Tema 7: Filosofía Greedy

Características

La idea es seleccionar en cada momento lo mejor entre un conjunto de candidatos, sin tener en cuenta lo ya hecho. Para aplicar esta técnica, se deben cumplir las siguientes características:

- Un conjunto de **candidatos**
- Una lista de candidatos **ya usados**
- Una **solución** (función) que dice cuándo un conjunto de datos es solución (criterio de parada)
- Un criterio que dice cuándo un conjunto de candidatos podrá llegar a ser una solución (**condiciones de factibilidad**)
- Una función **selección** que indica en cada instante cuál es el candidato más prometedor de los no usados
- Una función **objetivo** que a cada solución le asocia un valor

Algoritmo

1. Se parte de un conjunto de candidatos a solución vacío
2. De la lista de candidatos, con la función de selección, se coge al mejor candidato
3. Vemos si con ese elemento se puede conseguir una solución con las condiciones de factibilidad
4. Si el candidato no sirve, lo borramos de la lista de candidatos posibles
5. Evaluamos la función objetivo
6. Seleccionamos otro candidato (con la función selección) y repetimos el proceso

La primera solución que se consigue suele ser la óptima. Sin embargo, no hay garantía de que proporcione solución óptima y siempre habrá que estudiar la **corrección del algoritmo** para verificar las soluciones.
Si nos cuesta tanto encontrar un contraejemplo, suele pasar (no siempre) que suele ser óptima la solución

Esto es una ventaja en problemas en los que es casi imposible alcanzar una solución óptima:

- Problema del Coloreo de un Grafo
- Problema del Viajante de Comercio

Greedy y Optimalidad

Hay casos en los que un óptimo local podrá considerarse como global. Es decir, que en el paso de escoger un candidato que produzca el óptimo local en una etapa, se consigue el óptimo global del problema.
Por ejemplo, en conjuntos convexos (“Dados dos puntos del conjunto S, todo el segmento lineal que los une está en el conjunto”) se consigue la solución global a partir de una solución local.

Grafos

Un grafo se define con un **conjunto de vértices** y un conjunto de **aristas**.

Cuando las aristas tienen dirección, se habla de **grafos dirigidos**. Cuando a cada arista se le asigna un valor, se llama grafo **ponderado**.
En la asignatura, grafos no dirigidos.

Nociones básicas:

- Podemos unir dos vértices varias veces con múltiples aristas
- Podemos unir un vértice a si mismo (**lazo**)
- Si cualquier par de vértices está unido por una arista, hablamos de **grafo completo**
- Dos **vértices** son **adyacentes** si hay una arista que los une
- Dos **aristas** son **adyacentes** si comparten un vértice
- Si el grafo puede pintarse en el plano sin que se crucen sus aristas, hablamos de **grafo plano**.
- El **grado del vértice** es el número de aristas que pasan por el vértice
    - Si un vértice tiene un lazo, cuenta como 2 aristas
- Un **camino** es una sucesión de aristas distintas adyacentes que no se repiten
- Un **circuito** es un camino que comienza y termina en el mismo vértice (recordar no repetir aristas)
- Si cualesquiera dos vértices pueden unirse por un camino, **grafo conexo**. Si no, grafo inconexo
- Un **camino Euleriano** es un camino que pasa por todas las aristas solo una vez
- Un **circuito Euleriano** es un circuito que pasa por todas las aristas solo una vez
- Teorema de Euler
    - Si todos los vértices de un grafo son de grado impar, entonces no existen circuitos Eulerianos
    - Si un grafo es conexo y todos sus vértices son de grado par, existe al menos un circuito Euleriano
- Un **circuito Hamiltoniano** es un circuito que pasa por todos los vértices sólo una vez
- Un **árbol** es un grafo que no tiene ciclos
    - Un grafo conexo y sin circuitos es un árbol

Problema del Cartero Chino
Encontrar el circuito de longitud minimal que recorre cada arista de un grafo al menos una vez (determinación del circuito euleriano minimal)

Problema del Viajante de Comercio
Encontrar el circuito de longitud minimal que recorre todos los vértices de un grafo una sola vez (comienza y termina por el mismo vértice)

Árboles
Dado un grafo conexo con pesos en sus arcos, un **árbol generador minimal** es un árbol en el que la suma de los pesos de sus arcos es mínima.

Teorema de Cayley: Hay n^{n-2} árboles generadores de K_n(grafo completo de n vértices).

De aquí concluimos que un método de Fuerza Bruta para sacar el árbol generador minimal no es recomendable.

Cuando el grafo es ponderado: matriz de adyacencia y lista de adyacencia.

- En la **matriz** $n \times n$ se pone un 1 en las que conectan dos nodos unidos por un arista.
- En la primera fila de la **lista** aparecen el número de nodos y debajo de cada nodo, los nodos a los que se conecta con una arista. Si está dirigido, debajo de cada nodo, los nodos a los que apunta.

Cuando el grafo es no dirigido:

- Si hay una arista entre i y j, entonces $B_G(i, j) = 1$; si no, 0.

Cuando el grafo es dirigido:

- Si hay una arista que apunta desde i hasta j, entonces $B_G(i, j) = 1$, y $B_G(j, i) = -1$; en otro caso, 0.

Tema 8: Greedy sobre grafos

Problema del árbol generador minimal

Suponemos un grafo conexo $G = (V, A)$ sobre el que hay definida una matriz de pesos $L(i, j) \geq 0$. Queremos encontrar un árbol $T \subseteq A$ tal que todos los nodos permanezcan conectados cuando solo se usen aristas de T, siendo la suma de los pesos de sus aristas mínima.

El problema se resuelve con **Greedy**:

- Un conjunto de candidatos: la lista de aristas
- Una lista de candidatos ya usados
- Una solución será el conjunto de aristas que forme un AGM
- La condición de factibilidad es que la arista que se va a incluir no forme ningún ciclo
- Una función selección que escogerá la arista de menos peso
- Un función objetivo que es que la suma de los pesos de las aristas del árbol sea mínima

La **propiedad** de AGM:

- Sea $G = (V, A)$ un grafo no dirigido y conexo donde cada arista tiene una longitud conocida. Sea $U \subset V$ un subconjunto. Si (u, v) es una arista tal que $u \in U$ y $v \in V - U$ y, además, es la arista que verifica esto con menor peso, entonces existe un AGM que incluye a (u, v)
- Esta propiedad garantiza que el algoritmo greedy funcione correctamente

Demostración de la propiedad AGM:

- Por contradicción: Supongamos que T es un AGM que no contiene a (u, v).
  Si añadimos (u, v) a T se crea un ciclo. Este ciclo lo eliminamos quitando una arista (y, z). De esta forma, obtenemos un nuevo AGM que incluye a (u, v) y que tiene menor longitud que T, lo que es una contradicción.

Algoritmo de Kruskal

Resuelve el problema del árbol generador minimal con un grafo con V nodos. Los pasos a seguir son:

1. Ordenar las aristas de forma creciente de costo
2. Bucle hasta que tengamos (V-1) aristas en la solución:
    a. Coger la arista más corta (buscar los dos nodos más cercanos que sean distintos)
    b. Borrar la arista de la lista de candidatos
    c. Aceptar la arista si no provoca un ciclo. Rechazarla en caso contrario

Si nuestro gráfico tiene a aristas, entonces el tiempo del algoritmo es $O(a\ log(n)^2)$

Algoritmo de Prim

Es otro método para resolver el problema del AGM. La diferencia está en que en el algoritmo de Kruskal implica un crecimiento desordenado del AGM (porque empieza por la arista más corta de la lista) y el algoritmo de Prim propone un crecimiento ordenado (a partir de una raíz).

Los pasos son:

1. Tomar un conjunto de U nodos que contiene al nodo raíz
2. Crear el conjunto T de las soluciones (aristas)
3. Bucle hasta que $U = n$
    a. En cada etapa se busca la arista más corta que conecta U con $V - U$
    b. Añade el vértice obtenido al conjunto U y la arista obtenida a T
    c. En cada instante, las aristas que están en T constituyen un AGM para los nodos en U

El algoritmo de Prim requiere un tiempo $O(n^2)$

Luego, para grafos poco densos usamos Kruskal, para gráficos densos usamos Prim

Algoritmo de Dijkstra

El problema consiste en determinar el camino mínimo entre dos vértices de un grafo.

Suponemos un grafo dirigido $G = (V, E)$ con V el conjunto de vértices y E el conjunto de arcos. Se trata de hallar la distancia de los caminos mínimos desde un nodo raíz a todos los demás.

Los pasos son:

1. Crear S el conjunto de nodos cuya distancia desde el origen es mínima
2. Inicialmente S contiene al origen y, al final del algoritmo, S tendrá todos los nodos del grafo
3. Bucle:
    a. Elegir el nodo cuya distancia al nodo raíz sea menor
    b. Añadir el nodo a S
    c. Actualizar los valores $D[V]$ donde se almacenan las longitudes de cada nodo al nodo raíz
        i. Para hacerlo vemos si el nuevo vértice introducido en S consigue que un nodo v llegue al nodo raíz de forma más corta

Diremos que un camino del nodo raíz a otro es **especial** si todos los nodos intermedios están en S

Demostración: por inducción sobre el número de nodos seleccionados

- Base ($i = 1$), es decir, sólo hay 1 nodo seleccionado:
  - $D[a] = 0$ donde a es el nodo origen. La distancia a cualquier otro nodo que no sea él mismo, será mayor que 0
  - $T(1)$ es cierta.
- Supongamos que $\forall i \in S$ se cumple que $D[i]$ es la mínima distancia de un camino desde el origen hasta el nodo i
  - //no entiendo ni papa

El algoritmo consume un tiempo de $O(n^2)$
Importante: NO debe haber ninguna distancia negativa. Si la hay, el algoritmo no funciona.

Tema 9: Heurísticas Greedy

Definición
Son procedimientos que, basados en la experiencia, proporcionan buenas soluciones a problemas concretos. Es más importante que el tiempo en resolver el problema sea reducido a que se consiga la solución optimal.

Problema del Coloreo de un Grafo
Dado un grafo plano $G(V, E)$, determinar el número de colores que se necesitan para colorear todos sus vértices y que no haya dos de ellos adyacentes coloreados con el mismo color.
Como el problema es NP se necesitan heurísticas para resolverlo.
Pasos a seguir:

1. Elegimos un vértice no coloreado y un color y pintamos el vértice de ese color.
2. Bucle hasta pintar todos los vértices:
   a. Seleccionamos un vértice no coloreado v
   b. Si v no es adyacente a un vértice ya coloreado con el nuevo color, entonces coloreamos v con el nuevo color

Iniciar Coloreo (pone en NuevoColor los vértices de G que pueden tener el mismo color)
    Vector NuevoColor a nulo
    Para cada vértice no coloreado v de G
        Si v no es adyacente a ningún vértice en NuevoColor
            Marcar v como coloreado
            Añadir v a NuevoColor
End

No siempre da la función óptima, pero tiene un tiempo $O(n)$

Problema del Viajante de Comercio
Un viajante de comercio que reside en una ciudad tiene que trazar una ruta que, partiendo de su ciudad, visite todas las ciudades a las que tiene que ir una y sólo una vez, volviendo al origen y con un recorrido mínimo.
Es un problema NP

Suponemos un grafo no dirigido y completo $G = (N, A)$ y L una matriz de distancias no negativas. Se busca encontrar un circuito hamiltoniano minimal(que pase por todos los vértices una sola vez).
Las condiciones de factibilidad son:

- que al seleccionar una arista no se formen ciclos
- ??

Puede resolverse con dos heurísticas:

- Los nodos son candidatos. Empezar en un nodo cualquiera y en cada paso moverse al nodo no visitado **más próximo al último** nodo seleccionado
- Las aristas son candidatos (**Algoritmo Kruskal** garantizando que se empiece y se termine por la misma). Ordenar de forma creciente y coger la arista si no provoca un ciclo.

Problema de la Mochila
Tenemos n objetos y una mochila. El objeto i tiene un peso w_i y la mochila tiene una capacidad M. Si metemos en la mochila la fracción x_i con $0 \leq x_i \leq 1$ del objeto i, generamos un beneficio de valor $p_i x_i$ (p_i es el precio del objeto i)
El objetivo es rellenar la mochila de tal manera que se maximice el beneficio que produce el peso total de los objetos que se transportan. Es decir:

- Maximizar $\sum_{i=1}^{n} p_i x_i$
- Limitando que $\sum_{i=1}^{n} w_i x_i \leq M$

En este problema, Greedy proporciona soluciones optimales.
Solución Greedy:

1. Definimos la densidad del objeto por $A_i = p_i / w_i$
2. Seleccionamos los objetos en orden decreciente de densidad
3. Si es posible, se coge todo lo que se pueda de A_i. Si no se puede, se rellena el espacio que queda con una fracción del objeto.

Demostración:

- Sea $p_1/w_1 \geq p_2/w_2 \geq \ldots \geq p_n/w_n$; sea $X = (x_1, x_2, \ldots, x_n)$ la solución generada por Greedy y sea $Y = (y_1, y_2, \ldots, y_n)$ una solución cualquiera

- Vamos a demostrar que Greedy siempre encuentra la solución optimal del problema, es decir, $\sum_{i=1}^{n}(x_i - y_i)p_i \geq 0$ (Greedy maximiza mejor que cualquier otro)
- Si $x_i = 1\ \forall i$ entonces la solución es única y, por tanto, optimal
- En otro caso, sea k el menor número tal que $x_k < 1$, entonces $$\sum_{i=1}^{n}(x_i - y_i)p_i = \sum_{i=1}^{k-1}(x_i - y_i)w_i\frac{p_i}{w_i} + (x_k - y_k)w_k\frac{p_k}{w_k} + \sum_{i=k+1}^{n}(x_i - y_i)w_i\frac{p_i}{w_i}$$
    - Como se cumple que:
        - $\sum_{i=1}^{k-1}(x_i - y_i)w_i\frac{p_i}{w_i} \geq \sum_{i=1}^{k-1}(x_i - y_i)w_i\frac{p_k}{w_k}$
        - $\sum_{i=k+1}^{n}(x_i - y_i)w_i\frac{p_i}{w_i} \geq \sum_{i=k+1}^{n}(x_i - y_i)w_i\frac{p_k}{w_k}$
    - Entonces tenemos que
        - $\sum_{i=1}^{n}(x_i - y_i)p_i \geq \frac{p_k}{w_k}\sum_{i=1}^{n}(x_i - y_i)w_i$
- no sé cómo concluir

Asignación de tareas

Supongamos que disponemos de n trabajadores y n tareas. Sea $b_{ij} > 0$ el coste de asignarle el trabajo j al trabajador i. Una asignación de tareas puede ser expresada como una asignación de los valores 0 o 1 a las variables x_{ij}.

Una asignación es **válida** cuando a cada trabajador se le asigna una única tarea.

Dada una asignación válida tenemos el **coste de dicha asignación** como: $\sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}b_{ij}$

Una asignación es **óptima** si es de mínimo coste.

Dos estrategias Greedy aunque Greedy NO funciona para todos los casos:

- Asignar a cada trabajador la mejor tarea posible
- Asigna cada tarea al mejor trabajador disponible

También tenemos el **algoritmo Húngaro** que es el que sirve. Pasos:

1. Encontrar el elemento de menor coste en cada fila
    a. Construir una nueva matriz restando a cada costo el menor costo de su fila
2. En esta matriz encontrar el menor costo de cada columna
    a. Construir una nueva matriz (**costos reducidos**) restando a cada costo el menor costo de su columna
3. Rayar el mínimo número de filas y de columnas que se necesiten para tachar todos los ceros de la matriz de costos reducidos
    a. Si se necesitan n líneas para tachar todos los ceros, hemos encontrado una solución optimal en esos ceros tachados
    b. Si no, siguiente paso

4. Encontrar el menor elemento no nulo que no esté tachado (su valor será k). Restar k a cada elemento no tachado de la matriz de costos reducidos y sumar k a cada elemento de la matriz de costos reducidos que esté tachado por dos líneas.
5. Volver al paso 3

(Se cogen los últimos ceros que se hayan utilizado)