Algoritmo de Inteligencia Artificial para la Evolución de las Especies

Este algoritmo fue implementado para darle la capacidad a las entidades del proyecto para aprender de su entorno y "Evolucionar", para ser más específicos implementamos este sistema para las sociedades, convirtiéndolos formalmente en agentes de la simulación al conferirles la capacidad de tomar decisiones que pueden modificar el estado de la misma.

Centrándonos en la idea desde el punto de vista matemático, se trata de aprovechar la idea de que el sistema de características e interdependencias forma un grafo dirigido para simular el aprendizaje de las sociedades obteniendo conocimiento de los nodos que no pertenecen a la sociedad.

El conjunto de características de una sociedad es un subconjunto de los nodos del grafo, si además tomamos solo aquellas dependencias cuyo a y b pertenezcan ambos a esta sociedad, notamos que es igualmente un subconjunto de las aristas del grafo. La idea es que la sociedad solo pueda acceder a este subconjunto de aristas y nodos del grafo, para formar un subgrafo del grafo general que contiene la simulación, e intente comenzar a aprender de aquellos nodos que no pertenecen a este subgrafo y afectan positiva o negativamente en las características de la sociedad, para intentar generar nuevos nodos que afecten a esos externos en función de mejorar la supervivencia de la sociedad, lo cual más específicamente se traduce como afectar negativamente a aquellos nodos que disminuyen (directa o indirectamente) a la población de la sociedad, y afectar positivamente a aquellos que la aumenten. Como sabemos, las aristas del grafo no están ponderadas por valores, sino por funciones, así que no existe una manera directa de saber cuánto afecta una característica a otra a partir de una dependencia, luego, la sociedad intentará deducir si afecta positiva o negativamente una característica en otra.

A modo general, tenemos en las manos un problema de optimización a partir de datos, donde nuestro valor a optimizar será la población de la sociedad, y en el que simularemos un desconocimiento y aprendizaje del alrededor de la misma, para trabajar en base a los datos conocidos.

El algoritmo de Evolución reside en una clase Evolution, la cual se encuentra como atributo de Society para poder acceder a los datos de la clase.

Sus atributos son:

-society_name, el nombre de la sociedad a la que pertenece

-pos, la posición del land al que pertenece la sociedad en el grafo

-characteristic, es una lista de tuplas <nombre de característica, valor>, donde guardamos el nombre de todas las características, y un valor 0, 1 o -1, este representa si la característica a la que corresponde beneficia (1) o perjudica (-1) a Población, ya sea de forma directa o indirecta. Se define que una característica "a" afecta de forma directa a Población si existe una dependencia de "a" a población, y en el caso de indirecta nos referimos a si existe un camino en el subgrafo entre "a" y población que no es directamente solo una dependencia. En caso de que no se determine si la característica afecta o no a Población por alguna razón (porque hay más de una componente conexa por ejemplo), esta entonces tendrá como valor 0. Se infiere que la característica Población tiene como valor por defecto 1. Recalcamos que las características a las que nos referimos son solamente a las características de la sociedad, no a alguna externa a esta.

-inter_dependences, es una lista de listas, que representa dado la característica i-ésima de la sociedad, cuáles son las características tanto internas como externas que le afectan. Para dar mayor facilidad a la explicación digamos que una característica incidente es una característica que afecta a alguna de la sociedad actual en cuestión (esta característica incidente puede ser una característica externa a la población o interna) Esto se guarda de la forma: (entity, characteristic, value, pos), donde entity es el nombre de la entidad de la característica que afecta a la actual (que puede ser de un land, de la misma sociedad actual, u otra sociedad, tanto del mismo land como de otro), characteristic es el nombre de dicha característica incidente, el value es 0, -1 o 1, dependiendo de si afecta positiva (1), negativamente(-1), o de forma indeterminada (0) a la característica í-ésima, y pos, la posición en la simulación de la sociedad o land al que pertenece la característica incidente en cuestión. -values, aquí se guardan los valores de cambio que han tenido todas las características en una corrida del Move_One_Day de Simulation. Por ejemplo, si una característica "a" disminuyó la población en 3 y una característica "b" la aumentó en 10, el valor de cambio de Población será 7

-old_values, cumple la misma función que values, pero representan los datos del Move_One_Day anterior, se usan para comparar datos.

-attemps, es una lista de listas y una posición i,j que dada la í-ésima característica, representa cuántas veces hemos enviado un request a land con respecto a la característica incidente j-ésima

-attemp_succes, representa lo mismo que la lista anterior, pero en vez de la cantidad total de veces que ha sido enviado guarda la cantidad de veces que ha sido aceptado.

-request, representa el request que le envía la clase a Simulation o land (a través de society y/o land por supuesto) para crear un nuevo nodo que afecte una característica externa.

-request_pos, indica la posición de la característica que vamos a afectar con request

Estas variables y estructuras son usadas por los siguientes métodos para poder hacer funcionar correctamente el algoritmo de evolución:

-Learning_from_Interdependence(self, in_inter_dependence, value, change_value), sirve para avisarle a la sociedad actual de la existencia de una característica incidente. Recibe la interdependencia, el valor de la misma en el grafo (ojo, solo se recibe este valor y se puede usar en el método si la característica incidente también pertenece a la sociedad actual, ya que en otro caso no estaríamos simulando incertidumbre), y se guarda como 0, 1 o -1 en la lista inter_dependences, y el valor de cambio, para agregarlo a values, llamado change_value.

-Check_Characteristics_Values(self), sirve para evaluar los valores de 0, 1 y -1 guardados en characteristics, esto se logra haciendo un BFS por el grafo y revisando qué características inciden negativa o positivamente unas con otras. Para determinarlo nos basamos en el value que guardamos mediante Learning_from_Interdependence en la lista inter_dependence.

-Request_for_Evolution(self), sigue un criterio para escoger el request que va a enviar para modificar alguna característica incidente y ajena a la sociedad actual. El criterio consiste en escoger el que mayor valor absoluto tenga en la lista de los valores de cambio actual, puesto que es el mayor incidente, entonces de las características incidente aquellas descartar aquellas que antes ya hayan tenido un request exitoso, o que hayan excedido cierto número de intentos fallidos de enviar request (debido a una alta mutabilidad por parte de la característica), y al tener alguna escogida (en caso de tener varias candidatas se escoge la primera), se envía inicialmente en negativo o positivo dependiendo del valor guardado en characteristic, que determina primero si la posición actual beneficia o emperoa a la Población.

-Actualizate_Values(self), sirve para evaluar los valores de 0, 1 y -1 que estarán en las posiciones de inter_dependences, esto se hace cuando ya tienen un valor 1 o -1 de antemano (enviado por el método Request_for_Evolution), y sirve para analizar si el valor fue bien colocado. Esto lo hace revisando los valores encontrados en old_values y comparándolos con el values actual.

La idea una vez logramos concebir un request, es enviarlo a society desde evolution, society a su vez lo enviará a su entidad superior, que es Land, en este punto el Land debe analizar por sí mismo si es capaz de resolver el Request (porque la característica que se quiere afectar pertenece al propio Land), en caso de que pueda resolverlo, inmediatamente calcula si las probabilidades a partir de la mutación permiten que sea agregada la interdependencia, y avisa a society de esto (que a su vez le avisa a Evolution para marcar que fue satisfactorio el Request), en caso de que LAnd no pueda resolverlo, entonces le envía el Request a Simulation (su entidad superior), donde SImulation realiza el mismo procedimiento para Aceptar o no el Request.