Algoritmos golosos y heurísticas

Tecnología Digital V: Diseño de Algoritmos
Universidad Torcuato Di Tella

UNIVERSIDAD
TORCUATO DI TELLA

Heurísticas

Definición

Una heurística es un procedimiento computacional que intenta obtener soluciones de buena calidad para un problema, intentando que su comportamiento sea lo más preciso posible.

- Una heurística para un problema de decisión puede obtener una solución que sea correcta la mayor parte de las veces.
- Una heurística para un problema de optimización obtiene una solución con un valor que se espera sea cercano (idealmente igual) al valor óptimo.

Definición

Decimos que A es un algoritmo ϵ-aproximado ($\epsilon > 0$) para un problema si

$$\left| \frac{f_A - f_{OPT}}{f_{OPT}} \right| \leq \epsilon.$$

Algoritmos golosos

Construir una solución seleccionando en cada paso la mejor alternativa, sin considerar (o haciéndolo débilmente) las implicancias de esta selección.

- Habitualmente, proporcionan heurísticas sencillas para problemas de optimización.
- En general permiten construir soluciones razonables (pero sub-óptimas) en tiempos eficientes.
- Sin embargo, en ocasiones nos pueden dar interesantes sorpresas!

El problema de la mochila

Recordemos el problema de la mochila.

Problema de la mochila

Datos de entrada:

- Capacidad $C \in \mathbb{Z}_+$ de la mochila (peso máximo).
- Cantidad $n \in \mathbb{N}$ de objetos.
- Peso $p_i \in \mathbb{Z}_{>0}$ del objeto i, para $i = 1, \ldots, n$.
- Beneficio $b_i \in \mathbb{Z}_{>0}$ del objeto i, para $i = 1, \ldots, n$.

Problema: Determinar qué objetos debemos incluir en la mochila sin excedernos del peso máximo C, de modo tal de maximizar el beneficio total entre los objetos seleccionados.

El problema de la mochila

- **Algoritmo(s) goloso(s):** Mientras no se haya excedido el peso de la mochila, agregar a la mochila el objeto i que ...
  - ... tenga mayor beneficio b_i.
  - ... tenga menor peso p_i.
  - ... maximice b_i/p_i.
- Qué podemos decir en cuanto a la calidad de las soluciones obtenidas por estos algoritmos?
- Qué podemos decir en cuanto a su complejidad?
- Qué sucede si se puede poner una fracción de cada elemento en la mochila?

El problema de la mochila

- Supongamos que los objetos están ordenados de mayor a menor cociente b_i/p_i.

$L \leftarrow C$;
$i \leftarrow 0$;
while $L > 0$ **do**
    $x \leftarrow \min\{1, L/p_i\}$;
    Agregar una fracción de x del objeto i a la solución;
    $L \leftarrow L - x\ p_i$;
    $i \leftarrow i + 1$;
end while

Teorema

El algoritmo goloso por cocientes encuentra una solución óptima del problema de la mochila fraccionario.

Tiempo de espera total en un sistema

Problema

Un servidor tiene n clientes para atender, y los puede atender en cualquier orden. Para $i = 1, \ldots, n$, el tiempo necesario para atender al cliente i es $t_i \in \mathbb{R}_+$. El objetivo es determinar en qué orden se deben atender los clientes para minimizar la suma de los tiempos de espera de los clientes.

- Si $I = (i_1, i_2, \ldots, i_n)$ es una permutación de los clientes que representa el orden de atención, entonces la suma de los tiempos de espera es

$$\begin{aligned} T &= t_{i_1} + (t_{i_1} + t_{i_2}) + (t_{i_1} + t_{i_2} + t_{i_3}) + \ldots \\ &= \sum_{k=1}^{n} (n-k) t_{i_k}. \end{aligned}$$

Tiempo de espera total en un sistema

Algoritmo goloso

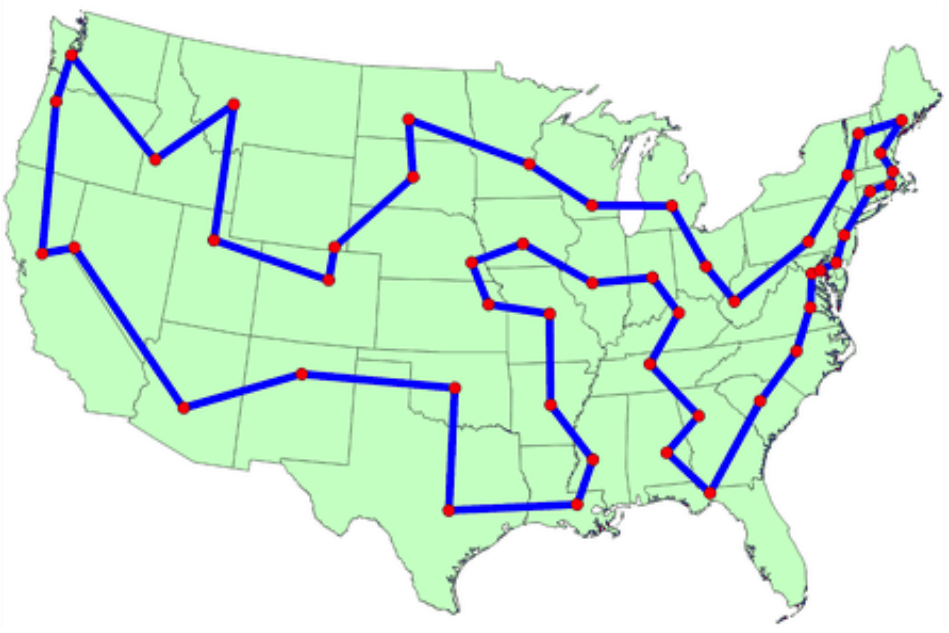
En cada paso, atender al cliente pendiente que tenga menor tiempo de atención.

- Este algoritmo retorna una permutación $I_{\text{GOL}} = (i_1, \ldots, i_n)$ tal que $t_{i_j} \leq t_{i_{j+1}}$ para $j = 1, \ldots, n-1$.
- Cuál es la complejidad de este algoritmo?

Teorema

El algoritmo goloso por menor tiempo de atención proporciona una solución óptima del problema de minimizar el tiempo total de espera en un sistema.

Problema del viajante de comercio (TSP)



Problema del viajante de comercio

Dado un grafo completo $G = (V, X)$ con longitudes asignadas a las aristas, $l : X \to \mathbb{R}_+$, encontrar un circuito hamiltoniano en G de longitud mínima.

Heurística golosa: vecino más cercano

elegir un nodo v
$orden(v) := 0$
$S := \{v\}$
$i := 0$
mientras $S \neq V$ **hacer**
    $i := i + 1$
    elegir la arista (v, w) más barata con $w \notin S$
    $orden(w) := i$
    $S := S \cup \{w\}$
    $v := w$
fin mientras
retornar $orden$

- Cuál es la complejidad computacional de este algoritmo?
- Qué se puede decir sobre la calidad de las soluciones que encuentra?

Heurísticas de inserción

$C :=$ un circuito de longitud 3
$S :=$ {nodos de C}
mientras $S \neq V$ **hacer**
    ELEGIR un nodo $v \notin S$
    $S := S \cup \{v\}$
    INSERTAR v en C
fin mientras
retornar C

- Cómo ELEGIR?
- Cómo INSERTAR?
- Cuál es la complejidad computacional de este algoritmo?
- Qué se puede decir sobre la calidad de las soluciones que encuentra?