T-Tree: Un simulador de tratamientos médicos

Ana P. Argüelles Terrón, Alejandro Beltrán Varela, Javier Lopetegui González y Abel Molina Sánchez

Universidad de La Habana, Cuba

Abstract. This paper will describe the process of creating **Treatment Tree(T-Tree)**, a medical treatment simulator. The structure of the project will be discussed, ranging from the creation of a domain-specific language and the entire compiler structure that would translate that language into Python code. The simulation mechanism will be presented with the strategies used for it, as well as the results obtained.

Resumen. En el presente trabajo se describirá el proceso de realización de **Treatment Tree(T-Tree)** un simulador de tratamientos médicos. Se debatirá la estructura del mismo, que va desde la creación de un lenguaje de dominio específico y toda la estructura del compilador que tanspilará dicho lenguaje a código Python. Se presentará el mecanismo de simulación con las estrategias utilizadas para el mismo, así como los resultados obtenidos.

Keywords: simulación de tratamientos · compilador · UCT · agentes · árbol de tratamientos

1 Introducción

El sistema surge a partir de la orientación de un proyecto integrador de las asignaturas de Simulación, Inteligencia Artificial y Compilación impartidas en 3er año de la licenciatura en Ciencias de La Computación de la facultad de Matemática y Computación(MATCOM) de la Universidad de La Habana(UH). Para el desarrollo del mismo se eligió por parte del equipo de trabajo un tema sobre el cual realizar la simulación. La simulación de tratamientos médicos se escogió en base al interés de tener un acercamiento desde la computación a esta área de la ciencia. Se plantearon diferentes estrategias para el enfoque de la simulación, las cuales se plantearán más adelante. En conjunción con el módulo de simulación se desarrolló un compilador con generador de parser LR(1) para la transpilación del DSL a código Python.

2 Simulación de Tratamientos

Durante el debate de las estrategias a seguir en el proyecto surgieron distintos enfoques de hacia donde debería ir enfocado el resultado del mismo. Por un lado

existió la idea de que el resultado de la simulación fuera la elección del mejor tratamiento en base a un conjunto de tratamientos(secuencia de intervenciones médicas en el tiempo) predeterminadas por el usuario, simulando la evolución del paciente con la elección de cada una de ellas y eligiendo aquella con mejor resultado. El otro enfoque planteaba que el resultado debía ser un árbol de decisiones de posibles tratamientos a aplicar de acuerdo a los distintos desarrollos de la enfermedad. Este último fue finalmente el elegido para desarrollar el modelo.

La simulación está basada en conjuntos de agentes reactivos: la enfermedad y el tratamiento, que interactúan en el ambiente(cuerpo), compuesto de diferentes parámetros cuyos valores definen sus distintos estados.

La enfermedad se define como un conjunto de *síntomas*, cada uno de los cuales es un agente.

El tratamiento se define con un conjunto de *intervenciones* posibles a aplicar, podemos verlo como el stock de posibles medicamentos disponibles para tatar a un paciente. Cada una de las intervenciones es un agente.

Los agentes pueden estar activos o inactivos, para lo cual, cada uno tiene sus condiciones de activación, que dependen del estado del ambiente y/o del tiempo de la simulación. Debido a las características del dominio, cada agente tiene un tiempo de efecto(tiempo durante el cual estará activo) y un tiempo de repetición de su acción. Por tanto, un agente estará activo durante su tiempo de efecto si se cumplen sus condiciones de activación y tendrá acción sobre el ambiente cada tiempo de repetición. De esta forma buscamos representar el comportamiento que pueden tener las enfermedades y los medicamentos, ya que estos tienden a tener efectos graduales en el tiempo más que resultados bruscos en un momento dado. Los agentes de las intervenciones también poseen una cantidad finita de activaciones, ya que sus aplicaciones también estarán condicionadas por la cantidad disponible de la misma.

Véase un ejemplo de definición de estos agentes:

Tenemos un síntoma que es la tos, que se activa si las plaquetas están bajas. La tos estará inicialmente activa durante 72h, teniendo efecto cada 4h. Luego cada 4h, va a aumentar(disminuir) algún(algunos) parámetro del ambiente.

A su vez, tenemos como posible intervención el jarabe, que se puede activar si las plaquetas están bajas igualmente, y tiene un tiempo de acción de 48h, tomándose una dosis cada 8h. Además se cuenta con solo 5 unidades del mismo. Igualmente en cada intervalo actuará modificando parámetros del ambiente.

Para la creación de las intervenciones se tiene una clase *Intervention* y para los sintomas una llamada *Syntom* ambas heredan de la clase *Agent*.

El ambiente se define como una lista de parámetros, cada uno con un valor inicial y 2 conjuntos de límites, el primero define el rango en el que se puede considerar que dicho parámetro está estable, el segundo otro rango que contiene al primero y que establece los límites extremos de este. Encontrarse contenido en el segundo pero no en el primero sirve como indicador de afección y estar fuera de este inidicaría estado de muerte del paciente.

Algoritmo de Simulación :

Para la ejecución de la simulación se parte de los siguientes parámetros:

- environment : el ambiente(parámetros sobre los que se va a realizar la simulación)
- tick : entero que representa la cantidad de horas de cada paso de tiempo en la simulación.
- end_time : el tiempo total que se quiere que duré la simulación
- simulations: la cantidad de veces que se va a realizar la simulación
- un conjunto inicial de síntomas
- un conjunto inicial de intervenciones

Cada simulación es una instancia de una clase *Simulation* que se inicializa con los parámetros definidos anteriormente. Cada tick de tiempo se simula que ocurre en el ambiente. Cada agente activo aumenta su tiempo de actividad y si se encuentra en tiempo de repetición, actúa sobre el ambiente en correspondecia. Si completó su tiempo de estar activo, pasa a formar parte de los agentes inactivos, en otro caso continúa interactuando con el ambiente. En el caso de las intervenciones se comprueba si quedan suministros de la misma luego de completado el efecto, en caso de no quedar, se elimina. Luego de actualizado el ambiente se pasa a comprobar si se cumple alguna regla de activación de alguno de los síntomas inactivos. Pasando a activarse los que las cumplan.

En este momento también se comprueba el conjunto de intervenciones inactivas, y se determinan aquellas candidatas a aplicar. De las posibles intervenciones candidatas se selecciona una y se agrega como agente activo. En el proceso de selección de las intervenciones aplicables en cada paso posible(habrá momentos donde nignuna intervención sea aplicable) se va construyendo y actualizando el árbol de tratamientos.

La construcción del árbol de tratamientos(*treatment_tree*) es el objetivo que perseguimos en la simulación y para ellos nos apoyamos en la inteligencia artificial. Este árbol es común a todas las simulaciones, y cada uno de sus nodos(excepto la raíz) representa la aplicación de una intervención en un instante de tiempo. Luego, cada rama del árbol terminará representando un tratamiento ante la enfermedad. Para la construcción del árbol de tratamientos seguimos una estrategia basada en UCT(upper confidence bounds applied to trees), política utilizada para la selección en el algoritmo de Monte Carlo Tree Search(Russell, S. et al., 2020).

Cada nodo del árbol va a tener asociado un valor de utilidad y un contador de visitas. Una vez que se alcanza un nodo final(último tratamiento aplicado cuando termina la simulación) se calcula el valor de utilidad basado en los valores del ambiente, y este se propaga por todos sus ancestros(back propagation).

El valor de utilidad lo calculamos en base a la siguiente métrica:

sea i un parámetro del ambiente:

p = valor del parámetro
ub = límite extremo superior del parámetro
lb = límite extremo inferior del parámetro
ug = límite bueno superior del parámetro
lg = límite bueno inferior del parámetro
$c = (ug - lg)/2$
$n(x) = \frac{x-lb}{ub-lb}$ función para estandarizar los valores absolutos del rango entre $[0, 1]$
$d = |n(p) - n(c)|$ si $p \in [lb, ub]$ y $p \notin [ug, lg]$
$d = 0$ si $p \in [ug, lg]$

$$uv = \sum_i \frac{1}{1+d_i}$$

esta fórmula se aplica cuando todo los valores de los parámetros se encuentran dentro del rango posible, y da mayor valor a la utilidad mientras más cerca del centro de la región estable se encuentre. Si se encuentra dentro de la región estable se suma el valor máximo posible(1).

Cuando al menos un valor de los parámetros se encuentre fuera de los límites aceptables, entonces se llegó a un estado final negativo y para calcular la utilidad, se iguala los valores(p) del resto de parámetros al límite extremos más distante del centro. De esta forma se garantiza que siempre las soluciones que no lleven al estado final al paciente tengan mejor ponderación que aquellas que por al menos un parámetro causaron un estado final.

A la hora de seleccionar qué intervención aplicar en un instante de tiempo, se compruba si ese nodo ya forma parte del árbol, en caso negativo, se agrega al mismo. Luego, para todos los nodos candidatos se calcula su valor de UCT(upper confidence bounds applied to trees) y se selecciona el de mayor UCT. La política UCT permite hacer compensación entre exploración y explotación de soluciones, dándole peso, no solo al nodo con mayor valor de utilidad, sino también relevancia a aquel con mayor cantidad de visitas. Para un nodo n, se calcula el límite de confianza superior UCB según esta política:

$$UCB = \frac{us(n)}{vc(n)} + \sqrt{\frac{\log vc(p(n))}{vc(n)}}$$

donde:
$vc(n)$ = cantidad de visitas de n
$us(n)$= valor de utilidad de n
$p(n)$ = padre de n.

Vemos como el primer término es el de explotación, nos da el valor promedio de utilidad, mientras que el segundo es el de exploración, donde el sumando va a ser mayor mientras menor sea la cantidad de visitas del nodo.

Si el nodo no ha sido explorado, o sea, su valor $vs(n) = 0$, se hace
$UCB(n) = \infty$, garantizando la exploración.

Cuando hay más de un nodo con el valor máximo, se selecciona random uno de ellos para continuar.

Una vez completadas todas las simulaciones, se tienen valores de probabilidad para cada arista, definidos como el total de visitas del nodo hijo entre el total de visitas del nodo padre. Luego, cada rama del árbol va a describir un tratamiento frente a la enfermedad, siendo el tratamiento de mayor probabilidad de resultar efectivo aquel que se construye eligiendo la arista de mayor probabilidad en cada paso.

References

1. Russell, Stuart J., Norvig, Peter., Davis, Ernest.: Artificial intelligence: a modern approach. 4th edn. (2020)