Sociedades Artificiales y la Ley del Más Fuerte: Insights desde SugarScapes con Inteligencia Artificial

Brian Ameht Inclán Quesada C-411
Davier Sánchez Bello C-412
Eric López Tornas C-411

May 2, 2024

Índice

Introducción

- Enfoque en la violencia y formación de agrupaciones sociales utilizando un modelo basado en inteligencia artificial.
- Objetivos: explorar influencias de variables como el rango de visión, movimiento y herencia en la supervivencia de agentes.

SugarScapes

- Es un modelo de simulacion clásico expuesto extensamente en el libro "Growing Artificial Societies", publicado en 1996 bajo la autoría de Joshua Epstein y Robert Axtell.
- A través de una simulación bastante simple, con agentes de un comportamiento muy sencillo y pocos rasgos a modelar, logra simular con resultados muy realistas los comportamientos de las sociedades humanas en cuanto a distribución de la riqueza, transmisión cultural, reproducción, combates, entre otros.

Yet another SugarScapes Adaptation

Nos hemos propuesto explorar el tema de la violencia y la formación de agrupaciones (bandas de gángster y demás) en la sociedad, a través de nuestra propia versión de SugarScapes.
Queremos estudiar cuánto influyen el rango de visión, de movimiento, la riqueza heredada por un agente, su nivel de consumo, así como la zona en que nacen en su supervivencia en un ambiente donde la repartición de los recursos naturales que aparecen distribuidos de manera desigual está signada por la ley del más fuerte, por la violencia.
Los resultados por ende, aspiramos a que reflejen la manera en que se adapatan los individuos con diferentes particularidades, digase inteligencia, contactos, fuerza, provenientes de diferentes origenes a situaciones similares, pero que se expresan en la vida real y surgen en medio de climas de caos, donde la violencia impone el orden y la supervivencia.

Rasgos comunes con SugarScapes

Emparentado a SugarScapes, nuestro modelo consiste en una grilla bidimensional, donde conviven diferentes tipos de agentes. Cada agente ocupa una casilla de la grilla, y cada casilla puede soportar una cantidad máxima de azúcar, que varía para cada una. Todos los turnos las casillas que no han alcanzado su máximo potencial de azúcar, incrementan en 1, aunque este parámetro es ajustable, la cantidad de azúcar que contienen. Cada vez que los agentes visitan una casilla, recogen todo el azúcar presente en esta, e incrementan sus reservas en la cantidad de azúcar recogida. Deben, todos los turnos, satisfacer su necesidad diaria de azúcar, consumiendo sus propias reservas. Si en algún momento sus reservas caen de 0, mueren.
Los agentes tendrán diferentes rangos y formas de movimientos, de visión y de ataques, así como nacerán con diferentes necesidades de consumo y reservas.

Particularidades

Los agentes podrán interactuar entre sí atacándose o haciendo asociaciones. Una asociación es un pacto entre dos o más agentes donde cada uno acuerda ceder un determinado porcentaje de sus ganancias a la asociación y luego a cada cual le corresponde una porción de las ganancias totales de la asociación. Podrían además implementarse otras formas de relaciones sociales como las extorsiones.
Los agentes tendran diferentes tipos de comportamiento, a diferencia de la mayoria de los modelos de SugarScapes donde el comportamiento de los agentes es homogeneo.
Los agentes usaran IA para su toma de decisiones en la simulacion de su tipo de comportamiento, a diferencia de la generalidad de los modelos de SugarScapes donde los agentes se comportan de una manera esquematica, regida por reglas muy muy simples, sin control propio de sus decisiones.

Ventajas

- Incluso tras introducir IA a los agentes y complejizar sus relaciones, la simulación sigue siendo lo sufici entemente simple como para incorporar un volumen considerable de agentes, de diferentes tipos, y correr un volumen suficiente de simulaciones como para llegar a conclusiones.
- Estando basado en la experiencia previa del SugarScapes, sabemos y podemos comprobar que emula muy bien los fenómenos ocurridos en la tan compleja sociedad humana, partiendo de una simulación muy sencilla.

Desventajas

- No toma en cuenta la compatibilidad de carácter entre los agentes, o sea, todos los agentes pueden tener el mismo tipo de relaciones con todos los demás agentes.
- No toma en cuenta relaciones familiares entre los agentes

Posibles Mejoras

- En "Growing Artificial Societies" modelan las características culturales de los individuos como cadenas de bits donde ciertas propiedades de la cadena, ya fuera la posición de los bits encendidos y apagados o la cantidad total de tales bits, sirven para agrupar a los agentes en diferentes categorizaciones, y sirven como su diferenciador cultural. Muy bien podríamos emplear esta variante para simular el carácter de los agentes y determinar cuáles tienen relaciones más fluidas con otros, lo cual pudiera ser un factor para retrasar sus comunicaciones, o que influya en sus tomas de decisiones.
- Introducir relaciones familiares entre los agentes, de manera que las relaciones familiares se convirtieran en un medio de poder, tal y como sucede en las agrupaciones mafiosas y las bandas violentas que intentamos emular.

Modelo de la Simulación

La simulación fue implementada de la manera más escalable posible. Por tal motivo, tenemos una interfaz ISimulation, que abstrae los métodos relativos a una simulación que pueden ser modificados de una simulación a otra, digase la forma de resolver los ataques o las propuestas de asociaciones

Step

La simulación es episódica, y simulamos cada episodio usando el método step. Cada episodio es una ejecución de step. Cada step ejecuta secuencialmente los siguientes metodos:

- actualize_agents_vision
- get_association_proposals
- get_attacks
- execute_association_proposals (metodo abstracto)
- execute_attacks (metodo abstracto)
- get_moves
- execute_moves
- feed_agents
- grow (aumenta la cantidad de azucar en cada posicion del tablero)

Los nombres de estos metodos son bastante autodescriptivos, no obstante una explicacion a fondo de toda la implementacion puede encontrarse en el informe Justo al final de cada step ademas se actualiza la condicion de parada

Map

El mapa de una simulación almacena tres tipos de información: la información relativa a la presencia de agentes y objetos en las diferentes posiciones, la información relativa a la cantidad de azúcar en cada posición y la información relativa a las acciones que han sido ejecutadas en cada posición en el turno anterior.
El mapa presenta métodos para insertar o mover objetos o agentes, insertar acciones, eliminar agentes o objetos en una posición específica o con un ID específico, así como averiguar la posición de un agente u objeto con un ID específico, o averiguar el contenido de una casilla en específico. Además, el mapa tiene métodos para aumentar en 1 la cantidad de azúcar que tiene cada casilla no llena y para que cada agente coseche la cantidad de azúcar que le corresponde debido a la casilla en que se encuentra.

Map y ActionInfo

Las acciones son almacenadas en el mapa como una lista por cada posición con las acciones tomadas desde esa posición en el turno anterior. Tales acciones son re presentadas por instancias de la clase Action o de la clase *Action_Info*, o alguno de los tipos que heredan de ellos. Ambas jerarquías de tipo se usan para guardar para su transformación información relativa a los eventos de la simulación, y se usan, al menos en el caso de su presencia dentro de Map, para informar al agente acerca de lo que ha ocurrido a su alrededor.

Agent_Handler

Las simulaciones cuentan con un diccionario que para cada ID de agente le hace corresponder un Agent_Handler.
Agent_Handler es la clase encargada de manejar toda la información relativa a un agente de la simulación, establecer comunicación con él, informándole de todo aquello que puede ver, las acciones que lo afectan y además preguntándole qué acciones desea realizar y verificando si tales acciones son válidas.

IMovement, IVision, IAttack

La capacidad de movimiento de un agente, su rango de visión y de ataque, están dados por instancias de las interfaces IMovement, IVision, IAttack_Range.

IAgent

Los agentes cumplen con la interfaz IAgent que plantea la API a través de la cual los agentes se comunican. Esta API tiene dos clases de métodos: los informativos que pasan al agente datos sobre la simulación, digase aquello que pueden ver, o las acciones que los afectan directamente, y métodos para obtener las acciones que el usuario desea realizar en cada episodio de la simulación.

Agent_with_Memory

Nuestros agentes todos heredan de Agent_with_Memory. Cada instancia de esta clase guarda toda la información que un agente puede necesitar para la toma de sus decisiones. Guarda toda la información proveniente de las visiones del agente en 4 tipos de memorias diferentes: la memoria para visiones de agentes, la memoria geográfica, la memoria para los ataques y la memoria para las asociaciones. Estas memorias están diseñadas de manera que se pueda realizar eficientemente cualquier tipo de query acerca de lo que un agente ha visto, proporcionando escalabilidad, por si es necesario para alguna nueva IA información que anteriormente no habíamos pensado que importara. Para lograr esa eficiencia, las memorias guardan la información que le corresponde a cada una en tablas de SQLite almacenadas en memoria del programa (es decir, una base de datos no persistente). Ademas esta clase guarda info llegada al agente a traves de otros metodos de la API IAgent y no solo de su vision, como es el caso de las acciones de otros agentes que lo afectan.

Association

Las asociaciones cuentan con un id, un conjunto de miembros y un diccionario que almacena para cada miembro su compromiso con la asociacion. Cada miembro de la asociacion tiene un compromiso con ella que regula el porciento de sus ganancias que debe entregar a la asociacion, asi como el porciento de la recaudacion total de la asociacion que le corresponde. La asociacion tiene metodos para repartir las ganancias de cada agente de la forma correspondiente. Las asociaciones tienen por ID al hash de la tupla formada por el ID de sus miembros en orden, de manera que no será posible que el mismo conjunto de agentes tenga más de una asociación que los una

SimpleSimulation

Simple_Simulation es la implementacion que hicimos de ISimulation, y se caracteriza por resolver de manera sencilla, pero a la vez extensible los asuntos relacionados a la manera de ejecutar los ataques y las propuestas de asociacion.
Los ataques cuestan a los agentes que los realizan, mientras más invierten en el ataque más daño causa este en el rival. Dada la naturaleza simple de la simulación, no modelamos el combate de una manera compleja, asumimos que los ataques siempre llegan a su destino y producen el mismo daño no importa la circunstancia. No obstante, en la propia *Simple_Simulation* dejamos la puerta abierta a cualquier tipo de simul ación de ataques de mucho más complejidad que será descrito a continuación.

Graph_of_Attacks

Primeramente, recogemos los ataques que desea realizar cada agente, de manera que los ejecutemos de manera simultánea. Luego, instanciamos un grafo con tales ataques y nos quedamos con sus componentes conexas. Cada componente conexa es un combate, y tiene sentido procesar cada componente conexa, porque, en una situación de combate real, si muchos atacan a la misma vez a un agente, por ejemplo, pueden potenciar el daño de sus ataques al no poder este esquivarlos, o si un agente es atacado, la fuerza de los ataques que realiza en ese instante se ve debilitada. O sea, podríamos, de haber querido, complejizar la simulación de la batalla tanto como hubiéramos querido, dada la expresividad que tiene la modelación de los combates como grafos.

Execute_Associations_Proposals

Para manejar cada propuesta de asociación, preguntamos a cada agente si desea formar parte de la asociación que se propone. En caso de que todos los potenciales miembros acepten, les informamos a cada uno acerca de su nueva afiliación y añadimos al mapa una acción de tipo Association_Creation con los datos de la asociación creada en cada posición de agente unido a la asociación, de manera que aquellos cercanos geográficamente a los miembros de la flamante asociación puedan conocer acerca de la ocurrencia de la asociación y sus detalles.

Miscelaneo

Implementamos unos pocos rangos simples para hacer funcionar a los agentes, digamos, caminar un solo paso en las direcciones principales en SimpleWalking o atacar y observar cualquier objeto en un cuadrado de radio provisto a su alrededor en SquareVision y SquareRange_of_Attack. Pero crear otros tipos de rangos de visión, ataque o movimiento no supone una dificultad, tan solo habría que heredar de las interfaces correspondientes. La implementacion del modelo de la simulacion tiene una enorme cantidad de detalles imposibles de abordar todos en esta breve exposicion. De hecho, en el propio informe fue dificil tratar de abordarlos todos y probablemente hayamos dejado de mencionar un par de cosas. O sea, para una informacion completa acerca de la implementacion del proyecto, chequear el informe.

¿Qué es un Sistema Experto?

- Un sistema experto es una forma de inteligencia artificial que emula la decisión de un experto humano en campos específicos.
- Utiliza conocimientos especializados y reglas inferidas para resolver problemas complejos.
- Consiste en una base de conocimientos y un motor de inferencia que aplica las reglas a la base de datos para deducir conclusiones.
- Proporciona explicaciones y justificaciones para las decisiones tomadas, simulando el razonamiento humano.

Ejemplos de Sistemas Expertos

- **MYCIN:** Desarrollado en los años 70 para diagnosticar enfermedades infecciosas y recomendar tratamientos.
- **XCON:** Utilizado por Digital Equipment Corporation para configurar sistemas informáticos en base a los pedidos de los clientes.
- **CADET:** Asistente para el diseño arquitectónico que ayuda en la creación de planos y estructuras.
- **DENDRAL:** Diseñado para analizar datos científicos y hacer predicciones químicas basándose en espectros de masas.

Agentes

- **Sistema Experto Incorporado:** Todos los agentes en la simulación están equipados con un sistema experto que les permite aprender y adaptarse a partir de su entorno. Este sistema recopila datos continuamente, mejorando las decisiones del agente a medida que avanza la simulación.
- **Base de Conocimiento:** Cada agente posee una base de conocimiento que incluye información sobre su entorno, eventos pasados, y resultados de interacciones previas. Esta base de conocimientos se actualiza constantemente con nueva información recopilada durante la simulación.
- **Motor de Inferencia:** Utilizando los datos de su base de conocimiento, el motor de inferencia de cada agente aplica un conjunto de reglas predefinidas para evaluar situaciones y tomar decisiones. Estas reglas son específicas para cada tipo de agente y definen su comportamiento y estrategias únicas.

Agente Random - Comportamiento General y Asociaciones

- **Inicialización:** Al crearse, se le asignan hechos iniciales como alianzas, enemigos, y movimientos posibles. Estos hechos son la base para sus decisiones.
- **Movimiento:** Decide su próximo movimiento de manera aleatoria entre los movimientos posibles.
- **Asociaciones:**
  - Decide aleatoriamente si acepta propuestas de asociación.
  - Puede proponer asociaciones aleatoriamente si ve otros agentes y no tiene ninguna alianza.

Agente Random - Ataque e Interacción con el Entorno

- **Ataque:** Decide aleatoriamente si ataca a agentes visibles en su entorno, distribuyendo los recursos de ataque de manera equitativa entre ellos.
- **Interacción con el entorno:** Procesa información sobre objetos, recursos y acciones de otros agentes.
- **Decisiones basadas en reglas:**
  - *Regla de Movimiento:* Activada si hay movimientos posibles. Elige un movimiento aleatoriamente.
  - *Regla de Ataque:* Activada si ve otros agentes; decide aleatoriamente realizar un ataque.
  - *Respuesta a Propuestas:* Considera aleatoriamente las propuestas de asociación recibidas.

Agente Pacífico - Comportamiento General y Estrategias

- **Inicialización:** Configura hechos iniciales como alianzas, enemigos, y otros agentes en el entorno, estableciendo su memoria geográfica y de ataques.
- **Estrategia General:** Prioriza la evasión y la preservación para evitar conflictos y minimizar interacciones hostiles.
- **Decisiones de Movimiento:** Se basan en un conjunto de reglas evaluadas continuamente para elegir rutas que maximicen la distancia de las amenazas.
- **Interacción Social:** Considera propuestas de asociación basándose en su estrategia de no confrontación y decide su participación de manera estratégica.

Agente Pacífico - Reglas de Decisiones Basadas en Sistema Experto

- **Regla para Moverse Lejos del Atacante:** Activa al recibir un ataque, calcula y ejecuta un movimiento que aumenta la distancia respecto al atacante, usando algoritmos como BFS para determinar la ruta más segura.
- **Regla para Observar Objetos:** Se activa cuando hay objetos visibles que pueden representar una amenaza. Evalúa y ajusta la ruta para evitar los objetos identificados como peligrosos.
- **Regla para Observar Acciones:** Analiza acciones observadas que pueden requerir una respuesta evasiva, ayudando al agente a decidir la mejor ruta de evasión.
- **Movimiento por Defecto:** Selecciona un movimiento al azar de las opciones disponibles cuando no hay amenazas inmediatas detectadas.

Agente Buscador de Comida - Comportamiento General y Estrategias

- **Inicialización:** Al crearse, se establecen hechos iniciales como recursos disponibles, posición actual y movimientos posibles.
- **Estrategia General:** Enfocado en la localización y recolección de recursos alimenticios, evitando conflictos siempre que sea posible.
- **Movimiento y Recolección:** Utiliza algoritmos de búsqueda avanzados para optimizar sus rutas hacia los recursos.
- **Interacción con el Entorno:** Evalúa continuamente su entorno para identificar recursos y evitar amenazas potenciales.

Agente Buscador de Comida - Reglas de Decisiones Basadas en Sistema Experto

- **Regla para Recolectar sin Enemigos:** Activada cuando hay recursos visibles sin enemigos cercanos. Calcula la ruta óptima hacia el recurso más valioso.
- **Regla para Recolectar con Enemigos:** Se activa cuando hay recursos y enemigos visibles. Decide si es viable atacar, ir en la busqueda de recursos o retirarse basado en la evaluación de riesgo.
- **Regla para Situaciones de Atasco:** Activa cuando está atascado y hay recursos disponibles. Busca rutas alternativas y evalúa la posibilidad de realizar movimientos estratégicos para acceder a recursos.
- **Movimiento por Defecto:** Selecciona un movimiento hacia el recurso más cercano o realiza un movimiento predeterminado en ausencia de amenazas directas y recursos visibles.

Agente de Combate - Comportamiento General y Estrategias

- **Inicialización:** Configura hechos iniciales sobre recursos, posición y enemigos, proporcionando una base para decisiones tácticas.
- **Estrategia de Recolección:** Se enfoca en maximizar la recolección de recursos cuando no hay enemigos visibles, utilizando rutas optimizadas.
- **Estrategia de Combate:** Prioriza el ataque a enemigos basándose en su fuerza relativa y la disponibilidad de recursos.
- **Asociación y Alianzas:** Busca formar alianzas cuando es estratégicamente ventajoso, especialmente en situaciones de desventaja numérica o de recursos.

Agente de Combate - Reglas de Decisiones Basadas en Sistema Experto

- **Regla para Recolectar sin Agentes Visibles:** Se activa cuando no hay agentes enemigos visibles y hay recursos accesibles. El agente calcula la ruta más eficiente hacia el recurso más valioso.
- **Regla de Ataque a Enemigo Visible:** Se activa cuando detecta enemigos en su campo de visión. El agente decide atacar basándose en la evaluación de la amenaza y su propia capacidad de ataque.
- **Regla de Propuesta de Asociación:** Se activa en condiciones de amenazas múltiples o recursos limitados. El agente propone alianzas para mejorar sus chances de supervivencia.

Agente de Combate - Reglas de Decisiones Basadas en Sistema Experto

- **Regla de Ataque cuando no hay Enemigos:** Se activa cuando hay agentes que no son enemigos en la vista, pero el contexto permite un ataque. También planifica su movimiento hacia recursos o posiciones estratégicas.
- **Movimiento por Defecto:** Selecciona un movimiento hacia el recurso más cercano o realiza un movimiento seguro cuando no hay acciones claras de combate o recolección.

Análisis de la Simulación

- Evaluación de los resultados obtenidos durante la simulación.
- Interpretación de los datos y conclusiones preliminares.

Conclusiones y Trabajo Futuro

- Recapitulación de los hallazgos clave.
- Implicaciones del trabajo y posibles direcciones futuras.