# Algoritmos de emparejamiento

Mager, Jesús

[1]Universidad Autónoma Metropolitana
Unidad Azcapozalco

2015

# Introducción

**El presente trabajo es un resumen del libro "Combinatorial Optimization. Algorithms and Complexity" escrito por Christos H. Papadimitriou y Kenneth Steiglitz.**

Un emparejamiento en un grafo es un conjunto de aristas que no comparten un mismo nodo. Buscaremos la mejor combinación para tener el máximo número de emparejamientos posible.
También podemos agregar pesos a las aristas, donde el reto sería encontrar el conjunto de aristas que contengan el peso mayor.
Estos algoritmos pueden ser consideras instancias del concepto camino de incremento usado en Max-Flow. Se tratarán los grafos bipartitos primero por ser más sencillos para su resolución.

# Conceptos

- Un emparejamiento $M$ es un subconjunto de $G = (V, E)$ con la propiedad de que ningún par de aristas comparten algún nodo.
- Un *emparejamiento máximo* es el que más se acerque a $|V|/2$.
- Dado $G$, el problema del emparejamiento es encontrar un emparejamiento máximo $M$ sobre $G$.
- Cuando la cardinalidad de un emparejamiento es $\llcorner |V|/2 \lrcorner$, el emparejamiento mas grande posible en un grafo de $|V|$ nodos, decimos que el emparejamiento es *completo* o *perfecto*.
- Los arcos en $M$ serán llamados *arcos emparejados*; los restantes nodos *libres*.
- Si un camino contiene nodos *libres* entonces será llamado un camino *alternativo*.
- Los vértices que se encuentran en caminos alternativos comenzando con un vértice expuesto y tienen un rango impar en este camino son llamados *exteriores*; que tienen un rango par se llaman interiores.

Con esto, planteamos:
**Lema:** Sea $P$ un conjunto de arcos en un camino de incremento $p = [u_1, u_2, \ldots, u_{2k}]$ en un grafo $G$ con respecto al emparejamiento $M$. Entonces $M' = M \bigoplus P$ es una cardinalidad de emparejamiento $|M| + I$. (El operador $S \bigoplus T$ denota la *diferencia simétrica* de $S$ y $T$, definida como $S \bigoplus T = (S - T) \cup (T - S)$ )
**Teorema:** Un $M$ de emparejamiento en un grafo $G$ es máximo, si y sólo si, no existe un camino de incremento en $G$ respecto a $M$.
Por esto, será posible plantear que dado un emparejamiento, se puede buscar caminos de aumento, hasta que ya no exista alguno nuevo, con lo que se resolverá el problema.

# Algoritmo de emparejamiento bipartita

**Entrada:** Un grafo bipartita $B = (V, U, E)$.
**Salida:** El máximo emparejamiento de $B$ representado como un arreglo *mate*.

**for all** $v \in V \cup U$ **do**
    $mate[v] \leftarrow 0$ ▷ Inizializando
**end for**
**for all** $v \in V$ **do**
    $exposed[v] \leftarrow 0$
**end for**
$A \leftarrow \varnothing$ ▷ Construimos el grafo auxiliar
**for all** $[v, u] \in E$ **do**
    **if** $mate[u] = 0$ **then**
        $exposed[v] \leftarrow u$
    **else if** $mate[u] \neq$ **then**
        $A \leftarrow A \cup (v, mate[u])$
    **end if**
    $Q \leftarrow \varnothing$
    **for all** $v \in V$ **do**
        **if** $mate[v] = 0$ **then**
            $Q \leftarrow Q \cup \{v\}, label[v] \leftarrow 0$
        **end if**
    **end for**
    Continua en la siguente página.
**end for**

# Algoritmo de emparejamiento bipartita (Continuación)

**while** $Q \neq \varnothing$ **do**
    Sea $v$ un nodo de $Q$
    remueve $v$ de $Q$
    **if** $exposed[v] \neq 0$ **then**
        AUGMENT(v), **goto** estado
    **else**
        **for all do** $v's \notin label$ tal que $(v, v') \in A$:
            $label[v'] \leftarrow v, Q \leftarrow Q \cup \{v'\}$
        **end for**
    **end if**
**end while**
**function** AUGMENT(v)
    **if** $label[v] = 0$ **then**
        $mate[v] \leftarrow exposed[v], mate[exposed[v]] \leftarrow v$
    **else**
        $exposed[label[v]] \leftarrow mate[v]$
        $mate[v] \leftarrow exposed[v]$
        $mate[exposed[v]] \leftarrow v$
        AUGMENT(label[v])
    **end if**
**end function**

Cada incremento también incrementa la cardinalidad del emparejamiento en uno, por lo que podemos tener a lo mucho mín$(|V|, |U|)$ estados. La complejidad de cada estado es de $O(|E|)$. La contrucción del grafo auxiliar y el calculo del arreglo *expuesto* requiern $O(|E|)$, ya que $|E|$ arcos deben de considerarse. Por lo que el algoritmo resuele correctamente el problema del emparejamiento sobre un grafo bipartito $B = (V, U, E)$ en $O(\text{mín}(|V|, |U|)|E|))$.

El resultado de $O(\text{mín}(|V|, |U|)|E|))$ se puede mejorar. Se puede recudir el problema de del emparejamiento bipartito a un problema de max-flow para redes sencillas. Para el grafo bipartita $B = (V, U, E)$ definimos la siguente red $N(B) = (s, t, W, A)$, donde $s$ y $t$ son dos nodos nuevos, $W$ es la unión entre $\{s, t\}$, $V$ y $U$, y $A$, es un conjunto de arcos consistentes de tres categorías:

- Los arcos $(s, v)$ para todo $v \in V$
- Los arcos $(u, t)$ para todo $u \in U$
- Los arcos $(v, u)$ para todo $v \in V$ y $v \in V$ tal que $[v, u] \in E$.

# Apareamiento no bipartito: blossoms

**Lema:** La cardinalidad del máximo apareamiento de un grafo bipartita $B$ equivale al valor del máximo flujo $s - t$
**Teorema:** Podemos resolver el porblmea del emparejameiento bipartita en $O(|V|^{1/2}|E|)$.

**Teorema:** Suponga que mientras se busca po run camino de incremento desde un no do expuesto en un grafo $G$ con respecto al apareamiento $M$, descubrimos un *blossom* en $b$. Entonces, existe un camino alternativo desde $u$ a cualquier nodo en $b$, terminando en un arco apareado.

**Teorema:** Suponga que, mientras se busca por un camino de incremento desde un nodo $u$ de un grafo $G$ con respecto al emparejamiento $M$, se descubre un *blossom* $b$. Entonces ixiste un camino de aumento desde $u \in G$ con respecto a $M$ hay uno de $u$ ($v_b$ si $u$ está en la base de $b$) en $G/b$ con respecto a $M/b$.

El algoritmo para la solución del problema de apareamiento no bipartita es de $O(|V|^4)$. Se utiliza el algoritmo del caso bipartita, junto a los métodos necesarios para enfrentarse a los blossoms. Se va a usar el digrafo auxiliar como en el primer algoritmo y se va a realizar un vértice expuesto a la vez.

**Teorema:** Suponga que un grafo $G$ no existe un camino de incremento desde un vértice expuesto $u$ con respecto al apareamiento $M$. Sea $p$ un camino de aumento que termina en otros nodos expustas $v$ y $w$. Entonces no existe un camino de incremento desde $u$ respecto a $M \bigoplus P$ tampoco.

**Crolario:** Si en algún estado no existe un camino de aumento desde un nodo $u$, entonces nunca habrá caminos de aumento desde $u$.

Por lo tanto, si la búsqueda falla en encontrar un camino de incremento desde el nodo $u$, $u$ nunca volverá a ser considerado como un potencial punto de partida. Se mantendrá esto en un registro llamado $considered[u] \leftarrow 1$

# Algoritmo no bipartita de emparejamiento

**Entrada:** Un grafo $G = (V, E)$
**Salida:** Un emparejamiento máximo de $G$ en terminos del arreglo *mate*.
**for all** $v \in V$ **do**
    $mate[v] \leftarrow 0, considered[v] \leftarrow 0$
**end for**
estado:
**while** $\exists v \in V$ con $considered[u] = 0$ y $mate[u] = 0$ **do**
    $considered[] \leftarrow 1, A \leftarrow \varnothing$
    **for all** $v \in V$ **do**
        $exposed[v] \leftarrow 0$
        **for all** $[v, w] \in E$ **do**
            **if** $mate[w] = 0$ y $w \neq u$ **then**
                $exposed[v] \leftarrow w$
            **else if** $mate[w] \neq v$ **then**
                $A \leftarrow A \cup (v, mate[w])$
            **end if**
        **end for**
        **for all** $v \in V$ **do**
            $seen[v] \leftarrow 0$
        **end for**
        $Q \leftarrow u, label[u] \leftarrow 0$
        Continúa...
    **end for**
**end while**

# Algoritmo no bipartita de emparejamiento. Continuación...

**while** $Q \neq \varnothing$ **do**
  Sea $v$ un nodo de $Q$
  Remueve $v$ de $Q$
  **for all** los nodos no etiquetados $w \in V$ tales que $(v, w) \in A$ **do**
    $Q \leftarrow Q \cup w$, $label[w] \leftarrow v$
    $seen[mate[w]] \leftarrow 1$
    **if** $exposed[w] \neq 0$ **then**
      AUGMENT(w)
      **goto** estado
    **end if**
    **if** $seen[w] = 1$ **then**
      BLOSSOM(w)
    **end if**
  **end for**
**end while**

**Teorema:** El algoritmo encuentra correctamente el emparejamiento máximo en un grafo $G = (V, E)$ en $O(|V|^4)$