Nombre del Alumno:

UNIVERSIDAD COMPLUTENSE MADRID

Examen de Sistemas Inteligentes

28 de junio de 2024

Instrucciones generales para realizar el examen:

- **Pon tu nombre en la cabecera**
- **Contesta a las preguntas digitalmente usando Word**. Puedes apoyarte en tantas hojas en sucio adicionales como quieras, pero las respuestas deben ir en el examen de forma digital.
- **El examen escrito se entrega en formato PDF para evitar modificaciones**. Para guardar el formato pdf ir a Archivo>exportar>crear pdf. **Entregarlo en otro formato se penalizará con hasta -1 puntos en la nota del examen.**
- El tiempo para la realización del examen es de **3h**. Consta de dos partes, una teórica con problemas que se contestan en el Word de 5 puntos divididos en dos preguntas de 2.5 puntos. Y una práctica que se contesta usando Júpiter Notebook. La entrega debe contener un zip con lo siguiente en la raíz del zip: El pdf con las respuestas de la parte teórica, el notebook con la implementación de la parte práctica y los datos que se necesiten. La parte práctica debe ejecutar sin intervención del profesor, así que cuidado con las rutas (hacedlas relativas) o se restaran puntos.

Problema 1 (2,5 p)

En un almacén disponemos de una grúa que es capaz de coger cuatro tipos de objetos:

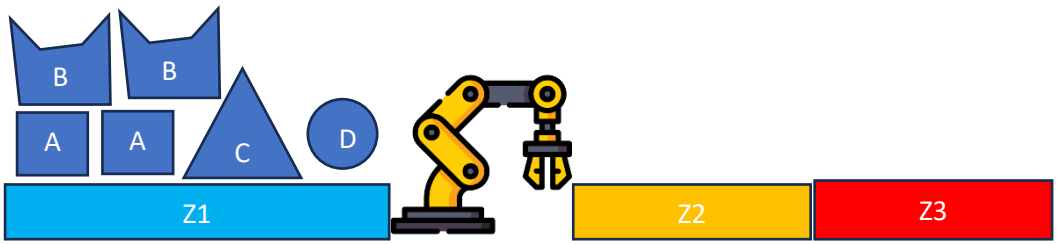
- El Bloque A es un bloque totalmente cuadrado de peso 100Kg. Ésta puede estar encima de un bloque A y un bloque B
- El bloque B es un bloque totalmente cuadrado salvo por que una de las caras, la de arriba, tiene una hendidura cóncava circular que permite colocar sobre él una esfera. El peso de este bloque es de 200Kg. Ésta puede estar encima de un boque A y un bloque B.
- Un triángulo equilátero C de 300kg de peso. Ésta sólo puede estar encima del bloque A
- Una esfera de metal D de 400Kg de peso. Ésta sólo puede estar encima del bloque B.

La grúa debe mover los bloques desde una zona donde los deposita la cadena de montaje, que denominaremos zona Z1 hasta las zonas Z2 y Z3. Puede haber objetos en la zona Z1 y Z2 que haya que retirar, para ello se pueden soltar en Z1. Dejar los objetos en Z2 y Z3 es igual de costoso. También es igual de costoso coger que soltar indistintamente de donde se haga. Solo influye en el coste el peso de los objetos. Queremos construir un agente que sea capaz de dada una situación inicial de bloques en Z1 que la grúa es capaz de percibir mediante una cámara, ésta sea capaz de colocar las cajas en Z2 y Z3 de la forma que el sistema desee. Un requisito indispensable de los promotores del proyecto es que la grúa siempre haga el menor número de combinaciones posibles para reducir al máximo la energía consumida por la grúa.

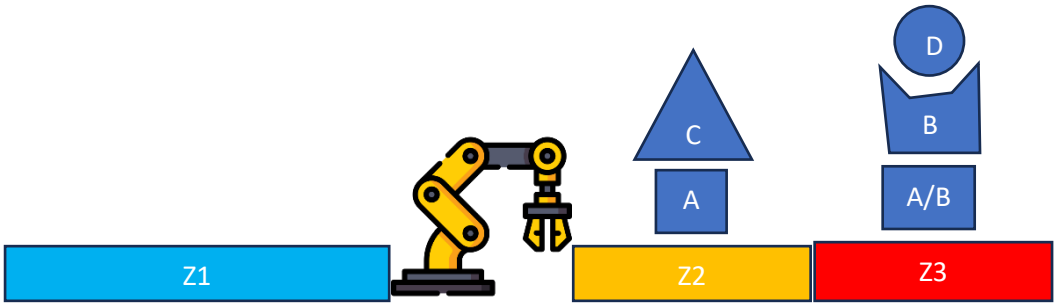
Existen por tanto 6 acciones genéricas que la grúa puede realizar:

- Coger cada tipo de bloque de la zona Z1
- Solar el bloque que previamente ha cogido en la zona Z2
- Solar el bloque que previamente ha cogido en la zona Z3
- Coger un bloque de la zona Z2
- Coger un bloque de la zona Z3
- Soltar el bloque en la zona Z1

En particular y a modo de ejemplo vamos a definir un caso concreto donde existen 2 cajas de tipo A, 2 de tipo B, una de tipo C y una de tipo D en la zona Z1.



Y queremos conseguir la siguiente configuración, donde la primera caja de Z3 puede ser una A o una B



Se pide:

Pregunta 1: (0.5 puntos) ¿Qué tipo de agente es? Razona la respuesta.

Pregunta 2: Según el tipo elegido, formaliza el problema de ejemplo para que éste pueda ser resuelto por el agente incluyendo todo lo necesario para que el agente pueda resolver el problema (1.5 puntos)

Pregunta 3: Plantea una solución alternativa en la que prime la velocidad de resolución del problema por parte del agente sobre el coste eléctrico de llevarlo a cabo. (0.5p)

Respuestas:

Pregunta 1: Es un agente deliberativo basado en objetivos, ya que debe planificar a futuro la sucesión de pasos que debe hacer. Los objetivos pueden ser cambiantes (la meta que debe conseguir puede ser distinta) y en función de la meta trazará un plan.

Pregunta 2:

Usaremos un agente basado en búsqueda heurística. Como buscamos el óptimo, necesitamos A* o sus derivados.

Representamos le problema como un problema de búsqueda: Cada elemento tiene cuatro campos, el tipo, un id para identificar a las cajas, la posición que ocupan en la pila (si están apiladas, siendo 0 no está apilada) y la zona donde se encuentra la caja: Z1, ARM (cuando está cogido por el brazo robótico/grúa), Z2 y Z3. Y ARM para identificar que el brazo está ocupado

Estado inicial: (A,1,0,Z1) (A,2,0,Z1) (B,3,0,Z1) (B,4,0,Z1) (C,5,0,Z1) (D,6,0,Z1) (ARM,0/1)

Estado final: (A,*,0,Z2) (C,1,1,Z2) (A/B,*,0,Z3) (B,*,1,Z3) (D,6,2,Z3)

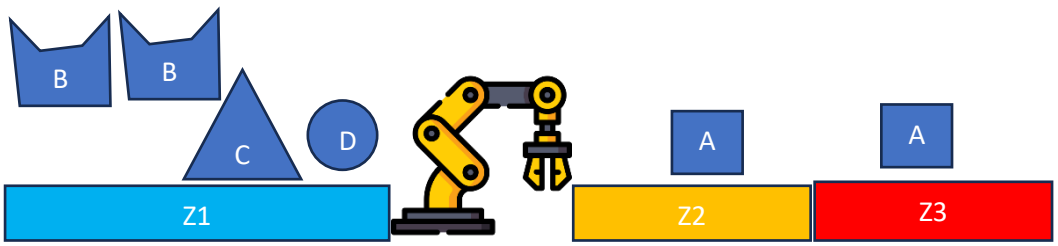
Siendo * un valor indiferente y el tipo A/B indistinto.

Operadores:

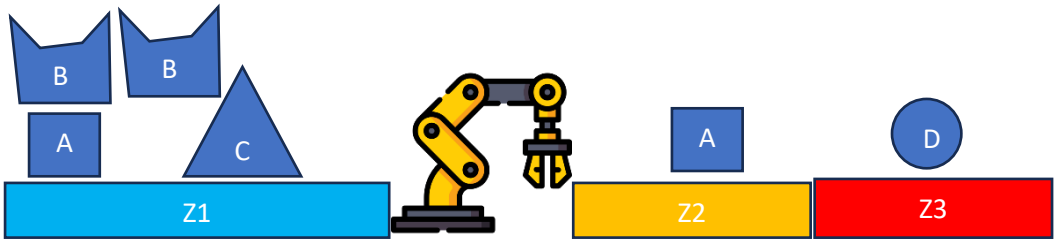
- CogerZ1(Bloque): precondición: bloque en Z1 y (AMR,0) => Bloque.zona=ARM,(ARM,1)
- CogerZ2(Bloque): precondición: bloque en Z2 y (AMR,0) => Bloque.zona=ARM,(ARM,1)
- CogerZ3(Bloque): precondición: bloque en Z3 y (AMR,0) => Bloque.zona=ARM,(ARM,1)
- SolarZ1(Bloque): precondición: bloque en ARM y (AMR,1) => Bloque.zona=Z1,(ARM,0) valor en la pila variará en función de lo que haya en Z1.
- SolarZ1(Bloque): precondición: bloque en ARM y (AMR,1) => Bloque.zona=Z2,(ARM,0) valor en la pila variará en función de lo que haya en Z2.
- SolarZ1(Bloque) : precondición: bloque en ARM y (AMR,1) => Bloque.zona=Z3,(ARM,0) valor en la pila variará en función de lo que haya en Z3.

Función heurística admisible.

Inicialmente podemos pensar en número de bloques mal colocados, por ejemplo, en este caso la función heurística daría: 3 ya que faltan por colocar 3



En este otro caso la distancia sería 4 ya que faltan por colocar 4 bloques



Como cada operación consume recursos, y se necesitan dos operaciones coger y soltar para mover una caja, la heurística afinaría más si multiplicamos el valor por 2 por cada posición pendiente de colocar, ya que al menos necesitaremos 2 movimientos, excepto en el caso de que el brazo esté ocupado por una caja, en ese caso, serian (numero de cajas mal colocadas -1)* 2 + 1.

H1()

Si (ARM,0) : num_bloques_malColocados * 2

Sino: (num_bloques_malColocados -1)*2 + 1

Podemos afinar mas si detectamos que hay una caja mal colocada, como en el caso 2 mostrado. En estos casos habrá que retirar el bloque antes de poner el que corresponda. Entonces podemos diferenciar entre bloques sin poner y bloques mal puestos.

H2()

Si (ARM,0) : num_bloques_sin_poner * 2 + num_bloques_mal_puestos * 4

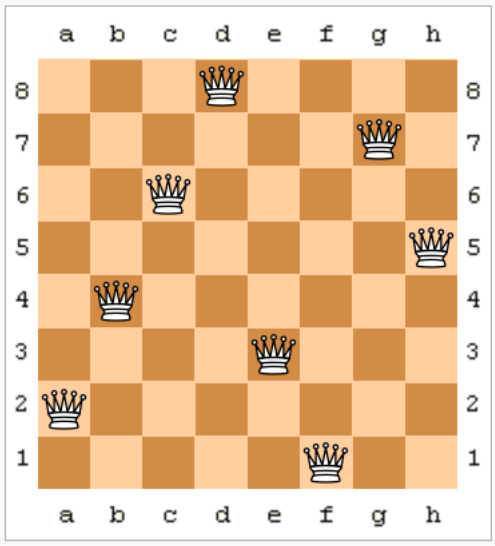
Sino: (num_bloques_sin_poner-1) * 2 + num_bloques_mal_puestos * 4 + 1

Afinando más podríamos identificar en el destino aquellos bloques mas pesados que sabemos que si o si tendrán que estar, por ejemplo en este caso tenemos 1 bloque de B mínimo, un bloque C y un bloque D, por tanto podemos saber que esos bloques moverlos implicaría 2, 3 y 4 y así ajustar mas la heurística.

Problema 2 (2,5 p)

Queremos resolver el problema de las N reinas. El problema de las N reinas es una versión modificada del problema de las 8 reinas en el que disponemos de 8 reinas para colocar en un tablero de ajedrez de tamaño 8x8. Las reinas comen en todas las direcciones, es decir en horizontal, vertical y ambas diagonales en las que se encuentre y además puede avanzar todas las casillas que ella quiera. Por lo tanto, no podemos poner dos reinas en una misma diagonal, en una misma fila o en una misma columna.

El problema de las N reinas es el mismo, pero hay que colocar N reinas en un table de tipo NxN. N puede ser cualquier valor, pero para concretar el problema, podemos poner como ejemplo el problema de las 20 reinas que se deben colocar en un tablero de 20 x 20.



Se pide:

Ejercicio 1: ¿Qué tipo de agente es? Razona la respuesta. (0.5p)

Ejercicio 2: Modela el agente para que resuelva el problema con el algoritmo que consideres más adecuado (1,5p).

Ejercicio 3: Sin describir como hacerlo, explica que otros algoritmos podrían resolver el problema y justifica porqué no los has usado. (0.5p)

Respuestas:

Ejercicio 1: Es un agente basado deliberativo basado en objetivos por el mismo motivo que antes.

Ejercicio2: podríamos modelarlo con A* el problema es que el tamaño para N=20 es inmenso y tardaría demasiado. Vamos a utilizar otro enfoque usando genéticos.

Pondremos un vector de 40 enteros, donde el gen ocupará dos valores, la posición x e y.

Los enteros pueden valer de 1 a 20 representando las posibles casillas (podríamos usar letras para seguir con la nomenclatura del ajedrez). Las operaciones de cruce y mutación son simples y no tiene mayor problema.

Las soluciones probablemente serán invalida, así que no consiste en puntuar con el fitness aquellas soluciones validas o no válidas, si no soluciones parcialmente válidas. Contaremos con 1 punto las reinas que estén bien colocadas. Una reina está bien colocada si no es comida por ninguna otra reina. Cuantas mas reinas haya bien colocadas, mas cerca estaré de la solución.

Ejercicio 3: Pues cualquier algoritmo de búsqueda parcial, como Hill Climbing, Simulated Simulated annealing, A* / voraz o búsqueda en anchura/profundidad. Los óptimos son demasiado caros y los otros como HClimbin o SA perfectamente podrían haber sido usados, pero no se han explicado en clase por lo que pongo la solución con el que se ha visto. Voraz es probable que no encuentre solución salvo que la heurística sea muy buena.

Ejercicio práctico: (5 puntos)

En el laboratorio, se entrega un Jupiter Notebook con las respuestas. El dataset está subido al campus.

Disponemos de un dataset sobre la adoptabilidad de mascotas y queremos utilizarlo para construir un sistema que se pondrá en funcionamiento dentro de un juego multijugador del estilo de NintenDogs o NintentCats donde los jugadores se conectan online y pueden adquirir una mascota pero en un momento, debido a que por ejemplo no le gusta su carácter o simplemente no son capaces de cuidarlo correctamente, pueden deshacerse de ella en el juego y devolverla el servicio de adopción. Así el sistema les predice cual puede ser el desenlace de la mascota si finalmente deciden dejarla en un centro de acogida y cambiarla por otra. El juego premiará o castigará las decisiones correctas de forma que, si un cuidador decide dejar en adopción una mascota que no es fácilmente adoptable y esta al cabo del tiempo finalmente no es adoptada por otro jugador, se le castigará reduciendo su valoración como cuidador de forma sustancial. El sistema opcionalmente podría permitir explicar el motivo por el cual la mascota no es fácilmente adoptable, aunque esta característica se considera algo deseable pero no imprescindible.

Nuestra labor como ingenieros de datos es construir el modelo para que sea usado por la empresa desarrolladora. Los datos del dataset son los siguientes:

- PetID: identificador único de cada mascota
- PetType: tipo de mascota (perro, gato, etc)
- Breed: Raza especifica de la mascota
- AgeMonths: edad de la mascota en meses
- Color: Color de la mascota
- Size: tamaño de la mascota (pequeña, mediana, grande…)
- WeightKg: peso en KG
- Vaccinated: Si está vacunado o no.
- HealthCondition: Si la mascota está sana o tiene alguna enfermedad
- TimeInShelterDays: Tiempo que lleva la mascota en el refugio
- AdoptionFee: Cuota de adopción en dollares.
- PreviousOwner: Si la mascota tuvo o no otros dueños.
- AdoptionLikelihood: Probabilidad de que la mascota sea adoptada. 0 improbable, 1 probable.

Se pide:

Ejercicio 1: Limpia el dataset de errores y formatéalo para poder ser usado por cualquier algoritmo. La idea es que la limpieza sea común a los diferentes algoritmos usados para que la comparativa posterior tenga sentido. (1p)

Ejercicio 2: Muestra la distribución del datase mediante una gráfica, hazlo de la mejor forma posible. (1p)

Ejercicio 3: Utiliza un perceptrón multicapa para construir el clasificador. El porcentaje de validación debe ser del 20% y un random_state de 13 tanto en la partición como en el uso del algoritmo. Se espera un accuracy igual o mayor al 91%. (1p)

Ejercicio 4: Utiliza un árbol de decisión para crear un segundo clasificador. El porcentaje de validación debe ser del 20% y un random_state de 13 tanto en la partición como en el uso del algoritmo. Se espera al menos un accuracy del 91%. (1p)

Ejercicio 5: Muestra las métricas que consideres adecuadas para ambos modelos y determina de forma argumentada cuál crees que es el mejor para clasificar este problema. (1p)