**MA4701. Optimización Combinatorial. 2013.**
**Profesor:** José Soto
**Escriba(s):** José Coronado y Gianmarco Sperone.
**Fecha:** 6 de Diciembre de 2013.

# Cátedra 27

**Ejemplo.** $TSP$ Métrico.

El problema del vendedor viajero métrico o $TSP$ métrico, se trata de encontrar un tour (paseo que pasa por todos los vértices) de largo total mínimo, dado un grafo $G = (V, E)$ que satisface desigualdad triangular.

**Observación.** $TSP$ Métrico es $NP$-difícil.

1. Idea: Tomemos $T =$ árbol generador de G de largo mínimo.

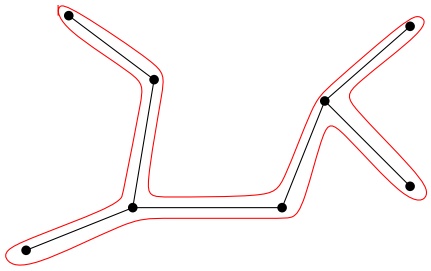

Figura 1: Tour Euleriano.

Sea $H$ el grafo obtenido al duplicar cada arista de $T$. Notar que $H$ es un grafo euleriano (cada vértice tiene grado par y es conexo / admite un paseo que pasa por cada arista una vez).
Sea $W$ el paseo euleriano de $H \Rightarrow W$ es un tour Hamiltoniano en $G$.

$$\ell(W) = 2\ell(T) \leq 2\ell(OPT)$$

ya que $OPT$ es un tour y por lo tanto

$$\ell(OPT) \geq \min_{Q \text{ conexo y generador}} \ell(Q) = \ell(T).$$

Lo que implica que es una 2-aproximación.

**Observación.** $TSP^*$ : $G$ es grafo completo y $\ell$: $E \to \mathbb{R}_+$ es métrico.

Veamos ahora cómo encontrar un tour que pasa exactamente una vez por cada nodo.

Se puede modificar $W$ para que pase por cada nodo exactamente una vez, visitando los vértices que visita $W$ en el mismo orden pero saltando los ya visitados para obtener $W^*$.

Notemos que por desigualdad triangular se tiene que $\ell(W^*) \geq \ell(W)$.

2. Christofides. ($\frac{3}{2}$-aproximación)

La idea es la misma:

- Encontrar multigrafo Euleriano generador.

- Devolver el paseo Euleriano encontrado.
- Atajos.

Idea: $T$ árbol generador de largo mínimo.

- Sea $I$ el conjunto de nodos de grado impar en $T$.
- Sea $M$ el matching perfecto de costo mínimo en $G[I]$.

  **Observación.** Esto existe si G es completo pues $|I|$ es par.

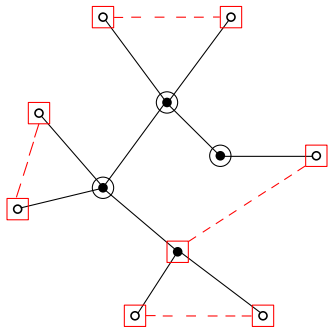

Figura 2: Los cuadrados representan a $I$, los círculos a los nodos de grado par y las líneas entrecortadas a $M$.

$T \cup M$ es Euleriano, esto significa que existe un paseo que usa cada arista de $T \cup M$ exactamente una vez.

Luego se devuelve ese paseo $W$.

Notar que $\ell(W) = \ell(T) + \ell(M)$ y $\ell(T) \leq \ell(OPT)$.

Sea $OPT$ el óptimo y $Q$ el ciclo obtenido al ignorar los vértices que no están en $I$ en $OPT$.

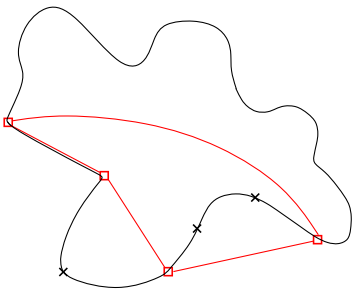

Figura 3: Los nodos cuadrados corresponden a los que están en $I$.

Se tiene que $\ell(OPT) \geq \ell(Q) = \ell(Q_{impar}) + \ell(Q_{par}) \geq 2\ell(M)$.

Donde $Q_{impar}$ y $Q_{par}$ son los matchings que componen $Q$.

Por lo tanto se tiene que

$\ell(W) \leq \ell(T) + \ell(M) \leq \ell(OPT) + \frac{\ell(OPT)}{2} = \frac{3}{2}\ell(OPT)$.

Lo que prueba que es una $\frac{3}{2}$- aproximación.

# 1. Aleatoriedad

Esta sección está dedicada a aquellos algoritmos que basan sus resultados en la toma de algunas decisiones al azar. De hecho, por **algoritmo aleatorio** nos referimos a un tipo de algoritmo que puede "lanzar monedas", es decir, que tiene acceso a un conjunto de bits aleatorios. Más específicamente, un **algoritmo aleatorio de aproximación con factor** $\alpha$ es un algoritmo que siempre devuelve una solución factible (digamos, $S$) del problema de optimización, tal que:

1. $\mathbb{E}[v(S)] \geq \frac{1}{\alpha}v^*$, para el caso de un problema de maximización, donde $v(S)$ es el valor de la solución $S$ y $v^*$ es el valor de la solución óptima.
2. $\mathbb{E}[v(S)] \leq \alpha v_*$, para el caso de un problema de minimización, donde $v_*$ es el valor de la solución óptima.

Nos centraremos en el siguiente problema de corte máximo (que es NP - difícil): dado un grafo $G = (V, E)$ y una función de pesos no negativa $w : E \to \mathbb{R}_+$, queremos hallar $S \subseteq V$ tal que $w(\delta(S))$ sea máximo, o equivalentemente, particionar $V$ en $T \cup S$ tal que la cantidad $w(E[S : T])$ sea máxima. Un algoritmo "tonto" sería partir con $S = \emptyset$ y después, para cada $v \in V$, agregar $v$ a $S$ con probabilidad $0,5$. Luego:

$$\mathbb{E}[w(\delta(S))] = \sum_{e \in E} \mathbb{P}[e \in \delta(S)]w(e) = \frac{1}{2}w(E) \geq \frac{1}{2}w^*,$$

donde $w^*$ es el valor de la solución óptima. Nos interesa "desaleatorizar" este algoritmo, es decir, transformarlo en un algoritmo determinista. Para ello, introducimos el siguiente método:

## 1.1. Método de las esperanzas condicionales

Ciertos algoritmos usan sólo "monedas" independientes, y en muchos casos se pueden lanzar todas las monedas al comienzo. Formalmente, considere $n$ variables aleatorias $X_1, X_2, \ldots, X_n$ **independientes**. Suponga además que el objetivo final es maximizar $f(X_1, X_2, \ldots, X_n)$, para cierta función apropiada $f$ que satisface:

$$\mathbb{E}[f(X_1, X_2, \ldots, X_n)] \geq \alpha.$$

(Por ejemplo, $f$ podría representar el recíproco del factor de aproximación, o la función objetivo del problema original).

La idea (con el objetivo de maximizar $f$) consiste en fijar los $X_i$ uno por uno, digamos, $X_i = S_i$, $\forall i = 1, \ldots, n$. Luego, para todo $i = 1, \ldots, n$, si ya tenemos fijadas $X_1 = S_1, X_2 = S_2, \ldots, X_{i-1} = S_{i-1}$, se calculan:

1. $A = \mathbb{E}[f(X_1, X_2, \ldots, X_n)|X_1 = S_1, X_2 = S_2, \ldots, X_{i-1} = S_{i-1}, X_i = 0]$
2. $B = \mathbb{E}[f(X_1, X_2, \ldots, X_n)|X_1 = S_1, X_2 = S_2, \ldots, X_{i-1} = S_{i-1}, X_i = 1]$

Así, si $A > B$, fijamos $S_i = 0$, y si $A \leq B$, fijamos $S_i = 1$. Al final, nuestro algoritmo debiese devolver $(S_1, S_2, \ldots, S_n)$.

Para nuestros propósitos, es necesario el siguiente lema.

**Lema 1.** $f(S_1, S_2, \ldots, S_n) \geq \mathbb{E}[f(X_1, X_2, \ldots, X_n)]$

*Demostración.* Por inducción. Para los $S_i$ elegidos por el algoritmo veremos que:

$$\mathbb{E}[f(S_1, \ldots, S_{i-1}, S_i, X_{i+1}, \ldots, X_n)] \geq \mathbb{E}[f(S_1, \ldots, S_{i-1}, X_i, \ldots, X_n)].$$

En efecto, en la $i-$ ésima iteración se tiene que $\mathbb{E}[f(S_1, \ldots, S_{i-1}, X_i, \ldots, X_n)] = \frac{A+B}{2}$. Se concluye al notar que:

$$\mathbb{E}[f(S_1, \ldots, S_{i-1}, S_i, X_{i+1}, \ldots, X_n)] = \text{máx}(A, B).$$

□

Usemos este método para el problema del corte máximo. Nuestra función $f(X_1, \ldots, X_n)$ representará el valor del corte $w(E(\delta(S))$ donde $X_i$ indica si el vértices $i$ está en $S$.

Comenzamos asignando $S = T = \emptyset$.

Después, para $i = 1, \ldots, n$, definamos $U_i = V - S - T - i$. En nuestro corte esperado puede ocurrir que:

1. Si decidieramos poner $i$ en $T$ (es decir, $X_i = 0$), tendríamos

$$\begin{aligned} A &= \mathbb{E}[f(X_1, X_2, \ldots, X_n) | X_1 = S_1, X_2 = S_2, \ldots, X_{i-1} = S_{i-1}, X_i = 0] \\ &= w(E[S:T]) + w(E[i:S]) + \frac{1}{2}w(E[S:U]) + \frac{1}{2}w(E[T:U]) + \frac{1}{2}w(E[U]). \end{aligned}$$

2. Si se pone $i \in S$, (es decir $X_i = 1$) entonces:

$$\begin{aligned} B &= \mathbb{E}[f(X_1, X_2, \ldots, X_n) | X_1 = S_1, X_2 = S_2, \ldots, X_{i-1} = S_{i-1}, X_i = 1] \\ &= w(E[S:T]) + w(E[i:T]) + \frac{1}{2}w(E[S:U]) + \frac{1}{2}w(E[T:U]) + \frac{1}{2}w(E[U]). \end{aligned}$$

Por lo tanto, $A > B$ si y sólo si $w(E[i:S]) > w(E[i:T])$ (y en este caso, nos conviene fijar $X_i$ a ser 0, es decir, poner $i$ en $T$). El algoritmo "desaleatorizado" será finalmente:

**Algoritmo 1** Corte Máximo

```
S ← ∅, T ← ∅.
for i = 1, ..., n do
    if w(E[i : S]) > w(E[i : T]) then
        T ← T + i.
    else
        S ← S + i.
    end if
end for
return S.
```

Gracias al lema anterior tenemos que el valor del corte entregado es:

$$w(\delta(S)) = f(S_1, \ldots, S_n) \geq \mathbb{E}[f(X_1, \ldots, X_n)] \geq \frac{1}{2}w^*$$

Es decir, nuestro algoritmo desaleatorizado devuelve una 2-aproximación.