Práctica Inteligencia Artificial 19-20

Respecto a la práctica

La práctica de la asignatura busca enseñar como un mismo enunciado puede ser resuelto utilizando dos tipos de técnicas de Inteligencia Artificial vistas en la asignatura.

Para el desarrollo de la cada una de las partes, se proporcionará un proyecto JAVA, exportado de Eclipse, que permita a los alumnos tener un mismo framework definido sobre el que programar la solución final.

Cada proyecto se proporcionará el mismo el día de la presentación de cada parte, y estará ajustado para realizar la implementación de la solución. Además, se proporcionará una guía del manejo de cada uno de ellos.

La práctica se realizará en parejas o de manera individual. Los alumnos deberán formar los grupos y comunicárselo a los profesores mediante el formulario correspondiente en Aula Global. Se podrá utilizar un foro para buscar compañero en caso de no tenerlo. Se establece como fecha límite para generar los grupos el jueves 05 de marzo de 2020.

La entrega de todas las partes la realizará un único miembro del grupo y la fecha límite será el viernes 10 de mayo de 2020, a las 23:55 horas.

Cualquier duda sobre las prácticas, por favor, contactad con **ambos** profesores de prácticas en el mismo correo para resolverla cuanto antes. Emails:

- damigo@inf.uc3m.es
- davsanch@inf.uc3m.es

Competición opcional

Este año hemos decidido introducir una pequeña competición sobre los resultados de las prácticas. El objetivo de esta se explica dentro de cada apartado.

La participación en la competición requiere el funcionamiento de todas las tareas de cada parte, los profesores considerarán cuando se aplica para cada grupo.

En la competición, el grupo con el mejor resultado obtendrán hasta un punto extra sobre la nota. Mientras que los dos siguientes grupos obtendrán hasta 0,5 extra sobre la nota.

En caso de existencia de empates, se valorará el caso por los profesores para decidir el ganador teniendo en cuenta el resto de la práctica.

En ningún caso se podrá superar la nota de 10 sobre la práctica, pero esa puntuación extra puede ayudar a alcanzar esa nota.

Enunciado

La Universidad Carlos III de Madrid está buscando mejorar el mantenimiento del campus de Colmenarejo. Para ello ha pedido a sus alumnos que propongan un sistema inteligente que realice la gestión diaria de las tareas de dichos trabajadores.

Se ha consultado con los trabajadores del campus y han dividido el mismo en varias áreas mostradas en la Ilustración 1. También han indicado que de media tardan cinco minutos en moverse entre zonas adyacentes en el mapa:

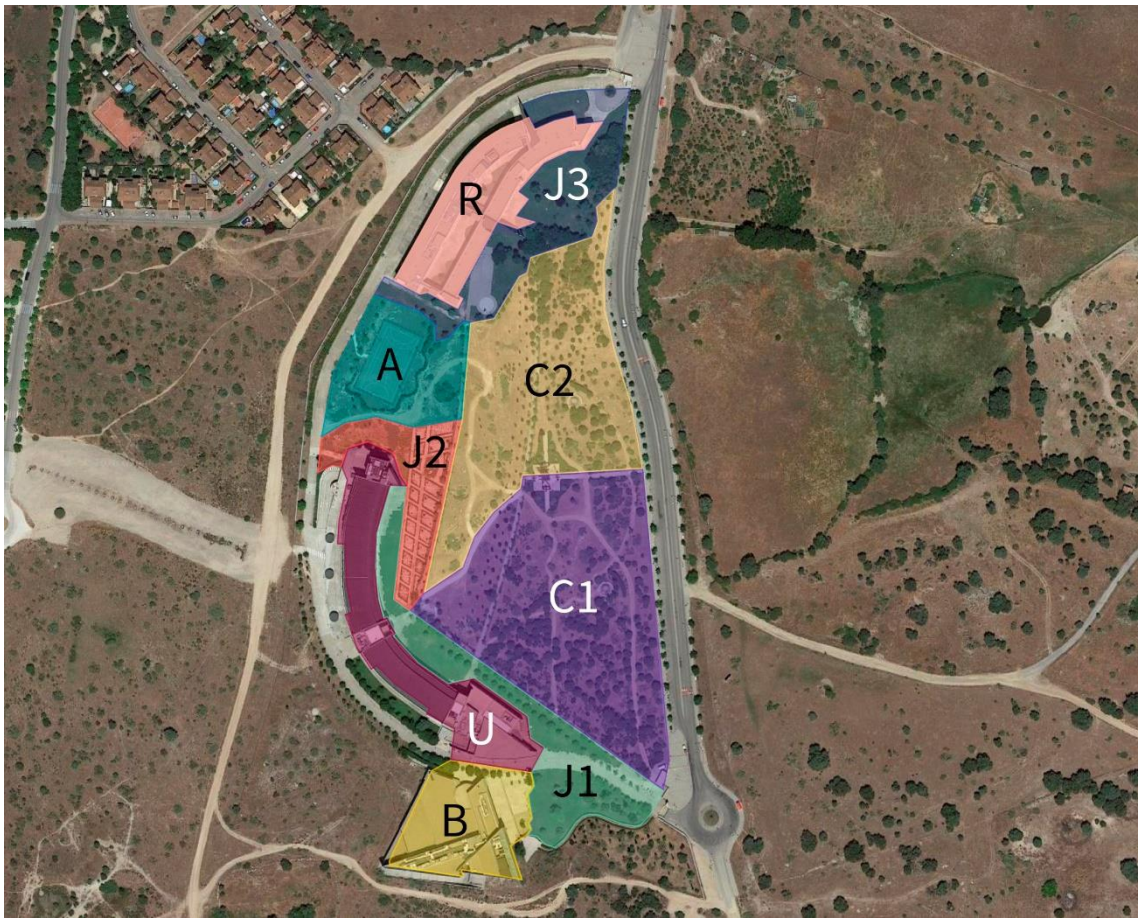

Ilustración 1. Mapa de áreas del campus

Además, han indicado una serie de trabajos principales a realizar en cada área del campus: Podar, limpiar y reparar. Para realizar cada uno de los trabajos es necesario el uso de una herramienta en concreto que tienen disponible al comienzo del día en el almacén. Cada trabajador puede llevar una herramienta como máximo. Al final del día las herramientas deben estar guardadas en el almacén.

Cada uno de los trabajadores nos ha indicado la habilidad que tiene, sobre diez, para realizar cada uno de los distintos trabajos.

Parte 1. Representación e inferencia

En primer lugar, se va a aproximar el problema mediante representación e inferencia, modelando los objetos y las reglas mediante la librería JEOPS.

A continuación, se presentan las consideraciones para tener en cuenta para modelar el problema.

Consideraciones sobre problema básico

Para adaptar el problema a el tipo de solución a desarrollar, se proponen las siguientes suposiciones:

- Para modelar la cantidad de trabajo a realizar en cada área, cada trabajo (poda, reparar o limpiar) se divide en una cantidad de tareas concreta. Por ejemplo, un trabajo de poda en el que existan 10 tareas se representa dentro del sistema como un trabajo de tipo poda con un valor de 10 unidades.
- La habilidad de cada trabajador en un trabajo se ha estimado que se traduce en tareas realizadas de dicho trabajo cada 60 minutos. Por ejemplo, un trabajador con habilidad 5 de poda completará en 60 minutos 5 unidades de poda.
- Para realizar los distintos trabajos es necesario hacer uso de las herramientas, las cuales, al igual que los trabajadores, comienzan el día dentro del área Almacén.

Consideraciones sobre problema avanzado

Adicionalmente a todo el trabajo anterior, un sistema más complejo deberá tener en cuenta las siguientes consideraciones:

A. Existen varios tipos de herramientas para cada trabajo. Las herramientas tienen una relación directa con el número de tareas que realiza un trabajador en 60 minutos. En concreto, se suma el número de mejora al número de tareas que completa el trabajador en 60 minutos. Por ejemplo, un trabajador que realizase 5 tareas de poda cada 60 minutos, con la motosierra completará 12.
B. El número de cada tipo de herramienta es limitado.
C. El desplazamiento de los trabajadores cuando llevan una herramienta está afectado por un consumo extra de tiempo por el peso. En concreto, se suma al desplazamiento realizado el número de kilos que pesa dicho objeto. EJ: Mover el aspirador en un área adyacente en vez de 5 minutos tardará 10, pero con unas tijeras de podar tardará 6 minutos.
D. La tarea de poda genera restos orgánicos que deben ser limpiados. Por ello, una vez se complete una unidad de poda en un área concreta, se generará una unidad de limpieza en dicha área.
E. Se debe realizar la cuenta de horas trabajadas por cada trabajador y mostrar las mismas al final de la ejecución.

Tareas

Para el problema básico (6,5 puntos):

- (1 puntos) Modelar y justificar en memoria los distintos elementos del problema para que funcionen en un sistema basado en el conocimiento basado en JEOPS.
- (4 puntos) Modelar y justificar en memoria las reglas de producción para que el problema propuesto funcione.
- (1,5 puntos) Funcionamiento correcto del sistema modelado para la resolución del problema propuesto.

Para el problema avanzado (3,5 puntos):

- (1 punto) Modelar y justificar lo relativo a la consideración "A" del problema avanzado.
- (0,5 puntos) Modelar y justificar lo relativo a la consideración "B" del problema avanzado.
- (0,5 puntos) Modelar y justificar lo relativo a la consideración "C" del problema avanzado.
- (1 punto) Modelar y justificar lo relativo a la consideración "D" del problema avanzado.
- (0,5 puntos) Modelar y justificar lo relativo a la consideración "E" del problema avanzado.

Problema propuesto

Tareas por zona			
Área	Podar	Limpiar	Reparar
Residencia (R)	0	6	0
Universidad (U)	0	5	5
Biblioteca (B)	0	0	3
Campo 1 (C1)	0	0	0
Campo 2 (C2)	5	8	6
Jardín 1 (J1)	10	0	5
Jardín 2 (J2)	2	0	8
Jardín 3 (J3)	12	0	0
Almacén (A)	-	-	-

Habilidad de trabajadores			
Nombre	Habilidad Podar	Habilidad Limpiar	Habilidad Reparar
Antonio	5	5	5
Bernardo	6	4	1
Cristian	2	8	2
Diego	3	3	6

Características de herramientas				
Herramienta	Trabajo	Peso*	Mejora*	Cantidad*
Tijeras de podar	Podar	0kg	+0	4
Escoba	Limpiar	0kg	+0	4
Destornillador	Reparar	0kg	+0	4
Motosierra*	Podar	10kg	+7	1
Aspirador*	Limpiar	5kg	+3	2
Caja de herramientas*	Reparar	3kg	+1	3

**Solo aplica para la consideración del problema avanzado que corresponda*

Competición (opcional)

La competición solo se aplicará sobre el **sistema avanzado**, funcionando la misma de forma completa. El sistema se probará sobre un problema distinto al propuesto, aunque se mantienen las áreas, el numero de trabajadores y las herramientas. Se modificarán las características propuestas (los números dentro del fichero de entrada).

El objetivo es minimizar el tiempo total trabajado, pero se quiere evitar una sobrecarga de los trabajadores, lo que implica un balanceo entre el trabajo realizado por los mismos.

Como métrica de estos dos elementos, se tomará el máximo de horas realizadas por los trabajadores y se premiarán las prácticas con el mínimo máximo.

Ejemplo:

Grupo	Trabajador 1	Trabajador 2	Trabajador 3	Trabajador 4	Máximo
A	1	1	8	0	8
B	3	2	4	2	4
C	5	2	6	1	6
D	5	6	17	4	17

La mejor solución es la proporcionada por el grupo B.

Parte 2. Búsqueda

Para esta segunda parte se va a aproximar el problema mediante búsqueda heurística, modelando distintas funciones que actúan dentro de un algoritmo A* modelado dentro de Java.

A continuación, se presentan las consideraciones para tener en cuenta en esta parte.

Enunciado específico parte 2

La segunda aproximación se busca ayudar a los empleados indicándoles cual sería el orden de trabajo mas eficiente, para ello los empleados tendrán asignados una serie de trabajos a realizar desde el principio del día.

En concreto es necesario calcular el día de trabajo de Antonio para la siguiente lista de tareas:

Área	Podar	Limpiar	Reparar
Residencia (R)	0	6	0
Universidad (U)	0	5	5
Biblioteca (B)	0	0	3
Campo 1 (C1)	0	0	0
Campo 2 (C2)	5	8	6
Jardín 1 (J1)	10	0	5
Jardín 2 (J2)	2	0	8
Jardín 3 (J3)	12	0	0
Almacén (A)	-	-	-

Tareas parte 2

Para el problema básico (5 puntos):

- (1,5 puntos) Partiendo de una heurística inicial que hayas propuesto para el problema, explica el proceso de mejora seguido para obtener la heurística final del problema básico y su comparación respecto a las fases iniciales. Para realizar el análisis se deben proponer e implementar métodos para obtener estadísticas de los resultados (cadena recorrida, espacio de búsqueda observado, métricas de expansión de nodos, etc.).

- (2,5 puntos) Desarrollar una implementación de búsqueda heurística haciendo uso del algoritmo A* proporcionado. Introduciendo en el mismo la heurística propuesta, las acciones posibles del problema y el cálculo de costes de las acciones.
- (1 punto) Funcionamiento correcto del sistema modelado para la resolución del problema básico.

Para el problema avanzado (5 puntos) se deben integrar las consideraciones del problema avanzado de inferencia. Además, se debe resolver el problema considerando a todos los trabajadores:

- (2 puntos) Partiendo de la heurística final del problema básico, continúa la explicación del proceso de mejora para llegar a la heurística final del problema avanzado.
- (2 puntos) Partiendo de la implementación de búsqueda heurística del problema básico, introduce y explica las modificaciones realizadas para cada una de las consideraciones del problema avanzado.
- (1 punto) Funcionamiento correcto del sistema modelado para la resolución del problema avanzado.

Competición (opcional)

El sistema se probará sobre un problema distinto al propuesto, aunque se mantienen las áreas, el trabajador y las herramientas. Se modificarán las características de los mismos (los números dentro del fichero de entrada).

El objetivo es minimizar el tiempo total trabajado, pero se quiere evitar una sobrecarga de los trabajadores, lo que implica un balanceo entre el trabajo realizado por los mismos.

No se busca obtener el mejor resultado explorando todos los nodos, por lo que se evaluarán otros factores del algoritmo de búsqueda (heurística implementada, tiempo de ejecución, nodos explorados, etc.) para decidir el vencedor.

Código a modificar

Ya que el código base puede resultar un poco confuso, para evitar problemas se van a identificar los elementos a modificar en vuestra entrega. No obstante, también se han introducido comentarios en el propio código que indican los aspectos que se deben modificar del código base.

practica.json

No se puede modificar nada de la clase *LectorJSON.java*. El problema entregado (*problema.json*) contiene la configuración propuesta a lo largo de este enunciado y es con el que se realizará la corrección, aunque se puede hacer las modificaciones que se consideren oportunas para realizar las pruebas que se deseen.

El fichero utilizado para la competición será uno distinto al propuesto dentro del enunciado.

practica.objetos

Se permite generar todos los objetos auxiliares necesarios. También se pueden modificar los existentes, pero solo añadiendo nuevas variables y métodos, no eliminar lo existente.

Es importante no modificar la llamada a los constructores existentes. Si se necesita añadir valores no constantes de las nuevas variables (EJ: un ID) se debe hacer mediante el uso de *setters* fuera del constructor. En caso de que se desee añadir un valor estático a la variable, se puede introduciendo el valor directamente en el constructor. Por ejemplo, si quiero una variable "a" que siempre inicializara con valor 0 dentro de Tarea, podría modificar el constructor de la siguiente manera:

```
public Tarea(String tipo, String area, int unidades) {
	this.tipo = tipo;
	this.area = area;
	this.unidades = unidades;
	// Añadir el estado inicial (estático) de las variables que se añadan
	// Si se necesita añadir valores variables, utilizar setters
	this.a = 0;

}
```

practica.inferencia

De la clase *MainClass.java* no se debe modificar las partes marcadas como tal (existe un comentario previo y uno posterior a las partes no modificables).

Los métodos para imprimir información del estado del problema y de las métricas de ejecución deben ser modificados con las propuestas que se realicen.

El fichero de reglas puede ser modificado sin ninguna restricción salvo las existentes en el propio lenguaje.

Es muy importante que la inteligencia quede dentro de las reglas generadas y del motor de inferencia. Aunque se usen métodos auxiliares dentro de los objetos para facilitar

ciertos cálculos, estos no deben buscar resolver el problema. Por ejemplo, un método que calcule la distancia entre dos puntos sería correcto, pero no lo sería uno que compute el camino óptimo.

practica.busqueda

De la clase *MainClass.java* no se debe modificar las partes marcadas como tal (existe un comentario previo y uno posterior a las partes no modificables).

El método para imprimir las métricas de ejecución debe ser modificado con las propuestas que se realicen. Para imprimir el estado se modificará el método *printNodeData* de la clase *Node.java*.

La clase *OpenList* no debe ser modificada, ya que su única funcionalidad es generar una lista entre nodos, la cual se ordenará mediante la evaluación.

La clase *AStar.java* se debe modificar el método *addAdjacentNodes* que se encarga de generar los nodos hijos de un nodo padre a partir de las distintas acciones del problema, para insertar los nuevos nodos se debe utilizar el método *insertAtEvaluation* de la lista abierta.

En la clase *Node.java* se deben modificar:

- Los distintos constructores (menos el que está vacío), los cuales tienen la función de crear un estado del problema y de copiar un estado del problema existente. Es importante que en este último no se realicen copias de los objetos, sino que se creen nuevos objetos en los que introducir su información, porque en caso contrario al modificar un nodo se podría modificar la información del nodo copiado.
- El método de *computeHeuristic* con la heurística propuesta y el método *equals* que se encarga de evaluar cuando el estado de dos nodos es el mismo, de este último es importante considerar que información del nodo pertenece al estado y que información no representa el estado del problema.
- Además, es posible generar cualquier método auxiliar que necesitéis y *getters* y *setters* necesarios para vuestras variables.

Formato de entrega

Se deberá realizar una única memoria que contenga el desarrollo de las dos partes de la práctica, con el siguiente formato "IA_NIA1_NIA2_MEMORIA.pdf".

Por otro lado, se pide el código exportado de Eclipse con el formato:

- IA_NIA1_NIA2.zip

Nota: sustituir NIA1 y NIA2 por el NIA de cada miembro de la pareja.

Nota 2: Cada uno de estos dos elementos se deberá entregar dentro del entregador que corresponda de aula global, la memoria en el de Turnitin y el código en el entregador normal.

Criterios de evaluación

Esta práctica abarca el 30% de la nota final destinada a prácticas, repartiéndose de la siguiente manera:

- Parte 1. 50% de la nota de prácticas → 1.5 puntos de la nota final.
- Parte 2. 40% de la nota de prácticas → 1.2 puntos de la nota final.
- Presentación de la memoria. 10% de la nota de prácticas → 0.3 puntos de la nota final.

Todas las tareas valoran con el porcentaje definido en su apartado, sumando en total 10 puntos por parte. Es importante tanto el trabajo realizado en JAVA como la explicación del mismo en la memoria. Es decir, es importante tanto el trabajo realizado como la explicación de las decisiones que se tomen. ¡No descuidéis la memoria!

Otros aspectos a tener en cuenta son:

- Los ejemplos de prueba que hayáis empleado para validar vuestro sistema deben ser presentados y explicados correctamente en la memoria, así como entregados.
- En la corrección se emplean algunos procesos automáticos que son muy sensibles a los casos en los que no se cumplen las especificaciones, lo cual incidirá negativamente en la nota.
- Se valorará la originalidad de los diseños que presentéis, cosa que debéis documentar expresamente. Por ejemplo, si consideráis que vuestro sistema está especialmente optimizado en cuanto al número de reglas, o del tiempo de ejecución, o que emplea un algoritmo alternativo o mejor (consultad con los profesores).