Manuel Arturo Chinchilla Sánchez - 2013009344

Resumen #11

Chapter 5.1: The Greedy Method

The Greedy Method.

El método “Goloso” se aplica a la optimización de problemas, es decir, problemas que involucran búsquedas a través de un conjunto de configuraciones para encontrar una que minimice o maximice una función objetivo definida en esas configuraciones. La fórmula general del método Goloso no podía ser más simple. Con el fin de resolver un problema de optimización dado, se procede por una secuencia de opciones. La secuencia se inicia desde alguna configuración inicial bien entendida y luego iterativamente toma la decisión que parece la mejor de todas en ese momento.

Este enfoque Goloso no siempre conduce a la solución más óptima (a largo plazo). Pero hay varios problemas para los que funciona de manera óptima, y este tipo de problemas se dice que poseen la propiedad greedy-choise. Esta es la propiedad que consiste en que una configuración global puede ser alcanzada por una serie de decisiones localmente óptimas (es decir, decisiones que fueron las mejores entre las posibilidades disponibles en el momento), a partir de una configuración bien definida.

Problema Fraccional de la Mochila.

Considere el problema donde se nos da un conjunto S de n ítems, de manera que cada elemento i tiene un beneficio positivo bi y un peso positivo wi y queremos encontrar el subconjunto de máximo-beneficio que no exceda de un peso dado W. Si nos limitamos a aceptar o rechazar por completo cada ítem, entonces tendríamos la versión 0-1 de este problema. Ahora nos permitimos tomar fracciones arbitrarias de algunos elementos. La motivación de este problema es que vamos de un viaje y tenemos una sola mochila que puede llevar elementos que en conjunto tienen un peso W como máximo. Además, se nos permite tomar una cantidad xi de cada elemento i tal que:

$$0 \leq Xi \leq Wi \; para \; cada \; i \; \in S \;\; y \;\; \sum_{i \in S} Xi \; \leq \; W$$

El beneficio total de los elementos tomado está determinado por la función objetivo

$$\sum_{i \in S} bi(^{Xi}/_{Wi})$$

Programación de Tareas.

Supongamos que se nos da un conjunto *T* de n tareas, de manera que cada tarea *i* tiene una hora de inicio *si*, y un tiempo de finalización *fi* (donde *si* <*fi*). La tarea debe empezar en el tiempo *si* y se garantiza que sea terminado por el tiempo *fi*. Cada tarea tiene que ser realizada en una máquina y cada máquina puede ejecutar una sola tarea a la vez. Dos tareas *i* y *j* no son conflictivas si *fi* ≤ *sj* ó *fj* ≤ *si*. Dos tareas se pueden programar para ser ejecutadas en la misma máquina sólo si no son confictivas. El problema de programación de tareas que consideramos aquí es programar todas las tareas en T en el menor número de máquinas posible de una manera no conflictiva. Alternativamente, podemos pensar en las tareas como las reuniones que tenemos que programar en el menor número de salas de conferencia como sea posible.