personalidades" de robot disponibles, cada una con sus propias preferencias y prioridades a la hora de jugar. Los robots pueden moverse libremente por el campo, pero para hallar su camino de un lugar a otro (esquivando obstáculos) utiliza una discretización (idealmente fina) del campo, en forma de grafo. Respecto al campo, los parámetros más importantes con los que se puede experimentar son el radio, a los cuántos segundos comienza a a reducirse, y a qué velocidad se reduce. El código de los jugadores, su cerebro, y la lógica del campo se pueden encontrar respectivamente en fighter.py, brain.py y scene.py.

2 Problemas abordados

2.1 Búsqueda

El problema de búsqueda es uno de los problemas más importantes en computación en general, y en particular dentro del campo de la Inteligencia Artificial. En nuestro trabajo se planteó un problema de búsqueda complejo y se propuso para resolverlo la implementación de un algoritmo evolutivo. Tanto el problema como la propuesta de solución se explicarán a continuación.

2.1.1 El problema

El problema definido es el siguiente: Dados dos equipos de robots, se quiere encontrar un equipo que sea capaz de obtener el mejor desempeño posible contra ambos de manera simultánea. Por que algoritmos evolutivos?: Si observamos la situación podemos identificar varios detalles importantes para responder a la interrogante anterior. El primero es la naturaleza de los robots. Las capacidades de los robots están determinadas por un conjunto de atributos (masa, fuerza, velocidad, etc.) y los equipos están formados por robots.
Esta 'fragmentación' de las posibles soluciones al problema en componentes más pequeñas es un buen punto de partida para considerar el empleo de algoritmos evolutivos. Más aún, no es descabellado asumir que un 'buen equipo' de robots está compuesto por 'buenos robots', mientras estos a su vez probablemente estén compuestos por 'buenos valores de sus atributos'. Esa misma idea es la que siguen los algoritmos evolutivos: 'Una solución a este problema es buena porque tiene fragmentos que son buenos'.
Aunque ya haya algún motivo para considerar este enfoque, todavía no queda justificado; al fin y al cabo, por qué utilizar una meta-heurística sin estar convencido de que no podemos sencillamente buscar la respuesta de manera exacta? El caso es que en el escenario que nos ocupa no podemos principalmente por un motivo; y es que a pesar de que el espacio de búsqueda no es nada pequeño, resulta mucho más problemático determinar 'que tan bien le fue a un equipo' dado que para ello es necesario simular el combate y el tiempo que tarda cada simulación está en el orden de los segundos.
Además, si tomamos en cuenta el hecho de que las simulaciones poseen muchos elementos estocásticos llegamos a la conclusión de que probablemente necesitemos varias de ellas para lograr aproximar q tan bueno es el desempeño del equipo con respecto a los otros dos; por lo cual un acercamiento ingenuo de fuerza bruta no es factible.

2.1.2 El algoritmo

A continuación se detallará cual es el flujo de ejecución del algoritmo, explicando cada etapa del mismo. El algoritmo recibe como parámetros la cantidad de integrantes del equipo, la cantidad de individuos de cada generación, el número de generaciones, la probabilidad de mutación, la cantidad de simulaciones a realizar

por individuo para determinar el valor de la función de ajuste, las posiciones iniciales de cada uno de los robots, los colores de cada equipo, un vector con los costos de cada unidad de los atributos, una cota superior para la suma ponderada de los valores de los atributos con dichos costos, un umbral de victorias a partir del cual se considera que el equipo es suficientemente bueno y los valores de los atributos de cada robot 'enemigo'. Por otra parte, devuelve una tupla con: una matriz representando los valores de los atributos del equipo resultante, el promedio del valor de ajuste de dicho equipo en su mejor actuación y el porcentaje de victorias que tuvo en dicha actuación.

1-La primera fase del algoritmo consiste en generar de manera aleatoria la población inicial de equipos, garantizando que se cumplan la restricción de que la suma ponderada sea menor que la cota. Además, los atributos deben tomar valores en unos intervalos definidos (esto está relacionado con el motor de físicas empleado para realizar las simulaciones), cosa que también se garantiza.
2-Luego, mientras no hayamos terminado la última generación, se evalúa el desempeño de cada individuo de la generación actual. Los equipos son luego ordenados de manera descendiente de acuerdo a esta evaluación. Los valores obtenidos se convierten en probabilidades (dividiéndolos entre la suma de todos los valores) para obtener un valor entre cero y uno. El uso de esas probabilidades se describe en el paso 4.
3-En este punto primero se verifica si el equipo con mejor desempeño ganó más del porcentaje definido por el umbral recibido de los combates que disputo. De ser así, se devuelve ese equipo, pues se considera que es suficientemente superior. En caso contrario se procede a efectuar el entrecruzamiento. Un detalle importante aquí es que se optó por emplear un criterio de 'elitismo' de modo tal que el equipo con mejor desempeño pasa directamente a la siguiente generación (favoreciendo así la explotación) y además un criterio de 'segregación' mediante el cual todos aquellos equipos que hayan ganado menos del 10% de sus combates son reemplazados por nuevos individuos aleatorios (favoreciendo así la exploración).
4-Una vez entrados en el entrecruzamiento se procede de la siguiente manera: Se selecciona de manera aleatoria a dos individuos de la población actual utilizando como distribución las probabilidades calculadas anteriormente y se escoge de forma aleatoria un punto k entre cero y la cantidad de miembros de un equipo. El nuevo equipo queda determinado por los k primeros robots del primer equipo ancestro y los restantes del segundo. Este proceso se repite hasta obtener todos los individuos de la siguiente generación.
5-El paso restante es la mutación, mediante la cual a cada individuo de la nueva generación se le aplica (con una probabilidad de mutación recibida como parámetro) una alteración en algún atributo de alguno de los robots que lo integran. Esto se hace siempre velando porque se mantengan las restricciones pertinentes. Luego de eso se retorna al paso numero 2.
A continuación se muestra una gráfica que describe en promedio el porcentaje de victorias de los equipos con respecto a la generación a la que pertenecen. El algoritmo se probó con cinco escenarios distintos y un total de 500 equipos de

entre los cuales existen 82 equipos con más de un 60% de triunfos, 42 con más de un 70%, 15 con más de un 80% y 4 con más de un 90%.

A pesar de que en promedio solo apreciamos un valor superior al 40%, solo se exploró utilizando 10 generaciones y se aprecia la convergencia del algoritmo hacia valores óptimos. Incluso un 8.4% de los equipos generados mostraron porcentajes superiores al 70%, lo cual no esta nada mal dado la poca profundidad utilizada.

2.2 Representación del conocimiento

En qué consiste el problema de representación del conocimiento? El problema de representación del conocimiento consiste en encontrar la manera más adecuada de estructurar y formalizar el conocimiento para que pueda ser procesado por sistemas de inteligencia artificial. La representación del conocimiento es crucial para permitir que los robots tomen decisiones informadas en función de su entorno y de las acciones de otros robots. En particular permite a los robots hacerse de un estado de creencias acerca de que están haciendo los robots a su alrededor realizando inferencia sobre el conocimiento representado.

2.2.1 Escenario del Proyecto

El escenario del proyecto consiste en un ring donde varios robots interactúan entre sí con el objetivo de expulsarse unos a otros. Cada robot cuenta con atributos que influyen en su comportamiento, tales como: fuerza, masa, velocidad, entre otros. Además, cada robot puede realizar diversas acciones, que incluyen: atacar, huir, moverse, flanquear, etc.

2.2.2 Problema de Representación del Conocimiento

El desafío principal radicaba en cómo representar el conocimiento experto para inferir las acciones que podrían realizar los otros robots desde la perspectiva de un robot en particular. La complejidad de las interacciones y la necesidad de anticipar las decisiones de los oponentes requerían un modelo que pudiera manejar la incertidumbre y la variabilidad en el comportamiento de los robots. Más aún, era necesario tomar en consideración el transcurso del tiempo ya que nuestro sistema es dinámico.

2.2.3 Solución Implementada: Modelo Oculto de Markov

Para abordar este desafío, se implementó un Modelo Oculto de Markov (HMM), que en nuestro caso fue de primer orden. Surge entonces la interrogante, que es un HMM?
Un Modelo Oculto de Markov es un caso particular de un modelo estadístico llamado Red Bayesiana Dinámica que representa un conjunto de variables aleatorias y sus dependencias condicionales mediante un DAG. En el caso de un HMM se asume que se está representando un proceso de Markov (La ocurrencia de un evento X_t depende únicamente de los k eventos anteriores), formalmente que:

$$P(X_t|X_0, X_1, \ldots, X_{t-k}, \ldots, X_{t-1}) = P(X_t|X_{t-k}, \ldots, X_{t-1})$$

donde k es el orden del modelo El modelo está compuesto por un conjunto de estados no observables y un conjunto de evidencias observables. Además se especifican las probabilidades incondicionales de cada uno de los estados, las probabilidades condicionales entre los estados (modelo de transición) y las probabilidades condicionales de las evidencias dados los estados (modelo sensor). Con estos elementos y las premisas de que se cumple la condición de Markov y de que el motivo que causa las transiciones es homogéneo en el tiempo queda completamente definido el modelo.

Una vez se tiene el modelo, sobre él se pueden realizar fundamentalmente 5 tareas de inferencia, las cuales son:
1: Filtrado: Estimar el estado oculto actual dado el modelo y una secuencia de observaciones hasta el momento actual. Esto se utiliza para monitorear el estado de un proceso en tiempo real.
2: Predicción: Predecir la siguiente observación o estado oculto más probable dado el modelo y una secuencia de observaciones.
3: Suavizado: Calcular la distribución posterior de los estados ocultos pasados, dado toda la evidencia observada hasta el momento.
4: Decodificación: Encontrar la secuencia más probable de estados ocultos que generaron una secuencia de observaciones. Esto se conoce como el problema de decodificación y se resuelve utilizando el algoritmo de Viterbi.
5: Evaluación: Calcular la probabilidad de una secuencia de observaciones dada un conjunto de parámetros del modelo.

En nuestro proyecto se implementaron las dos primeras tareas para ayudar a enriquecer la toma de decisiones. Cabe destacar que además existen métodos para entrenar el HMM utilizando datos. Esto es, dado el conocimiento experto en forma de datos relacionados con los estados y las observaciones ajustar las probabilidades condicionales del modelo. Para esto se puede utilizar el algoritmo de Baum-Welch (esto hubiera sido muy útil en nuestro proyecto, pero desafortunadamente al no poseer dichos datos las probabilidades fueron estimadas de manera empírica).

2.2.4 Resultados y Conclusiones

El uso del Modelo Oculto de Markov permitió a los robots inferir de manera efectiva las acciones de otros robots en el ring, mejorando su capacidad para tomar decisiones tácticas. Este enfoque no solo facilitó la representación del conocimiento experto, sino que también proporcionó un marco robusto para manejar la incertidumbre inherente a las interacciones en un entorno dinámico.

3 Toma de decisiones

Una vez un jugador infiere los estados del resto (atacándolo, huyendo, etc.), queda tomar decisiones al respecto. Esto se hace de forma heurística. Dentro de nuestra implementación, se puede escoger entre 3 funciones distintas para esto, que son como las "personalidades" de los agentes, ya que cada una de ellas prioriza un aspecto distinto a la hora de jugar. Las opciones son:
-IA estándar: intenta llegar a un balance entre ataque y defensa. Se arriesga, pero no mucho. Su ataque es el estándar (cargar directamente contra el enemigo, rotando).
-IA precavida/paranoica: prioriza la defensa. Trata de mantenerse muy alejada del borde del campo. Si hay muchos jugadores con una probabilidad relativamente alta de estarla atacando, huye, o empieza a rotar descontroladamente para alejarlos. Su ataque es el estándar.
-IA planificadora: Prefiere un ataque más complejo, de dos fases: se aleja para tomar impulso, y luego carga contra aquel al que decidió atacar. Si en la segunda fase pasa mucho tiempo, puede decidir dejar ese plan y pasar a otro. También cuenta con el ataque básico, pero lo elige menos.

Hay dos comportamientos que sí tienen en común las 3 opciones: si están muy cerca del borde del campo (que significa "muy cerca" lo define cada una), priorizar alejarse, y si no hay nadie en su radio de visión, explorar.

4 Procesamiento del lenguaje natural

Se implementó un "comentarista deportivo". Es decir, durante la partida se guardan los eventos clave y una vez acabada dicha partida (y después de un poco de post-procesamiento) se les da esta lista de eventos a un LLM, que con el prompt adecuado logra hacer un resumen de la pelea, emulando el estilo de un comentarista deportivo.

En particular, se registran las "muertes", colisiones entre robots, el momento en el que comienza a reducirse el tamaño del campo de batalla, y el final de la partida; para todos estos eventos se guarda en qué momento ocurrieron (medido en segundos desde que empezó la partida), y otros datos (cuál fue el equipo ganador, a qué velocidad ocurrieron las colisiones, etc.). Estos se convierten a una lista en texto plano para poder pasársela a un LLM. En concreto, se usa gemini 1.5 flash. Se usa el valor de temperatura por defecto, ya que así, para esta tarea, la IA respeta los hechos que están explícitos en la lista de eventos (es decir, no alucina), a la vez que le añade color a la narración. El prompt específico que se usa se puede encontrar en el archivo nlp.py.

Ahora, algunas observaciones interesantes que salieron de los experimentos hechos cuando se desarrollaba este sistema:
-Incluso con solo estos 4 eventos básicos, la IA logra hacer inferencia de eventos más complejos, o abstractos, que suceden en el campo. Por ejemplo, si hay muchas colisiones entre los mismos dos jugadores en un espacio reducido de tiempo, la IA lo narra como que están "enredados" o "ensañados". O si dos jugadores colisionan y uno de ellos pierde unas fracciones de segundo más tarde, a veces la IA se atreve a deducir que uno de ellos sacó al otro del campo, a pesar de no habérsele dicho explícitamente que lo intente.
-A veces, si hay muchas colisiones entre muchos jugadores en un período corto de tiempo, la IA se atreve a decir que los robots se están peleando por el centro del campo. Que escoja decir "el centro" en específico puede significar que reconoce su importancia estratégica, o que deduce que ese caos de colisiones sin muertes intermedias debe estar ocurriendo lejos del borde.
-En el prompt se le dice a la IA que se le dará por cada colisión no solamente qué jugadores que están colisionando, sino también a qué equipo pertenece cada uno. Se le dice explícitamente que no le de demasiada relevancia a este último dato, para evitar repititividad en el comentario, y obedece. Sin embargo, si al "no lo enfatices" se le añade algo como "a no ser que sea relevante", le da mucha más relevancia de la deseada.
-No le damos nombre explícito a este "deporte" que simulamos, pero la IA es extrañamente consistente llamándole "RoboRumble". Investigando un poco, resulta que hay varios videojuegos de batallas entre robots que tienen ese nombre (aunque todos con reglas muy distintas al presente), y al parecer por eso la IA le asigna una probabilidad alta a dicho nombre.