Análisis y resolución del problema de la mochila y el TSP

Héctor Abraham Galván García

***Abstract*—En el presente reporte se examina el procedimiento para resolver el problema del mochilero y el TSP mediante un algoritmo dinámico adjuntando los resultados obtenidos en un programa de Java.**

Palabras clave: Programación dinámica, peso, beneficio, mochilero, grafo ponderado no dirigido, circuito hamiltoniano

I. Introducción

Cuando se programa recursivamente soluciones en términos matemáticos, las soluciones normalmente son de orden exponencial, tornándose inviables. Se puede resolver este problema mediante la programación dinámica.

Basado en el principio de Divide y Vencerás, la programación dinámica caracteriza la solución de un problema en términos de las soluciones de subproblemas del mismo. Como ya se mencionó, si se combina con la recursividad proporciona métodos eficientes de solución para problemas. Para que un problema pueda ser abordado por esta técnica debe cumplir dos condiciones:
-La solución al problema ha de ser alcanzada a través de una secuencia de decisiones, una en cada etapa.
-Dicha secuencia de decisiones ha de cumplir el principio de óptimo ("En una secuencia de decisiones óptima toda subsecuencia ha de ser también óptima").

A grandes rasgos pues, el diseño de un algorimo dinámico consta de los siguientes pasos:
1.- Planteamiento de la solución como una sucesión de decisiones que cumplan el principio de óptimo.
2.- Definir de forma recursiva la solución.
3.-Calculo del valor de la solución almacenando soluciones a problemas parciales.
4.- Construir la solución óptima usando los datos del paso anterior.

II. Problema de la mochila

El problema de la mochila es un problema de optimización. Dados el peso y el beneficio de n artículos, el algoritmo encuentra la combinación que maximice el beneficio minimizando el peso.

Para la programación se trabajó una clase Item con los atributos peso y beneficio que sirve para la clase Mochila que es la que genera los items aleatoriamente en un rango establecido (número de items, peso, beneficio). Posteriormente, con el método buscar solución, se calcula una matriz de beneficios usando los valores de los items generados. Por último, se muestra en la salida la combinación de artículos que maximiza la salida con el mínimo de pesos, en el formato PESO/BENEFICIO.

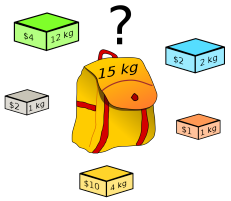

Fig. 1. Ejemplo del planteamiento del problema

III. Problema del TSP

El problema del vendedor viajante (Travelling Salesman Problem por sus siglas en inglés), constituye el problema NP-Hard más estudiado en la optimización combinatoria. Aunque siempre se busca siempre su optimización, se conocen bastantes heurísticas y métodos de solución para el problema. El problema consiste en, dadas n ciudades y una distancia par entre cada una de ellas, encontrar el camino más corto posible visitando todas las ciudades. En teoría de grafos, el problema se puede ver como un grafo ponderado no dirigido, y la solución como el ciclo hamiltoniano más corto posible.

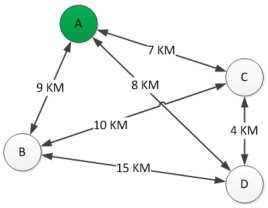

Fig. 2. Ejemplo del planteamiento del problema

Se utilizó con la siguiente matriz de distancias, la cual representa una ciudad.

0	13	33	28	37	7	32	40	80	26
13	0	39	83	50	68	16	98	81	55
33	39	0	80	88	49	53	75	63	55
28	83	80	0	94	4	20	6	59	76
37	50	88	94	0	81	87	85	4	19
7	68	49	4	81	0	96	53	40	37
32	16	53	20	87	96	0	80	57	68
40	98	75	6	85	53	80	0	65	41
80	81	63	59	4	40	57	65	0	97
26	55	55	76	19	37	68	41	97	0

Fig. 3. Matriz de distancias

Para la programación se trabajó en una sola clase. En el constructor se asegura de que el problema no exceda la capacidad del procesador. Un método solucionar encuentra el camino mínimo posible usando atributos de clase que es el peso del camino y el camino (que se maneja como un ArrayList).

IV. Pruebas de la mochila

-

```
22,77.0
8,89.0
8,24.0
4,42.0
13,87.0
8,59.0
12,36.0
12,43.0
6,80.0
```

Fig. 4. 30 items, 100 de peso máximo y 100 en los rangos de peso y beneficio

```
43,99.0
56,72.0
18,61.0
23,109.0
27,69.0
21,118.0
```

Fig. 5. 30 items, 200 de peso máximo y 150 en los rangos de peso y beneficio

V. Pruebas del TSP

```
Camino: [0, 5, 8, 4, 9, 7, 3, 6, 1, 2, 0]
Costo del camino: 225.0
BUILD SUCCESSFUL (total time: 1 second)
```

Fig. 6. Camino y costo del camino

VI. Conclusiones

El análisis de los algoritmos dinámicos durante la realización de la práctica ayudó a comprender la teoría detrás de ella, puesto que no es fácil implementar una solución dinámica, y menos para problemas NP-difíciles. Ahora bien, aunque el algoritmo no sea el más eficiente que existe, que funcione y que siga los principios de la programación dinámica muestra el éxito de la práctica.