UNIVERSIDADE DA CORUÑA

Inteligencia Artificial
4º Ingeniería Informática. Curso 2008/09
28 de Noviembre de 2008
Práctica 1: Implementación de métodos de búsqueda

1. Problema

Se trata de aplicar técnicas de búsqueda para resolver un sencillo juego recreativo con números de tres cifras entre el 000 y el 999. Se parte de un número inicial I y un número objetivo O y de las operaciones $SUMA$ y $RESTA$. En cada paso del juego se puede transformar el número actual sumando o restando una unidad a uno de sus dígitos. Por ejemplo, podemos pasar del 734 al 744 o al 634. Tenemos además un conjunto de números prohibidos de forma que ninguna transformación puede llevar a alguno de estos números. Además no se podrá sumar 1 al dígito 9 o restar 1 al 0, ni tampoco cambiar el mismo dígito en dos movimientos sucesivos. La solución al puzzle consiste en pasar de I a O en el menor número de movimientos posibles.

En esta práctica se debe dar solución al problema formulándolo como un problema de búsqueda. Se implementan distintos algoritmos de búsqueda ciega e informada, se evalúan y se extraen las conclusiones pertinentes.

2. Objetivos

Los objetivos de esta práctica son los siguientes:

1. Aplicar estrategias de búsqueda *ciega* y búsqueda *informada* con el fin de planificar una secuencia de acciones que permitan encontrar una solución cumpliendo con las restricciones del problema.
2. Realizar una comparación entre los distintos métodos aplicados.
3. Determinar el método de búsqueda más adecuado para este tipo de problemas.
4. Desarrollar heurísticas apropiadas para el problema, valorar su aportación y evaluar su calidad.

En cuanto a las estrategias de búsqueda ciega, se deberá realizar un análisis del problema que ayude en la elección de uno de los dos métodos clásicos (recorrido en anchura o profundidad) **antes** de implementarlos. Esta elección deberá **justificarse** en términos del objetivo del problema y los costes computacionales y espaciales de los algoritmos. Como estrategias de búsqueda informada se aplicarán los métodos de ascensión de colinas (hill-climbing[1]), y el algoritmo A*.

3. Implementación

La práctica se podrá realizar en cualquier lenguaje siempre y cuando su elección esté justificada, con la restricción de que debe ser ejecutable bajo UNIX/Linux en las máquinas asignadas a las prácticas de la asignatura sin necesidad de ninguna plataforma o entorno de desarrollo

[1]Dada la variedad de implementaciones del algoritmo de ascensión a colinas, se admitirá cualquiera de las que se encuentran en la bibliografía recomendada al final de este documento

(p.e. Netbeans, Eclipse, etc...) El software desarrollado deberá poder ser ejecutado en la línea de comandos, como un applet en un navegador, etc.

Se incluirá la implementación de una **sencilla** interfaz (se recuerda que éste no es el objetivo de la práctica) que permita:

1. Indicar el estado inicial del problema, I, es decir, es decir, el número I de partida.
2. Seleccionar el estado meta deseado, O, es decir, es decir, el número O final.
3. Seleccionar el número de números tabú. Si es mayor que cero, elegir si se indican explícitamente (a través de un fichero de texto o del teclado) o si se generan aleatoriamente.
4. Seleccionar el método de búsqueda y la función heurística, en su caso, a aplicar.
5. Visualizar la solución mostrando el camino desde el estado inicial I al objetivo O junto con todos los pasos seguidos. Es decir, la salida por pantalla deberá permitir seguir la traza de ejecución del algoritmo seleccionado. Para cada paso del algoritmo se mostrará:

   - Algoritmos de Búsqueda Informada:
     - Lista de nodos abiertos (estados candidatos para continuar la búsqueda)
     - Mejor nodo seleccionado de entre los candidatos
     - Camino actual hasta el mejor nodo seleccionado
     - Nuevos descendientes generados
   - Algoritmo en Anchura:
     - Nivel actual de exploración (lista de nodos abiertos)
   - Algoritmo en Profundidad
     - Camino actual
     - Nuevos descendientes generados
     - Mensaje en caso de *backtracking*

   Para mostrar los nodos del árbol, se seguirá obligatoriamente la siguiente estructura:

   - Algoritmos de Búsqueda Ciega: (ID)
   - Algoritmo Ascensión de Colinas: (ID, $h(n)$)
   - Algoritmo A*: (ID, $g(n)$, $h(n)$)

   donde ID es el nuevo nodo generado, $h(n)$ es el valor de la heurística y $g(n)$ es el coste del camino óptimo actual (encontrado hasta ese momento).

En cualquier caso, al final de la ejecución se mostrará el camino encontrado y el coste del camino. Si no fuese posible encontrar una solución al problema se mostrará un mensaje indicando que el problema no es resoluble.

Durante el período de realización de la práctica se habilitará un apartado *Preguntas frecuentes* en la Facultad Virtual `http://fv.udc.es`

4. Memoria de prácticas

La realización de la práctica exige la redacción y entrega de una memoria escrita, dividida en dos partes, que se ajustará a las normas publicadas en la web (se facilitarán modelos de documentos en LaTeX o Microsoft Word/OpenOffice) y contemplará **obligatoriamente** los siguientes apartados.

4.1. Primera parte

La primera parte de la memoria de prácticas tendrá una extensión total máxima de 10 páginas (excluyendo portada, resumen e índice).

1. **Portada.** Indicará el título (*Resolución de problemas de búsqueda. Memoria de Prácticas de Inteligencia Artificial. Primera Entrega*), los autores, el "login" de cada uno, el directorio de depósito y la fecha.

2. **Resumen.** Escribir con no más de 150 palabras los objetivos fundamentales de esta parte de la práctica.

3. **Índice.** Incluir la tabla de contenidos.

4. **Métodos.** (Página 1)

   a) Definir los conceptos de:
   - Estado inicial
   - Estado meta
   - Conjunto de reglas que describen las acciones (operadores y sus restricciones) disponibles.
     Estos tres elementos definen el *espacio de estados* del problema
   - Prueba de meta
   - Función de coste (algoritmo A*)

   y aplicar las definiciones anteriores al problema en cuestión para describir **formalmente** estos conceptos.

   b) Análisis que justifique la elección de uno de los dos métodos de búsqueda ciega (anchura o profundidad) para este problema teniendo en cuenta los requisitos (teóricos) de memoria y tiempo de ejecución y sus características en cuanto a óptimos y completos.

   c) Definir el concepto de función heurística. Proponer al menos dos heurísticas para el problema. Para cada una de ellas describir:
   1) La expresión en términos matemáticos
   2) Una justificación coherente acerca de su idoneidad
   3) La demostración de que en ningún caso sobreestima el coste real
   4) Una explicación acerca de si la heurística da lugar a la aparición de mínimos locales en el espacio de estados. Un espacio de estados contiene un mínimo local si la heurística proporciona un valor menor (mejor) para un estado que para otro que realmente se encuentra más cerca de la meta

5. **Bibliografía**

4.2. Segunda parte

1. **Portada.** Indicará el título (*Resolución de problemas de búsqueda. Memoria de Prácticas de Inteligencia Artificial. Segunda Entrega*), los autores, el "login" de cada uno, el directorio de depósito y la fecha.

2. **Resumen.** Escribir con no más de 150 palabras los objetivos fundamentales de esta parte de la práctica.

3. **Índice.** Incluir la tabla de contenidos.

4. **Resultados.** (Página 1)

a) Ejemplos de ejecución.

Se mostrará la traza completa de la ejecución de cada uno de los tres algoritmos de búsqueda (al menos en un ejemplo) siguiendo estrictamente el formato expresado en el apartado 3.

b) Caracterización de la calidad de las heurísticas empleadas.

Sea $h^*(s)$ la función que devuelve el coste real de un camino de coste mínimo desde el estado s hasta el estado meta O. La función de evaluación heurística $h(s)$ es una estimación de $h^*(s)$. Una heurística admisible es aquella que nunca sobrestima $h^*(s)$: $h(s) \leq h^*(s)$. La medida de precisión de heurísticas que utilizaremos será el *error absoluto esperado*: $E(h^*(s) - h(s))$. Este error se puede calcular computando la media de los errores absolutos a través de un conjunto n de estados aleatoriamente seleccionados (en nuestro caso $n \geq 15$). Una vez obtenidos los valores implicados, se construirá una tabla con la siguiente cabecera:

Tabla 1: Tabla comparativa de rendimiento de heurísticas

Ejemplo	$h(s)$				$h^*(s)$
estado s_1	$h_1(s_1)$	$h_2(s_1)$	...	$h_n(s_1)$	...
...	...	...	...	...	...
estado s_n	$h_1(s_n)$	$h_2(s_n)$	...	$h_n(s_n)$	...
Promedio	$E(h_1(s))$	$E(h_2(s))$	...	$E(h_n(s))$	$E(h^*(s))$

donde el estado s_i, con $1 \leq i \leq n$, es un ejemplo en particular, y $h_1(s_i)$, $h_2(s_i)$, ..., $h_n(s_i)$ son los valores correspondientes a las heurísticas propuestas (al menos 2) en s_i, y E es el operador esperanza.

c) Comparación del rendimiento de los distintos métodos de búsqueda implementados.

Se construirá una tabla con la siguiente cabecera:

Tabla 2: Tabla comparativa de rendimiento de métodos de búsqueda

	Coste de la búsqueda					Factor Ramificación Eficaz				
d	Ciega	Hill(h1)	Hill(h2)	A*(h1)	A*(h2)	Ciega	Hill(h1)	Hill(h2)	A*(h1)	A*(h2)
-	-	-	-	-	-	-	-	-	-	-

Para rellenarla es necesario generar n problemas aleatorios, resolverlos con los distintos métodos de búsqueda implementados, y *agruparlos* en función de la longitud de la solución obtenida para obtener *valores promedios* en cada una de las filas.

El significado de los parámetros de la tabla es el siguiente:

- d: es el valor de profundidad del árbol-grafo de exploración en el cual se ha hallado la solución.
- Coste de la búsqueda: número total de nodos seleccionados para expandir durante el proceso de búsqueda (nodos en la lista de cerrados).
- Factor de ramificación efectivo: Si el número total de nodos generados es N y la profundidad de la solución es d, entonces $b*$ es el factor de ramificación que tendría un árbol uniforme de profundidad d con N nodos[2]. Esta medida representa el número medio de caminos intentados para cada estado. Si el factor de ramificación efectivo fuera 1, nuestro camino de búsqueda obtenido por el algoritmo sería un camino solución. Es decir, cuanto más se aproxima a 1 nuestra heurística, la búsqueda estará más dirigida al objetivo, con muy pocas ramificaciones en

[2]Un árbol uniforme es aquél en el cual todos los nodos expandidos (no hojas) tienen el mismo factor de ramificación y la profundidad de todos los caminos de búsqueda es constante.

el árbol. Normalmente, para una heurística h, $b*$ es mas o menos constante en varias instancias del problema.
El factor de ramificación efectivo b se calcula solucionando la incógnita b en la ecuación siguiente (donde N es el número de nodos visitados y d la profundidad de la solución), asumiendo un árbol uniforme:

$$N = 1 + b + b^2 + ... + b^d = \frac{(b^{d+1} - 1)}{(b - 1)} \quad (1)$$

En la Facultad Virtual se encuentra el código que resuelve los cálculos.

5. **Discusión**

   a) Conclusiones sobre los resultados obtenidos en la comparativa entre heurísticas del apartado $4b$.

   b) Comentar el origen de las ventajas e inconvenientes que aportan para este problema los métodos de búsqueda propuestos (no se espera en este apartado una copia de las comparativas generales entre estos métodos que se pueden encontrar en la bibliografía). Se incluirá una discusión sobre los resultados obtenidos en el apartado $4c$.

   c) Inventa una función heurística para el problema que en ocasiones sobreestime y muestra, utilizando el algoritmo A*, cómo produce una solución subóptima en un ejemplo en particular.

   d) Supongamos una nueva situación en la que se elimina una de las restricciones de forma que se pueda restar una unidad al 0 para dar lugar al 9, y sumar una unidad al 9 para dar lugar al 0. En ese caso, responde:

      1) ¿Cómo afectaría este nuevo planteamiento a la función de coste en el algoritmo A*?
      2) ¿Y a la heurística? Discute si las heurísticas diseñadas siguen siendo válidas, y si sería posible diseñar una nueva heurística que refleje mejor esta nueva situación.

   Y si además consideramos la situación de que la operación suma es dos veces más costosa que la operación resta. En ese caso, responde:

      1) ¿Cómo afectaría este nuevo planteamiento a la función de coste en el algoritmo A*?
      2) ¿Y a la heurística? Discute si las heurísticas diseñadas siguen siendo válidas, y si sería posible diseñar una nueva heurística que refleje mejor esta nueva situación
      3) En general, ¿cuándo tenderá a realizar operaciones $SUMA$ frente a operaciones $RESTA$ el algoritmo A*?

6. **Bibliografía**

7. **Apéndices** Código de las principales unidades de la práctica (algoritmos, generación de sucesores, comprobación de repetidos, etc.) y cualquier otra cosa de interés.

5. Entrega de la práctica

La práctica tendrá dos plazos de entrega:

- El plazo de entrega de la primera parte de la práctica será hasta el **16 de Enero de 2009**, y será improrrogable.

- El plazo de entrega de la segunda parte de la práctica será hasta el **13 de Marzo de 2009**, y será improrrogable.

La entrega de las dos partes de la memoria y el **código** (junto con un fichero *makefile* o, en su defecto, instrucciones claras de compilación y ejecución exclusivamente "en línea") se efectuará **electrónicamente** depositando los ficheros en los servidores `xurxo` o `limia`, para su recogida automática, bajo el directorio `/PRACTICAS/EI/IA/P1/login_alumno`.

6. Evaluación y Defensa de la práctica

Para poder aprobar la práctica deberá estar completa y cubrir todos los apartados propuestos. Los profesores de prácticas citarán a la defensa de esta primera práctica a los autores de los trabajos incompletos (apartados de la memoria, falta de instrucciones de compilación, fallo de ejecución, etc.) y a todos aquellos que tengan que realizar alguna aclaración a su práctica.

No se valorarán elementos ajenos al tema de la práctica tales como gráficos, música, etc.

7. NOTAS IMPORTANTES

- **Se recuerda que las prácticas DEBERÁN realizarse por parejas.**
- No existen horas de docencia presencial en laboratorio. El profesorado atenderá las dudas en el despacho, a través de la Facultad Virtual o por correo electrónico.
- Profesorado:
  - Mariano Cabrero Canosa. Tutorías en http://www.fic.udc.es email: cicanosa@udc.es
  - Juan Monroy Camafreita. Tutorías en http://www.fic.udc.es email: jmonroy@udc.es

Referencias

[1] V. Moret, A. Alonso, M. Cabrero, B. Guijarro, E. Mosqueira. *Fundamentos de Inteligencia Artificial (2^a Ed)*. Servicio de Publicaciones, UDC, 2000

[2] N. Nillson. *Inteligencia artificial: una nueva síntesis*. McGraw-Hill, 2001

[3] Prieditis, R. Davis. Quantitatively relating abstractness to the accuracy of admisible heuristics, *Artificial Intelligence*, vol.74, pp. 165-175, 1995

[4] E. Rich, K. Knight. *Inteligencia Artificial (2^a ed)*. McGraw-Hill, 1994

[5] S. Russell, P. Norvig. *Inteligencia Artificial: Un enfoque moderno*. Prentice-Hall, 2004.