Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

Práctica 4: Planificación

Ramon Ruiz Dolz
4CO21-2017

1. Introducción

Esta práctica consiste en el uso de dos planificadores diferentes para la resolución de problemas de planificación. Disponemos del planificador lpg-td, un planificador que realiza una búsqueda local heurística mediante grafos de planificación como base para las estimaciones y que además no es determinista. Este planificador por lo tanto en cada ejecución que realicemos nos dará una solución diferente. Por otra parte tenemos el mips-xxl, este planificador, obtiene una solución determinista mediante el uso de grafos de planificación relajados.

2. Comparación de planificadores

En esta parte vamos a comparar estos dos planificadores en tres dominios diferentes, el dominio ROVER, que consiste en la planificación del robot de Marte, el dominio STORAGE consistente en planificar el traslado de cajas desde contenedores a depósitos mediante el uso de grúas y finalmente el dominio PIPES, con el objetivo de planificar el flujo de los derivados del petróleo a través de una red de tuberías.

Los datos obtenidos se reflejan en la siguiente tabla:
(en cada celda los 3 valores son, en orden, el nº de acciones, el tiempo de cómputo en segundos y la duración del plan. Para LPG, al ser no determinista, he realizado 3 ejecuciones y he elegido la mediana.)

	Dominio ROVER			
	1	2	3	4
LPG	64 0.028 80(75)	55 0.032 66	80 0.02 72	89 0.024 53
MIPS	10 0.008 76	8 0.012 57	13 0.036 95	8 0.024 70

	Dominio STORAGE			
	1	2	3	4
LPG	8 0.032 3	28 0.028 3	60 0.02 3	58 0.028 15
MIPS	3 0.008 5	3 0.008 5	3 0.004 5	8 0.008 12

	Dominio PIPES			
	1	2	3	4
LPG	128 0.028 12	128 2.4 154(62)	772 0.068 18(14)	772 0.076 22(20)
MIPS	5 0.004 10	14 0.032 28	9 0.08 18	13 0.144 26

Como podemos observar, en general mips-xxl encuentra soluciones mejores y en menos tiempo para estos problemas estudiados aunque, como se ha indicado entre paréntesis, con lpg también se pueden encontrar soluciones de menor duración.

3. Definición de dominio y el problema

El problema elegido para diseñar un dominio y sus problemas es el conocido problema del viajante de comercio. Este problema consiste en un comerciante que viaja entre ciudades vendiendo su mercancía y trata de pasar por todas recorriendo el camino más corto terminando en la misma ciudad de la que parte. Para el diseño del dominio de este problema se han definido dos tipos, el de ciudad y el de comerciante (merchant y city en el código), se han definido dos predicados at y visited para llevar el control de la posición del comerciante y las ciudades que ha visitado ya; y se han definido también dos funciones, distance y visit que, para el primer problema solo se utilizará visit ya que el grafo es no dirigido y, por lo tanto, las duración entre cualesquiera 2 ciudades será simétrica, pero en las modificaciones posteriores se utilizarán ambas, estas funciones nos permiten obtener los valores que tarda el comerciante en visitar una determinada ciudad o lo que tarda en desplazarse de una ciudad a otra. Finalmente se han definido dos acciones durativas, la de visitar ciudad y la de viajar, como he indicado anteriormente, la de viajar en esta primera instanciación del problema será igual para todas ciudades estableciendo su duración en 35, mientras que la de visitar dependerá de la función visit declarada anteriormente.

Por otra parte, el problema se ha definido con 4, 8 y 10 ciudades. En mi instanciación del problema, el comerciante, Shinosuke se dispone a recorrer Kobe, Osaka, Kyoto y Himeji (Tokyo, Yokohama, Nagoya y Odawara, en la versión de 8 y, Sendai y Sapporo en la de 10 ciudades) para vender sus productos, el tiempo de visita de las ciudades está declarado en el código problema00_Xc.pddl sea X el número de ciudades:

```
(= (visit Kobe) 15)
 (= (visit Kyoto) 120)
  (= (visit Osaka) 60)
 (= (visit Himeji) 10)
 (= (visit Tokyo) 200)
(= (visit Yokohama) 70)
 (= (visit Nagoya) 55)
(= (visit Odawara) 24)
 (= (visit Sendai) 35)
(= (visit Sapporo) 48)
```

Con todos estos parámetros establecidos, ya se puede lanzar el planificador, sus resultados los analizaremos en el punto 5 de esta memoria.

Código: dominio00.pddl y problema00_4c.pddl, problema00_8c.pddl y problema00_10c.pddl

4. Modificaciones

Al dominio y problema modelado en el apartado anterior se pide realizar dos modificaciones. La primera de ellas, consiste en que no se necesita recorrer todas las ciudades ni es necesario que el viajante regrese a la ciudad de origen. Esta modificación solamente implica cambios en el archivo de problema, el dominio se mantiene igual. Concretamente en la parte de goal donde debemos sustituir el and por un or y añadir todos los casos así como eliminar el requisito de acabar en la ciudad de origen.

```
(or
        (visited Kobe)
        (visited Kyoto)
        (visited Osaka)
        (visited Himeji)
        (not (visited Kobe))
        (not (visited Kyoto))
        (not (visited Osaka))
        (not (visited Himeji))
    )
```

Código: dominio01.pddl y problema01.pddl

Finalmente, la última modificación consiste en que ahora las distancias entre ciudades no son simétricas, es decir el grafo ahora es dirigido. Para esta modificación debemos realizar cambios tanto en el dominio como en el problema. En el dominio ahora se modifica el valor de duración de viaje, antes constante, por la función distance definida anteriormente. Por otra parte, en el problema ahora debemos definir las distancias, al igual que hacíamos con los tiempos de visita:

```
       (= (distance Kobe Osaka) 25)
       (= (distance Kobe Kyoto) 50)
       (= (distance Kobe Himeji) 30)
       (= (distance Osaka Kobe) 40)
       (= (distance Osaka Kyoto) 15)
       (= (distance Osaka Himeji) 60)
       (= (distance Kyoto Osaka) 20)
       (= (distance Kyoto Kobe) 45)
       (= (distance Kyoto Himeji) 120)
       (= (distance Himeji Osaka) 49)
       (= (distance Himeji Kobe) 33)
       (= (distance Himeji Kyoto) 100)
```

```
:duration (= ?duration(distance ?c1 ?c2))
```

de esta manera, ahora las distancias serán diferentes para cada par de ciudades elevando así la complejidad del problema.

Código: dominio02.pddl y problema02.pddl

5. Obtención y evaluación de resultados

- Problema base:

Para el problema base del viajante de comercio he diseñado 3 problemas diferentes, uno con 4 ciudades y otros dos con 8 y 10 ciudades. Como podemos observar, con la ejecución en mips-xxl activando la optimización nos encuentra soluciones a los 3 problemas prácticamente al instante. Podemos observar que los planes óptimos, por cada ciudad añadida implica en un aumento de 2 acciones, teniendo 8 acciones el problema de 4 ciudades, 16 acciones el problema de 8 y finalmente 20 acciones el problema con 10 ciudades.

```
Plan-quality: 8.00
ff: found legal plan as follows

0.00: (VIAJAR SHINOSUKE KYOTO KOBE ) [35.00]
35.01: (VISITAR SHINOSUKE KOBE ) [15.00]
50.02: (VIAJAR SHINOSUKE KOBE OSAKA ) [35.00]
85.03: (VISITAR SHINOSUKE OSAKA ) [60.00]
145.04: (VIAJAR SHINOSUKE OSAKA HIMEJI ) [35.00]
180.05: (VISITAR SHINOSUKE HIMEJI ) [10.00]
190.06: (VIAJAR SHINOSUKE HIMEJI KYOTO ) [35.00]
225.07: (VISITAR SHINOSUKE KYOTO ) [120.00]

  345.00
```

4 ciudades.

```
Plan-quality: 16.00
ff: found legal plan as follows

0.00: (VIAJAR SHINOSUKE KYOTO KOBE ) [35.00]
35.01: (VISITAR SHINOSUKE KOBE ) [15.00]
50.02: (VIAJAR SHINOSUKE KOBE OSAKA ) [35.00]
85.03: (VISITAR SHINOSUKE OSAKA ) [60.00]
145.04: (VIAJAR SHINOSUKE OSAKA HIMEJI ) [35.00]
180.05: (VISITAR SHINOSUKE HIMEJI ) [10.00]
190.06: (VIAJAR SHINOSUKE HIMEJI TOKYO ) [35.00]
225.07: (VISITAR SHINOSUKE TOKYO ) [200.00]
425.08: (VIAJAR SHINOSUKE TOKYO YOKOHAMA ) [35.00]
460.09: (VISITAR SHINOSUKE YOKOHAMA ) [70.00]
530.10: (VIAJAR SHINOSUKE YOKOHAMA NAGOYA ) [35.00]
565.11: (VISITAR SHINOSUKE NAGOYA ) [55.00]
620.12: (VIAJAR SHINOSUKE NAGOYA ODAWARA ) [35.00]
655.13: (VISITAR SHINOSUKE ODAWARA ) [24.00]
679.14: (VIAJAR SHINOSUKE ODAWARA KYOTO ) [35.00]
714.15: (VISITAR SHINOSUKE KYOTO ) [120.00]

  834.00
```

8 ciudades.

```
Plan-quality: 20.00
ff: found legal plan as follows

0.00: (VIAJAR SHINOSUKE KYOTO KOBE ) [35.00]
35.01: (VISITAR SHINOSUKE KOBE ) [15.00]
50.02: (VIAJAR SHINOSUKE KOBE OSAKA ) [35.00]
85.03: (VISITAR SHINOSUKE OSAKA ) [60.00]
145.04: (VIAJAR SHINOSUKE OSAKA HIMEJI ) [35.00]
180.05: (VISITAR SHINOSUKE HIMEJI ) [10.00]
190.06: (VIAJAR SHINOSUKE HIMEJI TOKYO ) [35.00]
225.07: (VISITAR SHINOSUKE TOKYO ) [200.00]
425.08: (VIAJAR SHINOSUKE TOKYO YOKOHAMA ) [35.00]
460.09: (VISITAR SHINOSUKE YOKOHAMA ) [70.00]
530.10: (VIAJAR SHINOSUKE YOKOHAMA NAGOYA ) [35.00]
565.11: (VISITAR SHINOSUKE NAGOYA ) [55.00]
620.12: (VIAJAR SHINOSUKE NAGOYA ODAWARA ) [35.00]
655.13: (VISITAR SHINOSUKE ODAWARA ) [24.00]
679.14: (VIAJAR SHINOSUKE ODAWARA SENDAI ) [35.00]
714.15: (VISITAR SHINOSUKE SENDAI ) [35.00]
749.16: (VIAJAR SHINOSUKE SENDAI SAPPORO ) [35.00]
784.17: (VISITAR SHINOSUKE SAPPORO ) [48.00]
832.18: (VIAJAR SHINOSUKE SAPPORO KYOTO ) [35.00]
867.19: (VISITAR SHINOSUKE KYOTO ) [120.00]

  987.00
```

10 ciudades.

Por otra parte, si lanzamos la ejecución con lpg-td se pueden obtener diversas soluciones, sin embargo ninguna es mejor que las mostradas arriba.

- Modificación 1:

Las modificaciones explicadas en el apartado anterior sobre la instanciación del problema derivan en la solución trivial, puesto que es un problema de minimización de tiempo y no estoy exigiendo que visite nada ni que acabe en un sitio concreto, lo óptimo para esta instanciación del problema es quedarse quieto y no hacer nada.

```
Plan-quality: 0.00


ff: found legal plan as follows


  0.00
```

- Modificación 2:

Finalmente, en la última modificación se nos pide modificar las distancias de manera que el grafo sea dirigido, es decir que las distancias entre ciudades no sean simétricas. Para ello entra en uso la función que declaramos anteriormente como distance que nos servirá para obtener la distancia entre 2 ciudades diferentes, además de añadir igual que hacíamos con visited, los valores de las distancias en la instanciación del problema. Esto aumentará considerablemente la complejidad del problema puesto que hay mas casos que valorar para poder obtener el óptimo con mips-xxl.

```
Plan-quality: 8.00


ff: found legal plan as follows

0.00: (VIAJAR SHINOSUKE KYOTO KOBE ) [45.00]
45.01: (VISITAR SHINOSUKE KOBE ) [15.00]
60.02: (VIAJAR SHINOSUKE KOBE OSAKA ) [25.00]
85.03: (VISITAR SHINOSUKE OSAKA ) [60.00]
145.04: (VIAJAR SHINOSUKE OSAKA HIMEJI ) [60.00]
205.05: (VISITAR SHINOSUKE HIMEJI ) [10.00]
215.06: (VIAJAR SHINOSUKE HIMEJI KYOTO ) [100.00]
315.07: (VISITAR SHINOSUKE KYOTO ) [120.00]

  435.00
```

6. Conclusiones

Esta práctica me ha permitido entablar contacto por primera vez con los planificadores, tras lanzar las pruebas con los 3 primeros dominios he podido observar la relevancia del determinismo en un planificador así como las ventajas que estos tienen no encontrando la solución óptima (siempre) pero alcanzando buenas soluciones en tiempos más que aceptables. Finalmente he diseñado mi propio dominio y esto me ha servido para entender completamente el funcionamiento de los dominios y los problemas.