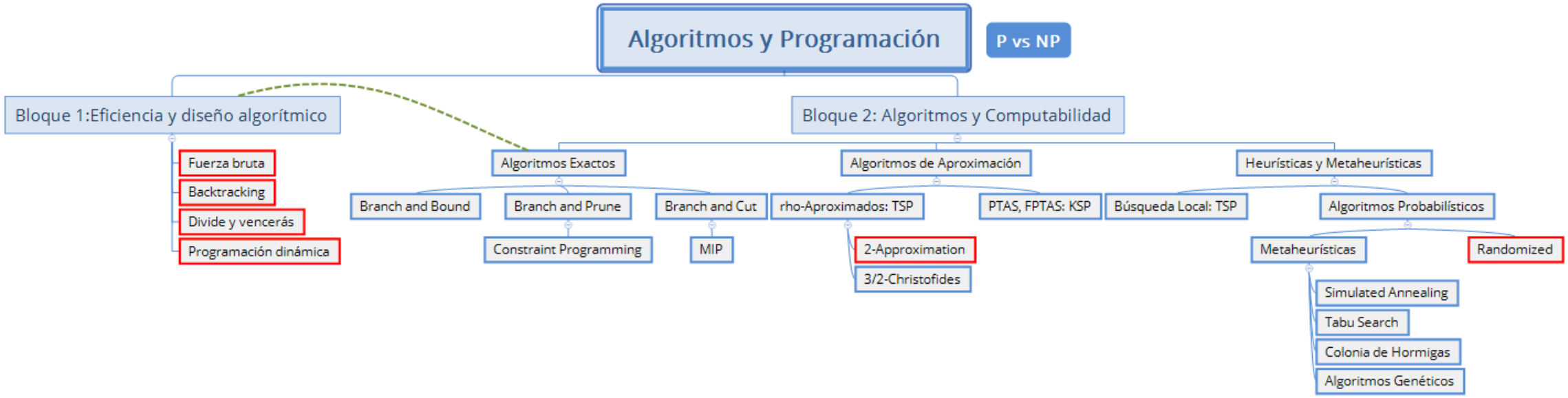
Algoritmos y Programación

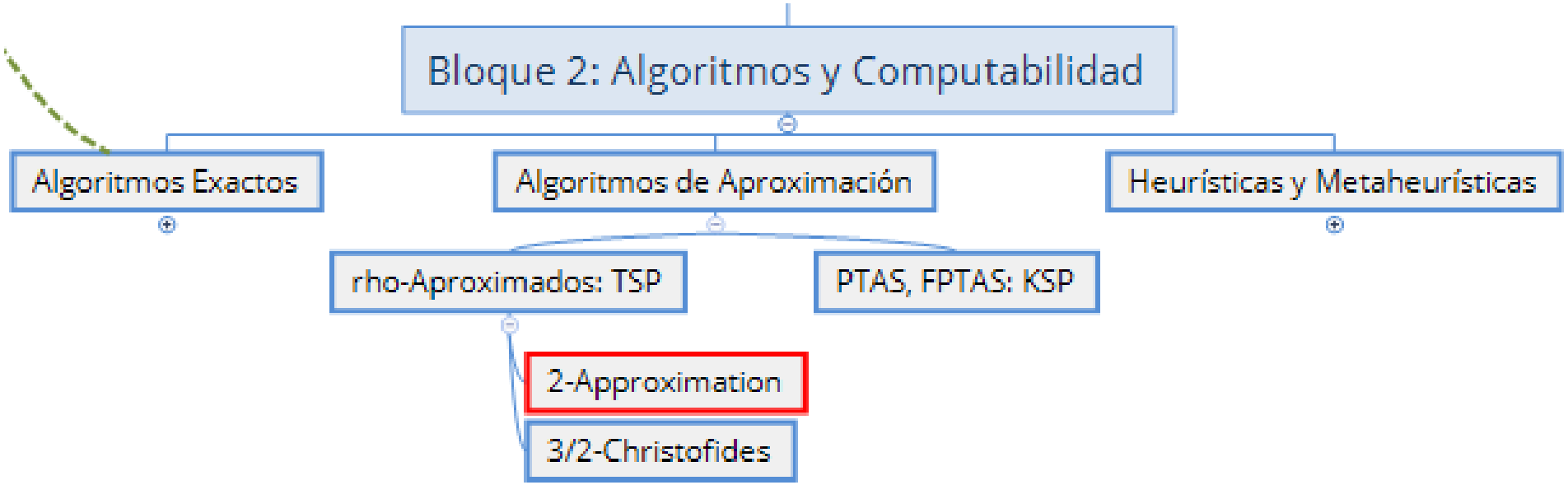
Algoritmos de Aproximación

Bloque 2: Algoritmos y Computabilidad

Algoritmos y Programación



Algoritmos y Programación



P≠NP

- Encontrar soluciones óptimas
- En tiempo polinómico
- Para cualquier dimensión del problema

- En general, elegir 2 de las 3

Algoritmos de Aproximación

Intentan acotar la solución a una cierta distancia del óptimo

Algoritmos de aproximación

En ciencias de la computación e investigación de operaciones, un **algoritmo de aproximación** es un algoritmo usado para encontrar soluciones aproximadas a problemas de optimización. Están a menudo asociados con problemas NP-hard; como es poco probable que alguna vez se descubran algoritmos eficientes de tiempo polinómico que resuelvan exactamente problemas NP-hard, se opta por encontrar soluciones no-óptimas en tiempo polinomial. A diferencia de las heurísticas, que usualmente sólo encuentran soluciones razonablemente buenas en tiempos razonablemente rápidos, lo que se busca aquí es encontrar soluciones que está demostrado son de calidad y cuyos tiempos de ejecución están acotadas por cotas conocidas… un ***ρ*-algoritmo de aproximación** se ha demostrado que la aproximación no será mayor (o menor, dependiendo de la situación) que un factor ρ veces la solución óptima.

https://es.wikipedia.org/wiki/Algoritmo_de_aproximación

ratio de aproximación o factor de aproximación

Algoritmos de aproximación

- Esquemas de aproximación en tiempo polinómico (*PTAS Polynomial-Time Approximation Scheme*). Ejemplos:
  - Problema del agente viajero (TSP)
  - Problema de la mochila
  - …

Strict NP

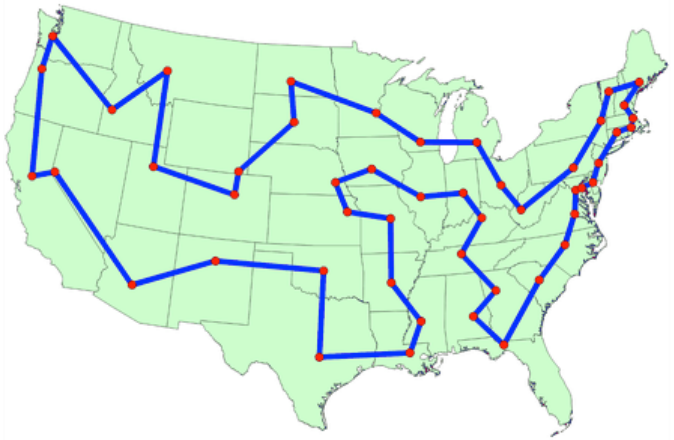
- Problemas MAX SNP. Para cualquier problema que pertenezca a la clase MAX SNP no existe un PTAS para el problema a menos que P=NP. Ejemplos:
  - Problema de satisfacibilidad (*maximum satisfiability problem*)
  - Problema de corte máximo (*maximum cut problem*)

https://en.wikipedia.org/wiki/SNP_(complexity)#MaxSNP

Problema del Agente Viajero (TSP)

"Si un viajante parte de una ciudad y las distancias a otras ciudades son conocidas,

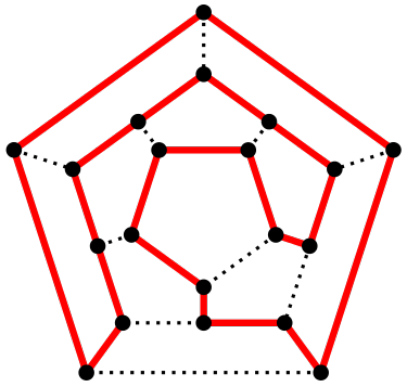
¿cuál es la ruta óptima que debe elegir para visitar todas las ciudades y volver a la ciudad de partida?"



https://es.wikipedia.org/wiki/Travelling_salesman_problem

Conceptos relacionados

- Un **camino hamiltoniano** en un grafo es un camino que "visita" cada vértice una y sólo una vez
- Un **ciclo** (o circuito) es un camino que empieza y acaba en el mismo vértice



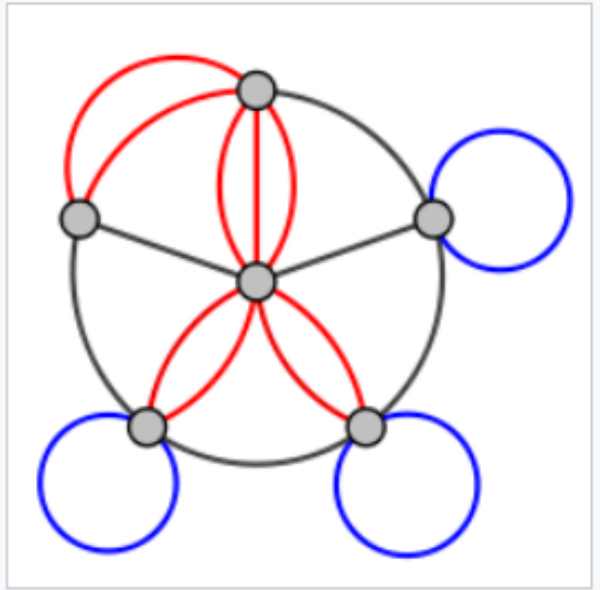
Hamilton

1805-1865

https://es.wikipedia.org/wiki/Anexo:Glosario_de_teor%C3%ADa_de_grafos

Conceptos relacionados

- Multigrafo: es un grafo que está facultado para tener aristas múltiples, es decir, aristas que relacionan los mismos nodos



Un multigrafo con múltiples aristas (en rojo) y tres bucles (en azul). No todos los autores permiten multigrafos con bucles.

https://es.wikipedia.org/wiki/Multigrafo

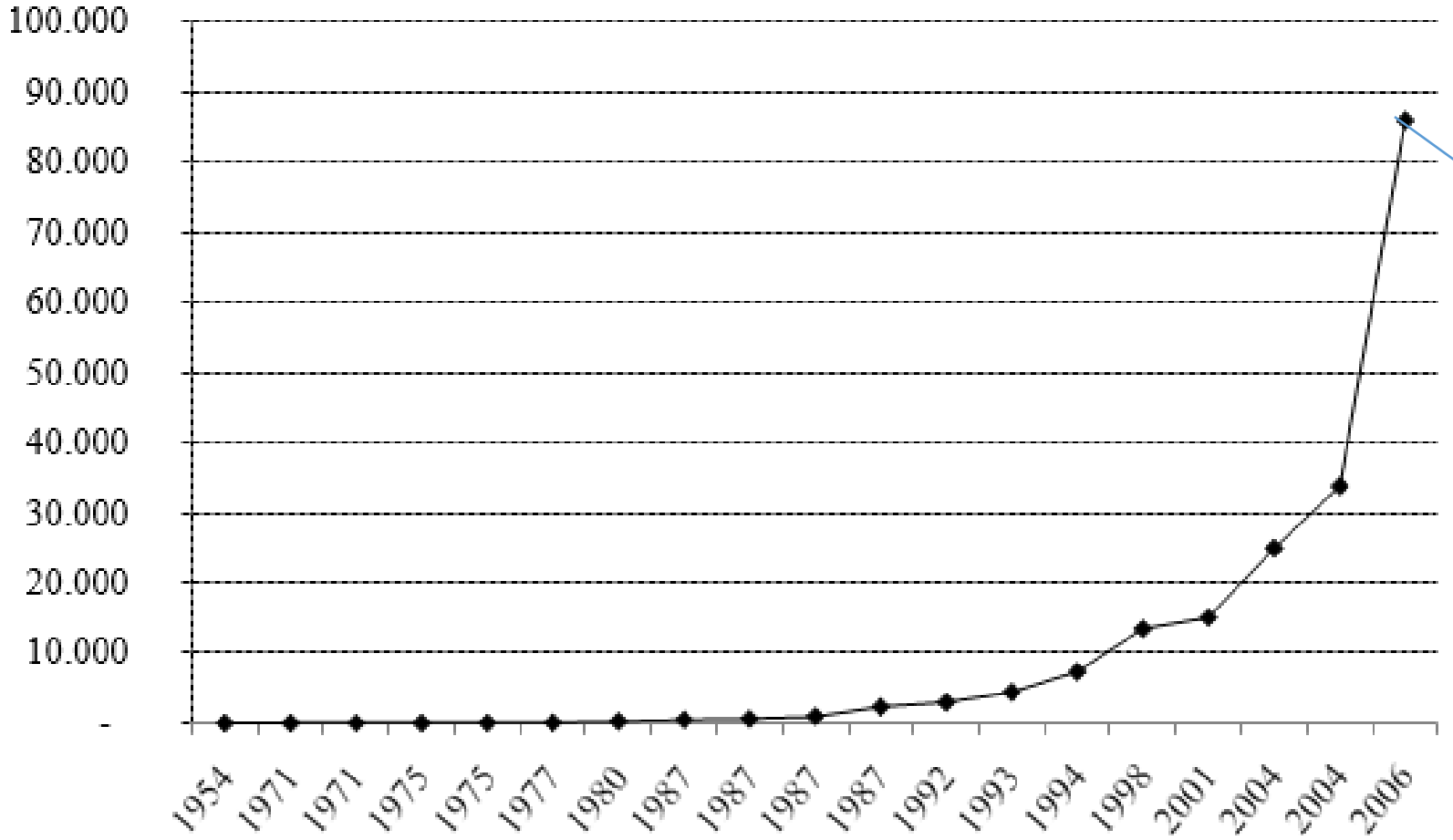
Variaciones del TSP

- TSP simétrico (sTSP)
- TSP asimétrico (aTSP)
- mTSP: Dado un conjunto de nodos donde tenemos m vendedores localizados en un único nodo al que llamaremos almacén, y el resto de nodos (ciudades) que tiene que visitar les llamaremos nodos intermedios. El mTSP consiste en encontrar circuitos de coste mínimo para los m vendedores, que empiecen y terminen en le almacén y de manera que cada nodo intermedio se visita una única vez (VRP)
- Variaciones de mTSP

Progreso en el número de ciudades

1954	G. Dantzig, R. Fulkerson, S. Johnson	49 ciudades
1971	M. Held, R.M. Karp	57 ciudades
1971	M. Held, R.M. Karp	64 ciudades
1975	P.M. Camerini, L. Fratta, F. Maffioli	67 ciudades
1975	P. Miliotis	80 ciudades
1977	M. Grötschel	120 ciudades
1980	H. Crowder and M. W. Padberg	318 ciudades
1987	M. Padberg and G. Rinaldi	532 ciudades
1987	M. Grötschel and O. Holland	666 ciudades
1987	M. Padberg and G. Rinaldi	1002 ciudades
1987	M. Padberg and G. Rinaldi	2392 ciudades

Progreso en el número de ciudades



- Algoritmo Concorde
- 85.900 ciudades
- 130.000 líneas de código en C

Aplicaciones de TSP

- Logística
  - Planificadores de rutas
  - Rutas escolares
  - Reparto de correo
- Industria
  - Secuenciación de tareas
  - Producción de circuitos electrónicos
    - Problemas de perforado
    - Conexión de circuitos integrados (minimizar la cantidad de cable necesaria)
- Secuenciación de ADN

Coste computacional

- n ciudades => (n-1)!
- En 1962, Held y Karp encuentran un algoritmo $O(n^2 2^n)$, y es el mejor de los descritos hasta el momento. Pudo resolver 50 ciudades.
- TSP euclídeo: NP-Completo

2-Aproximado

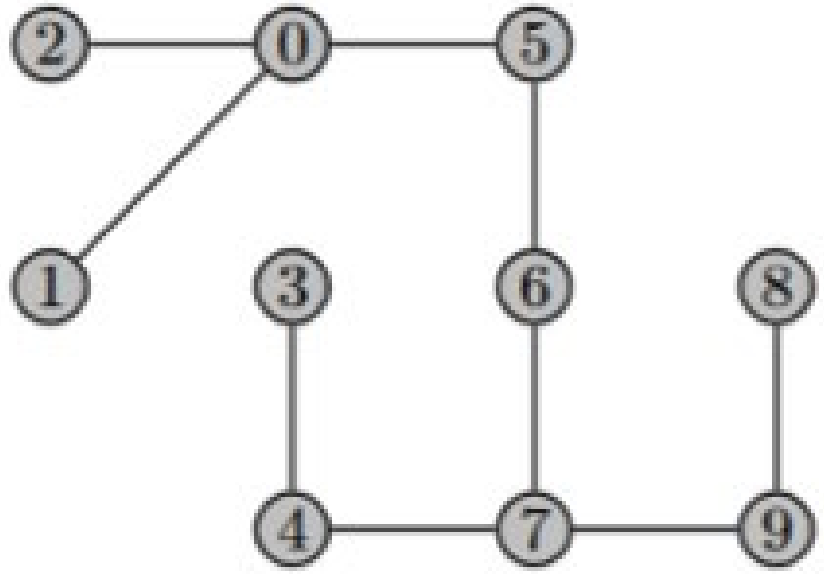
- Supongamos:
  - S = “Subconjunto de aristas de un multigrafo, G”
  - C(S) = “suma de los costes de S”
  - Definimos un ciclo que pasa por todos los vértices, y queremos obtener el ciclo de coste mínimo -> Queremos obtener el ciclo hamiltoniano, H* de coste mínimo del grafo G, *min C(H*(G))*

  - ¿Qué conecta todos los vértices de un grafo con coste mínimo?

2-Aproximado

- ¿Qué conecta todos los vértices de un grafo con coste mínimo?
  - Árbol de Expansión Mínima, T (Kruskal, Prim, Boruvka)

Un árbol no es un ciclo



(a) Paso (1) Árbol de expansión mínima. T

TSP: Algoritmo del árbol

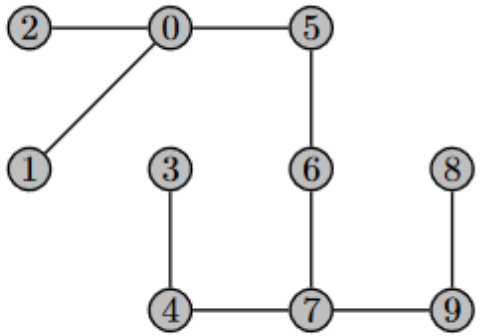
- 2-Aproximado del sTSP
- El coste de las aristas debe cumplir la desigualdad triangular. Dados 3 vértices se debe cumplir:

$$c_{ij} \leq c_{ik} + c_{kj}$$

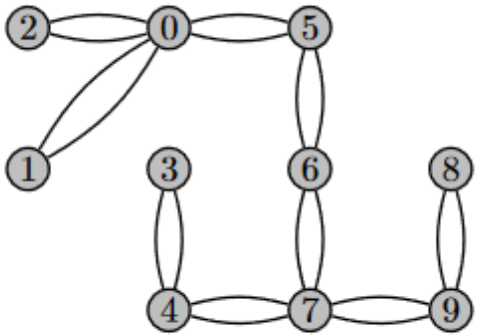
- Encontrar el Árbol de Expansión Mínima, T (Kruskal, Prim, Boruvka)
- Crear un multigrafo G* duplicando todas las aristas del árbol T
- Encontrar una cadena euleriana de G* y un circuito hamiltoniano, H*, embebido en ella

2-Aproximado

- Para convertirlo en un ciclo, C, podemos duplicar todas las aristas



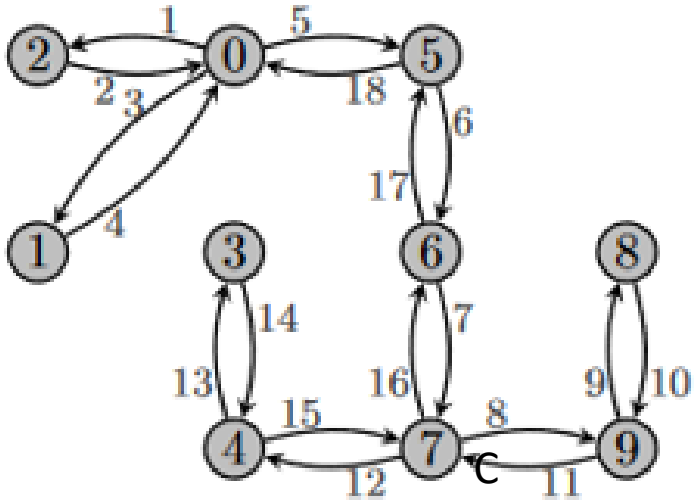
(a) Paso (I) Árbol de expansión mínima. T



(b) Paso (II) Creando el multigrafo.

2-Aproximado

- Supongamos que la raíz del árbol es el vértice 0, y que utilizamos un recorrido en preorden

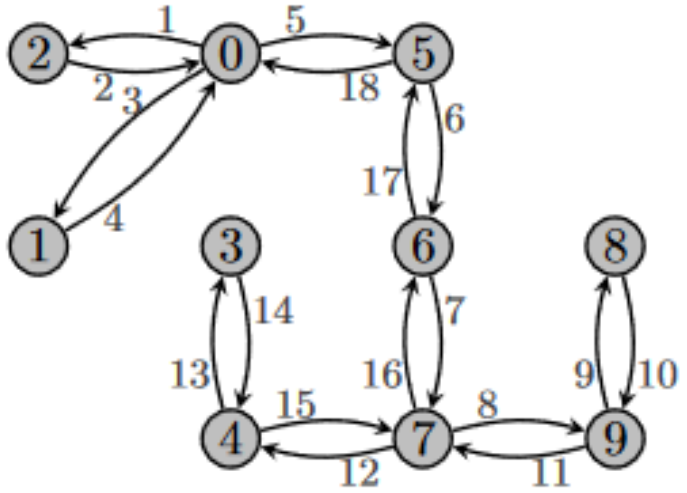


(c) Paso (III) Cadena de Euler:
$(\underline{0}, \underline{2}, 0, \underline{1}, 0, \underline{5}, \underline{6}, \underline{7}, \underline{9}, \underline{8}, 9, 7, \underline{4}, \underline{3}, 4, 7, 6, 5, \underline{0})$.

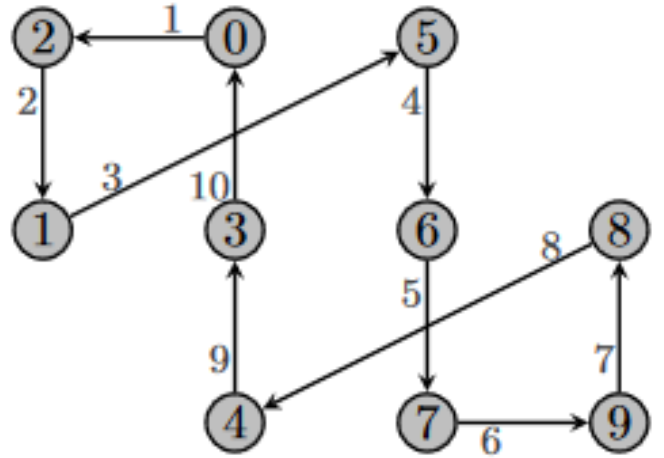
- El problema es que atraviesa los vértices más de 1 vez
- Por ejemplo: 0->2->0->1->…

2-Aproximado

- Como se cumple la desigualdad triangular (0->2->0->1 …)
  - 2->1 siempre será menor o igual que 2->0->1
  - Por tanto, podemos eliminar el paso por los vértices duplicados, y obtenemos C’ (C sin duplicados)



(c) Paso (III) Cadena de Euler: C
$(\underline{0}, \underline{2}, 0, \underline{1}, 0, \underline{5}, \underline{6}, \underline{7}, \underline{9}, \underline{8}, 9, 7, \underline{4}, \underline{3}, 4, 7, 6, 5, \underline{0})$.



(d) Paso (III) Circuito hamiltoniano: C’
$(0, 2, 1, 5, 6, 7, 9, 8, 4, 3, 0)$.

2-Aproximado

- C -> cadena o circuito de Euler (con vértices duplicados)
- C’ -> circuito hamiltoniano (sin vértices duplicados)
- $c(C') \leq c(C)$, el coste de C’ es menor o igual que el coste de C
- Como C se obtiene a partir de la duplicación de las aristas del árbol de expansión mínima: $c(C) = 2 \cdot c(T)$
- Por tanto, $c(C') \leq 2 \cdot c(T)$ [ec. 1]

2-Aproximado

- $c(C') \leq 2 \cdot c(T)$ [ec. 1]
- Si al camino óptimo H* le quitamos una arista, *a*, obtenemos un árbol de expansión, no necesariamente mínimo, T'
- $c(H_G^*) \geq c(H_G^* - a) \geq c(T)$ [ec. 2]

- **Lema 1:** El coste mínimo (OPT) del TSP es siempre mayor o igual que el coste del MST, T

- Combinando [ec. 1] y [ec. 2]

$$c(C') \leq 2 \cdot c(H_G^*)$$

Que demuestra que el circuito C' es mejor o igual que 2 veces el mejor camino

TSP

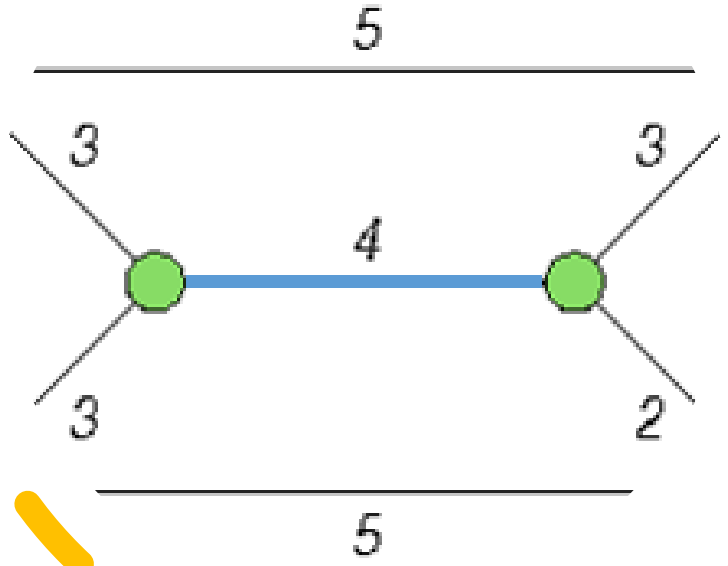
- ¿Se podría hacer mejor?

- Antes de continuar, … supongamos:
  - $S \subseteq V$
  - H_S^* es el camino hamiltoniano de S

  - $c(H_S^*) \leq c(H_G^*)$, esto es cierto debido a la desigualdad triangular.

  (Eliminar vértices no puede aumentar el tamaño del circuito)

Emparejamiento perfecto de coste mínimo

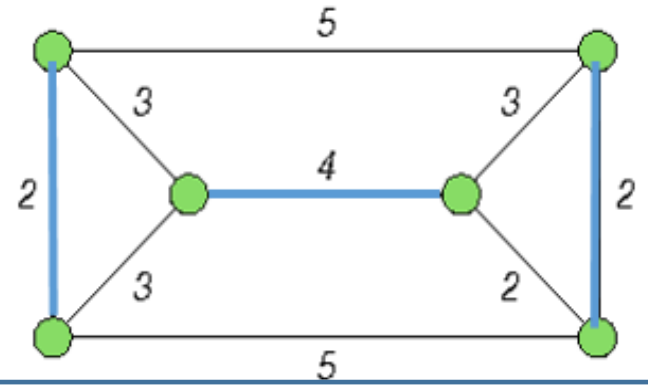


- El problema del emparejamiento perfecto asociado a un grafo *G= (V,A)* con un número par de vértices y unos costes c, consiste en emparejar cada vértice i de V con un y sólo un vértice j de V a coste mínimo, $\forall i, j \in V, i \neq j$

El algoritmo de Blossom (Edmond, 1961) resuelve el problema en tiempo polinómico $O(|V|^4)$

Problema en Aula: emparejamiento_perfecto

Dado un grafo, con un número par de vértices diseñar un modelo que nos permita obtener el emparejamiento perfecto de coste mínimo. El coste entre un par de vértices (coste de la arista) viene determinado en una tabla. (Ejemplo, coste mínimo 8)



[El problema del emparejamiento perfecto asociado a un grafo *G=(N,M)* con un número par de vértices y unos costes c, consiste en emparejar cada vértice i de N con un y sólo un vértice j de N a coste mínimo, $\forall i, j \in N, i \neq j$]

Parámetros

```
include "globals.mzn";

int: n = 6;
set of int: VERTEX = 1..n;

array[VERTEX, VERTEX] of int: weights = [|
10000,   5,     3,     10000,  2,     10000|
5,       10000, 10000, 3,      10000, 2|
3,       10000, 10000, 4,      3,     10000|
10000,   3,     4,     10000,  10000, 2|
2,       10000, 3,     10000,  10000, 5|
10000,   2,     10000, 2,      5,     10000|
|];
```

Salida esperada:

8 -> [5, 6, 4, 3, 1, 2]

Circuito euleriano

- Dado un grafo, recorrer las aristas sin levantar el lápiz del papel
- Básicamente, lo que hacemos es ir a un vértice y salir de él, tantas veces como haga falta. Eso implica que debe haber un número par de aristas para cada vértice, una que entra y otra que sale.
- Un grafo tiene un circuito euleriano si y solo si, cada vértice tiene grado par, y todo grafo con todos sus vértices de grado par tiene un circuito euleriano.

3

Algoritmo de Christofides

- ¿Se podría hacer mejor que con el algoritmo del árbol de expansión mínima, con cota 2 veces el óptimo?

- ¿Por qué aparece el factor 2?
  - Porque duplicamos todas las aristas del árbol de expansión mínima, T.
- ¿Por qué duplicamos todas las aristas?
  - Porque necesitamos un grafo euleriano, de ahí obtenemos un ciclo euleriano y de éste un ciclo hamiltoniano
- ¿Podemos aumentar T para obtener un grafo euleriano sin tener que incluir todas las aristas?

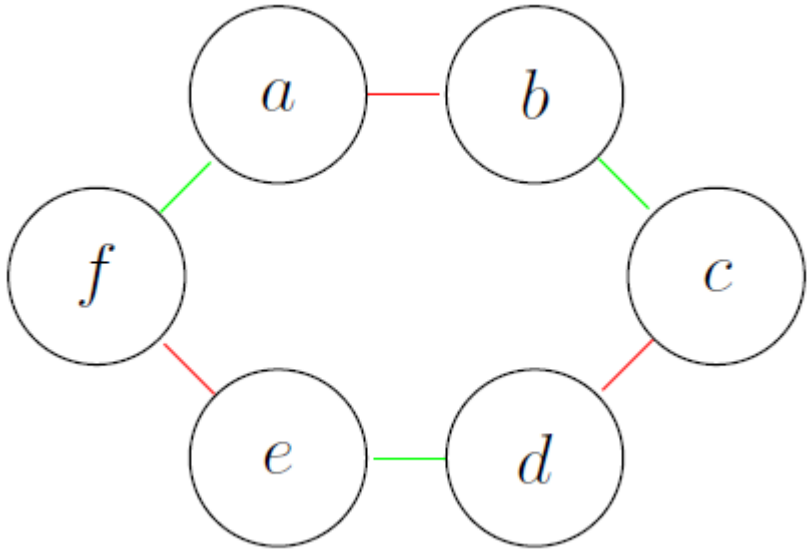
Algoritmo de Christofides

- ¿Podemos aumentar T para obtener un grafo euleriano sin tener que incluir todas las aristas?
- **Lema 2:** Sea $S \subseteq T$, un subconjunto par de vértices y M el coste mínimo del **emparejamiento perfecto** de S, entonces

$$\sum_{e \in M} c_e \leq \frac{1}{2} \cdot OPT \ [ec. 3]$$

Donde OPT denota el camino óptimo del TSP

Algoritmo de Christofides



- Supongamos que el camino OPT es el que figura en el grafo
- Y separamos las aristas en 2 subconjuntos disjuntos de emparejamiento perfecto (verde y rojo). Uno mínimo y otro no.
- El emparejamiento perfecto de coste mínimo, será siempre, por definición, menor o igual que el otro emparejamiento
- Como la suma de las aristas es OPT, la suma de las aristas del emparejamiento perfecto será menor o igual que OPT/2

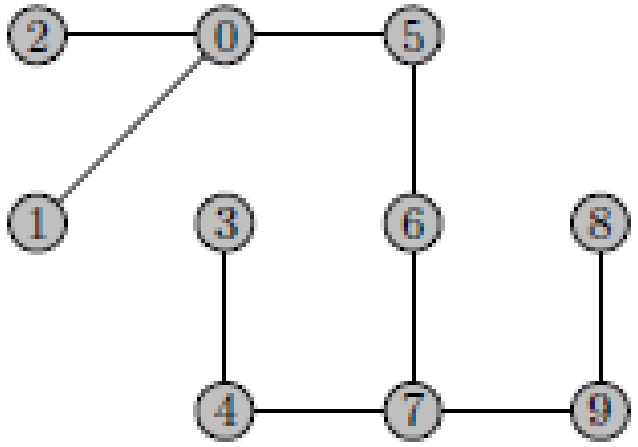
$$\sum_{e \in M} c_e \leq \frac{1}{2} \cdot OPT$$

Algoritmo de Christofides

1. Obtener el Árbol de Expansión Mínima, T
2. Separar los vértices de T de grado impar, W
3. Obtener M, el emparejamiento perfecto de coste mínimo de W
4. Añadir las aristas de M a T
5. Obtener C, el camino euleriano de H
6. Eliminar las ocurrencias de vértices repetidos para obtener el camino hamiltoniano

Algoritmo de Christofides

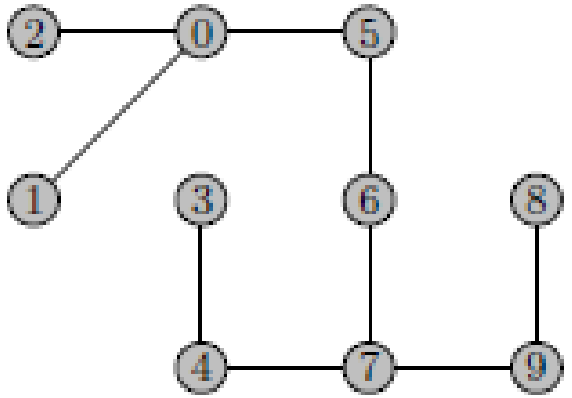
1. Obtener el Árbol de Expansión Mínima, T



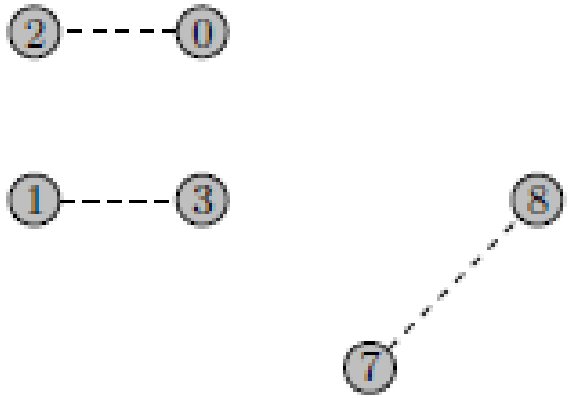
(a) Paso (I) Árbol de expansión mínima.

Algoritmo de Christofides

2. Separar los vértices de T de grado impar, W
3. Obtener M, el emparejamiento perfecto de coste mínimo de W



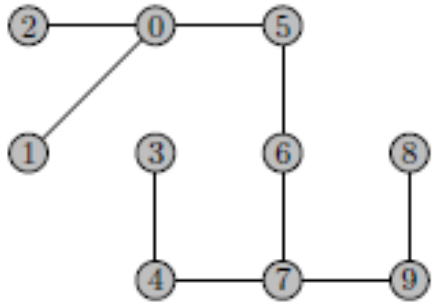
(a) Paso (I) Árbol de expansión mínima. T



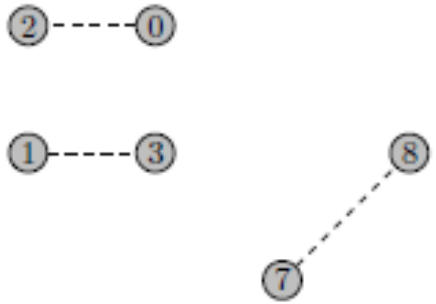
(b) Paso (II) Nodos de grado impar y emparejamiento óptimo. M

Algoritmo de Christofides

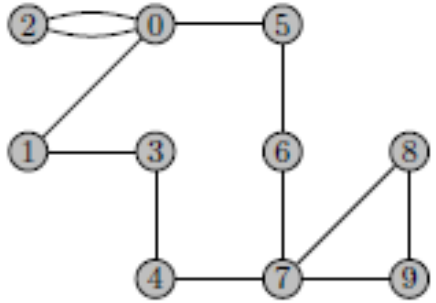
4. Añadir las aristas de M a T



(a) Paso (I) Árbol de expansión mínima.



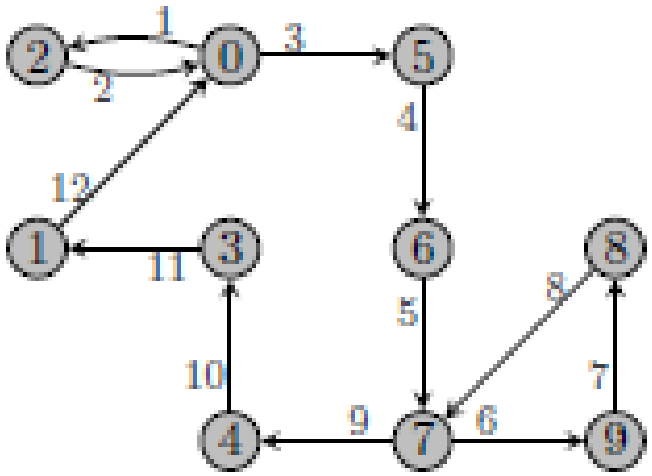
(b) Paso (II) Nodos de grado impar y emparejamiento óptimo.



(c) Paso (II) Creando el multigrafo.

Algoritmo de Christofides

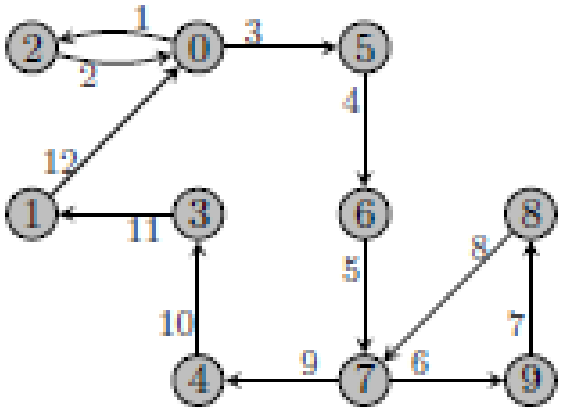
5. Obtener C, el camino euleriano de H



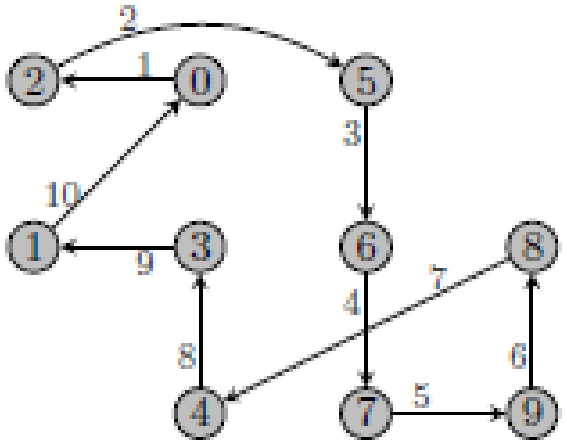
(d) Paso (III) Cadena de Euler:
$(\underline{0}, \underline{2}, 0, \underline{5}, \underline{6}, \underline{7}, \underline{9}, \underline{8}, 7, \underline{4}, \underline{3}, \underline{1}, \underline{0})$.

Algoritmo de Christofides

6. Eliminar las ocurrencias de vértices repetidos para obtener el camino hamiltoniano



i) Paso (III) Cadena de Euler:
$(\underline{0}, \underline{2}, 0, \underline{5}, \underline{6}, \underline{7}, \underline{9}, \underline{8}, 7, \underline{4}, \underline{3}, \underline{1}, \underline{0})$.



(e) Paso (III) Circuito hamiltoniano:
$(0, 2, 5, 6, 7, 9, 8, 4, 3, 1, 0)$.

Coste del algoritmo de Christofides

Lema 1: El coste mínimo (OPT) del TSP es siempre mayor o igual que el coste del MST, T

Lema 2: Sea $S \subseteq T$, un subconjunto par de vértices y M el coste mínimo del **emparejamiento perfecto** de S, entonces

$$\sum_{e \in M} c_e \leq \frac{1}{2} \cdot OPT$$

$$c(T \cup M) \leq OPT + \frac{1}{2} OPT = \frac{3}{2} OPT$$

Algoritmo de Christofides es $\frac{3}{2}$ Aproximado

Laboratorio

- VPL: Implementar en Python
  - Implementar el Algoritmo de Christofides, para lo cual puede usar el paquete de python NetworkX