Universidad de Buenos Aires

Facultad de Ingeniería

75.29 Teoría de Algoritmos

Trabajo Práctico 2

Integrantes

- Arjovsky, Tomás
- Gavrilov, Seva
- Pereira, Fernando
- Pérez Dittler, Ezequiel

Segundo cuatrimestre de 2016

Contents

1 Programación dinámica

1.1 El problema de la mochila

En este problema, tenemos una cantidad n de items, cada uno de los cuales posee un peso w no negativo y un valor v. Con estos items, nosotros vamos a ir llenando una mochila, la cual posee una capacidad máxima determinada. El problema plantea encontrar los items que se incluirán en la mochila, de tal forma que la suma de todos sus pesos particulares w no supere la capacidad máxima de la mochila y, además, la suma de todos los valores v de los objetos que se incluyen, sea máxima.

Contrario a lo que uno puede intuir, no existe un algoritmo greedy eficiente que lo resuelva, por lo que caemos en la programación dinámica como una nueva técnica para encontrar soluciones óptimas a determinados problemas, como el de la mochila, partiendo el mismo en sub-problemas cada vez más pequeños (los cuales se resolverán mucho más fácilmente), y solapando las soluciones a dichos sub-problemas para llegar a la solución del problema original. Es decir, debemos procurar resolver sub-problemas cada vez más sencillos, los cuales, en conjunto, forman la solución al problema mayor.

En este informe se verá un pequeño análisis general del orden de complejidad de la solución encontrada, como así también las diferencias entre los tiempos de ejecución de dos enfoques distintos para implementar la solución (Bottom-up y Top-Down)

1.1.1 Solución y orden de complejidad

1.2 El problema del viajante de comercio

El algoritmo Bellman-Held–Karp fue propuesto en 1962 independientemente por Bellman (Bellman 1962) y por Held y Karp (Held and Karp 1962).

En la formulación de Bellman, se define la función $D(v, S)$ la distancia mínima desde v hasta la ciudad de origen, S el conjunto de ciudades a visitar. Si el conjunto S se encuentra vacío, $D(v, S) = d_{v0}$. Se define d_{ij} como la distancia desde la ciudad i hasta la ciudad j. Para el resto de los casos, $D(v, S) = \min_{u \in S}(d_{vu} + D(u, S - \{u\}))$

Un pseudocódigo para calcular la distancia del ciclo hamiltoniano mínimo es el siguiente:

```
function TSP (M, n)
  for k := 2 to n do
    C({1, k}, k) := M[1,k]
  end for

  for s := 3 to n do
    for all S in {1, 2, . . . , n}, |S| = s do
      for all k in S do
        C(S, k) = min [C(S - {k}, m) + M[m,k] ]
      end for
    end for
  end for

  opt := min[C({1, 2, 3, . . . , n}, k) + M[k,1]]
  return (opt)
end
```

Sin embargo, como se debe construir el camino mínimo, para el costo hacia cada conjunto se guarda el padre desde el cual se llega.

```
function TSP (M, n)
  for k := 2 to n do
    C({1, k}, k) := (M[1,k], 1)
  end for

  for s := 3 to n do
    for all S in {1, 2, . . . , n}, |S| = s do
      for all k in S do
        C(S, k) = min[(C(S - {k}, m) + M[m,k], m)]
      end for
    end for
  end for

  // TODO: Código para obtener el camino
end
```

2 Flujo de redes

Referencias

Bellman, Richard. 1962. "Dynamic Programming Treatment of the Travelling Salesman Problem." *Journal of Association for Computing Machinery.*

Held, Michael, and Richard M. Karp. 1962. "A Dynamic Programming Approach to Sequencing Problems." *Journal for the Society for Industrial and Applied Mathematics.*