Universidad de Santiago de Chile
Departamento Ingeniería Informática
Ingeniería Civil Informática
Algoritmos Avanzados
Laboratorio

1-2020

Resolución del problema de las n-mochilas utilizando Backtracking

Esteban Andrés López Garrido
e-mail: esteban.lopez.g@usach.cl
github.com/ealopezg/optimalTransport

RESUMEN: *Demostración de una solución al problema de las n-mochilas utilizando backtracking, dando por resultado que el algoritmo no es eficiente y que es recomendable no utilizarlo si el tiempo de respuesta es importante.*

1 INTRODUCCIÓN

Este documento tiene la finalidad de mostrar la solución utilizada para el problema de las n-mochilas utilizando Backtracking.

Primero se verá una pequeña explicación del método utilizado, y finalmente un análisis de la solución obtenida.

Es importante recalcar que en el enunciado hablaba de Transportes pero para generalizar se hablará de **Mochilas**.

2 DESCRIPCIÓN DEL MÉTODO

Primero es importante detallar cómo se estructuró la solución.
k = número de paquetes
n = número de mochilas/transportes
Estado: Nodo de un arbol que posee un arreglo de k enteros, donde cada valor está entre -2 y n-1.
Ej:
Para 3 paquetes y 2 mochilas:

-1	0	1

Se tiene que el paquete 0 no está en ninguna mochila, el paquete 1 está en la primera y el paquete 2 está en la segunda mochila.

Si el valor es -1 significa que no será guardado, si el valor es -2 significa que no se ha decidido que hacer con ese paquete.

El nodo inicial está dado por:

-2	-2	-2

El método se basa en ir creando un nodo por cada solución y ver evaluando si es válida o no.
Una solución es válida cuando la cantidad de pesos de los paquetes no sobrepasan el peso máximo admitido por la mochila.

3 DESCRIPCIÓN DE LA SOLUCIÓN

La solución consta de la siguiente manera:

1. *Lectura de archivos de entrada.*
2. *Generar el primer nodo*
3. *Mientras no esté completado:*
    a. *Revisar si ha llegado al nodo inicial después de haber empezado (Caso final)*
    b. *Calcular beneficio de la solución*
    c. *Actualizar valores de la mejor solución si es necesario*
    d. *Si no ha llegado a una solución válida o no se puede crear un nodo hijo:*
        i. *Crea un nodo hermano*
        ii. *Sino crea un nodo hijo*

iii. Repetir con la nueva solución

4. *Devolver la mejor solución*
5. *Guardado de archivo de salida*

3.1 ORDEN DE COMPLEJIDAD

La complejidad de la solución está dada por la suma del algoritmo aplicado a todos los nodos, en el peor caso, es decir, cuando todas las soluciones son válidas y se crean todos los hijos necesarios, está dado por:

$$T(n,k) = \sum_{i=0}^{k} (n+1)^i (c + 2n + k)$$

Resolviendo la sumatoria se tiene:

$$T(n,k) = \frac{\left(n(n+1)^k + (n+1)^k - 1\right)(c + k + 2n)}{n}$$

Luego, en el caso de n=k:

$$T(n) = 3\left(n(n+1)^n + (n+1)^n - 1\right)$$

$$O(n(n+1)^n)$$

4 ANÁLISIS DE LA SOLUCIÓN

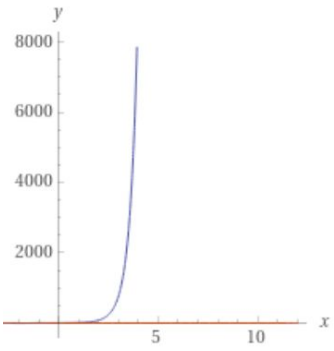

Figura 2: T(n) graficado

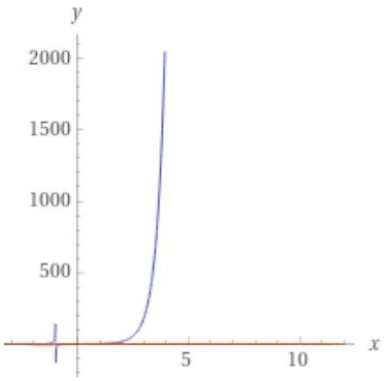

Figura 2: O(n) graficado.

Como se puede apreciar, el algoritmo no es eficiente aún cuando las entradas son pequeñas.
Para 2 mochilas y 2 paquetes: T(2) = 78
Para 4 mochilas y 4 paquetes: T(4) = 9372
Para 5 mochilas y 5 paquetes: T(5) = 139965

4.1 ANÁLISIS DE IMPLEMENTACIÓN

Es un algoritmo ya que tiene una secuencia de pasos para resolver un problema, y termina con la solución óptima al problema. Tiene una complejidad de O(n(n+1)^n) lo que lo hace un algoritmo poco eficiente. Se podría mejorar el algoritmo restringiendo el límite del árbol, eliminando paquetes que no entren en ninguna mochila o proponiendo otro método que lo pudiera resolver de manera más eficiente.

4.2 EJECUCIÓN

Para ejecutar el programa, escribir en la consola:

```
$ make
```

Y luego para ejecutarlo

```
$ ./optimal.out entrada.in salida.out
```

o

```
> optimal.exe entrada.in salida.out
```

Para compilar el modo debug
`$ make`

Y luego para ejecutarlo

```
$./optimal_debug.out entrada.in
salida.out
```

o

```
>optimal_debug.exe entrada.in
salida.out
```

5 CONCLUSIONES

El algoritmo presentado en este documento soluciona el problema de las n-mochilas, sin embargo, cuando las entradas son “medianas” el costo en tiempo es altísimo. Al ser Backtracking es necesario ir recorriendo solución por solución hasta buscar todas las soluciones posibles.

Para este tipo de problemas, si no es importante obtener la solución óptima es conveniente utilizar otro método como Goloso, ya que no será necesario recorrer todas las soluciones posibles.

Una posible mejora a este algoritmo es reducir si es que se puede la cantidad de paquetes a utilizar, eliminando los que no pueden entrar en ninguna mochila.