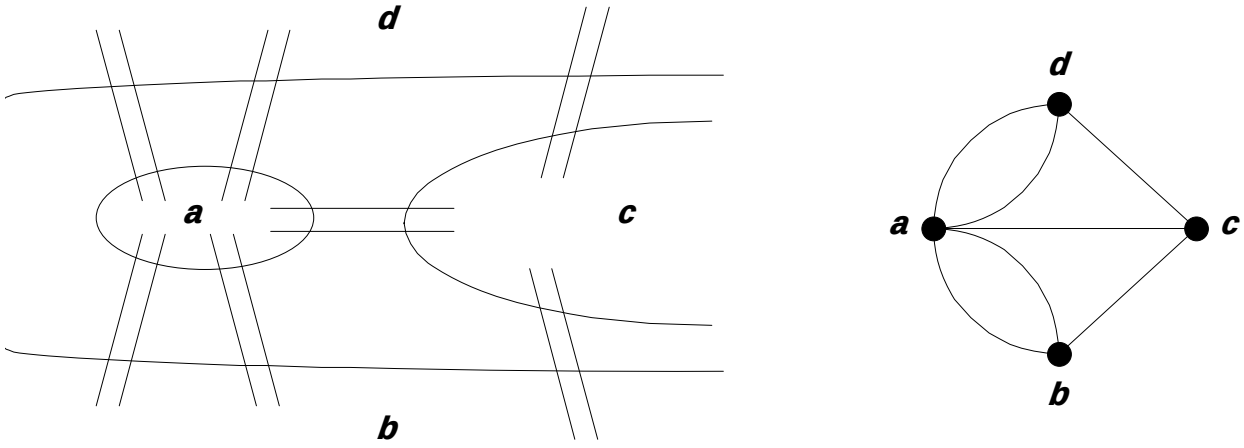


Figura 1: Los puentes de Königsberg y su grafo asociado.

Algoritmo de Fleury

```
para cada v∈V hacer
   { si G es dirigido entonces
        { d+(v) ← Grado de salida de v
          d-(v) ← Grado de entrada de v
        }
     en otro caso
        d(v) ← Grado de salida de v
   }
si G es dirigido entonces
   { si d+(v) = d-(v), ∀v∈V entonces
        { Existe un circuito
          x ← Elegir un vértice inicial arbitrario
          y ← x
        }
     en otro caso
        si (d+(v) = d-(v), ∀v∈(V-{s, t})) y
           (d+(s) = d-(s) + 1) y (d-(t) = d+(t) + 1)
        entonces
           { Existe un camino entre los vértices s y t
             x ← s
             y ← t
           }
     en otro caso
        { No existe ni circuito ni camino euleriano
          PARAR
        }
   }
```

```
en otro caso
   { si (d(v) módulo 2) = 0, ∀v∈V entonces
        { Existe un ciclo
          x ← Elegir un vértice inicial arbitrario
          y ← x
        }
     en otro caso
        si (d(v) módulo 2) = 0, ∀v∈(V − {s, t})
        entonces
           { Existe una cadena entre los vértices s y t
             x ← s
             y ← t
           }
     en otro caso
        { No existe ni ciclo ni cadena euleriana
          PARAR
        }
   }
Aristas ← 0
mientras (x ≠ y) o (Aristas ≠ |E|) hacer
   { Elegir un sucesor s del vértice x de forma que
     G − {(x, s)} sea conexo
     G ← G − {(x, s)}
     Aristas ← Aristas + 1
     x ← s
   }
```

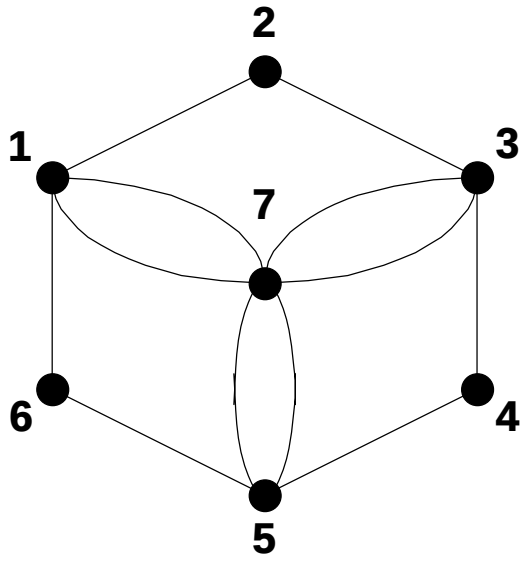


Figura 2: Multigrafo no dirigido.

Recorrido	Nº de Aristas	Vértice Actual	Sucesor	¿Grafo conexo?
{1}	0	1	2	SI
{1, 2}	1	2	3	SI
{1, 2, 3}	2	3	4	SI
{1, 2, 3, 4}	3	4	5	SI
{1, 2, 3, 4, 5}	4	5	6	SI
{1, 2, 3, 4, 5, 6}	5	6	1	SI
{1, 2, 3, 4, 5, 6, 1}	6	1	7	SI
{1, 2, 3, 4, 5, 6, 1, 7}	7	7	1	NO
{1, 2, 3, 4, 5, 6, 1, 7}	7	7	3	SI
{1, 2, 3, 4, 5, 6, 1, 7, 3}	8	3	7	SI
{1, 2, 3, 4, 5, 6, 1, 7, 3, 7}	9	7	1	NO
{1, 2, 3, 4, 5, 6, 1, 7, 3, 7}	9	7	5	SI
{1, 2, 3, 4, 5, 6, 1, 7, 3, 7, 5}	10	5	7	SI
{1, 2, 3, 4, 5, 6, 1, 7, 3, 7, 5, 7}	11	7	1	SI
{1, 2, 3, 4, 5, 6, 1, 7, 3, 7, 5, 7, 1}	12	—	—	—

Tabla 1: Traza para calcular el circuito euleriano del grafo anterior.

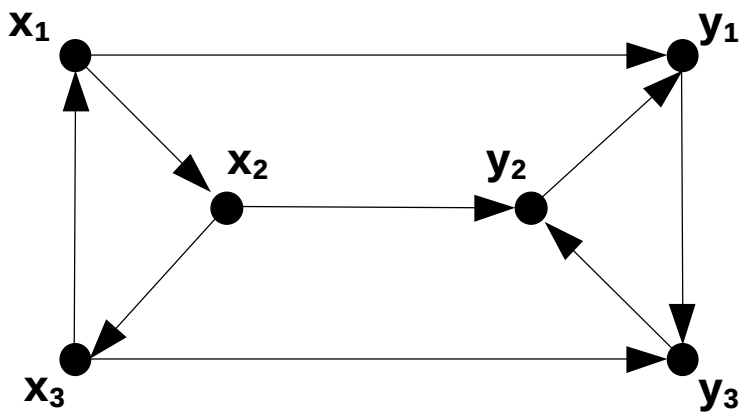


Figura 3: Grafo sin circuito euleriano.

Algoritmo del Cartero Chino Asimétrico

Construir el grafo G' añadiendo al grafo original G los vértices fuente X y sumidero Y.
Encontrar un flujo máximo de costo mínimo en G'.
Construir G' mediante la repetición de aristas del grafo G.
Encontrar un circuito euleriano en G', y con ello, un circuito del cartero de mínimo coste en G.

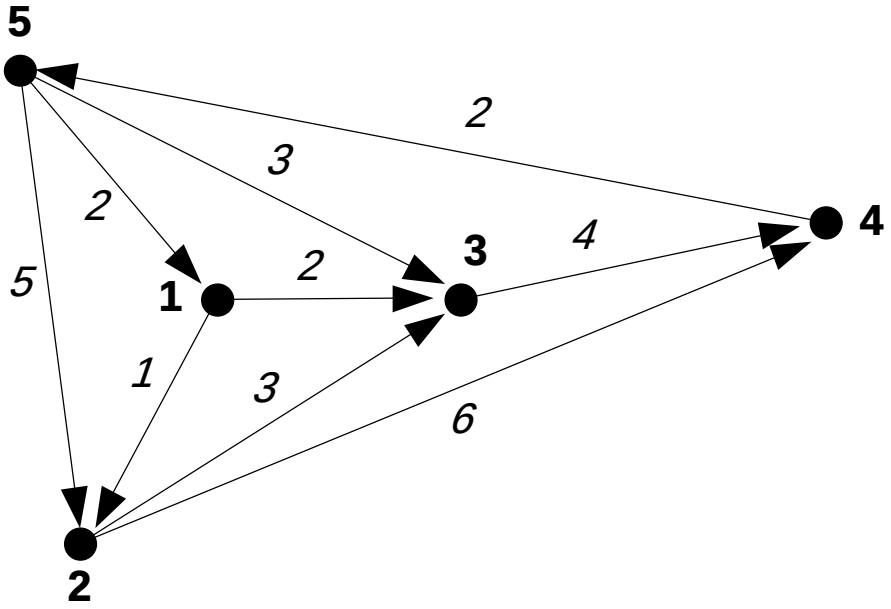


Figura 4: Grafo dirigido y no balanceado.

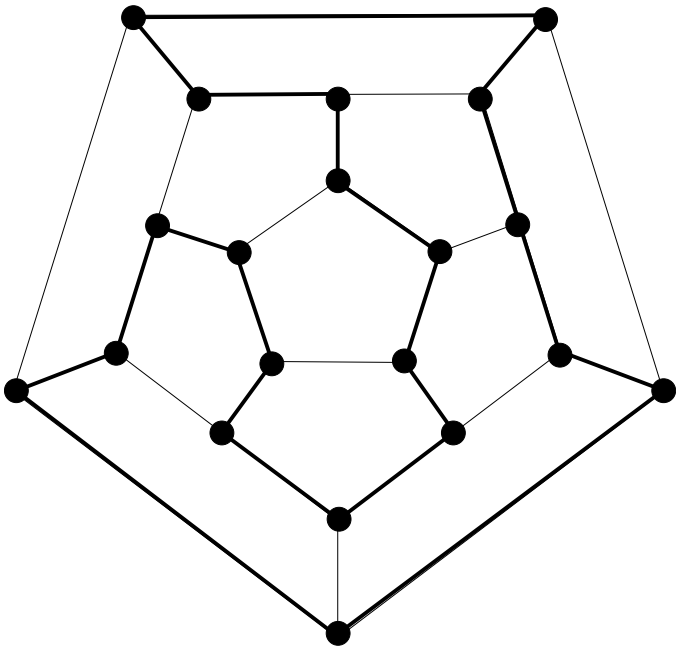


Figura 5: Circuito alrededor del mundo.

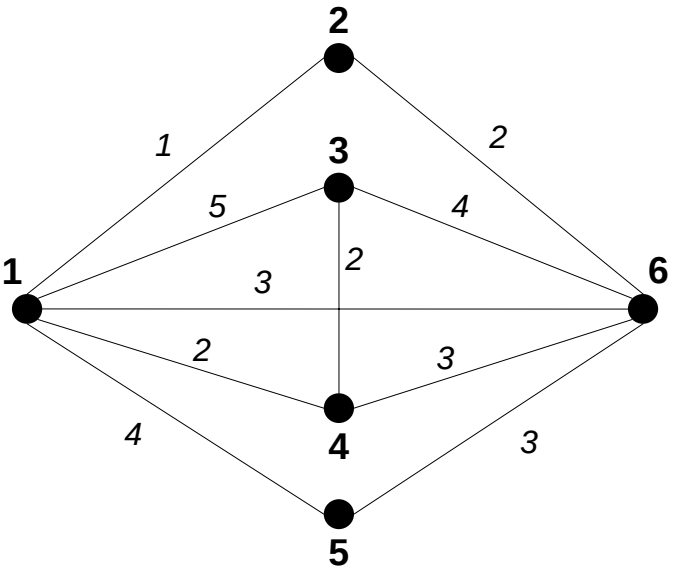
Algoritmo del Cartero Chino Simétrico

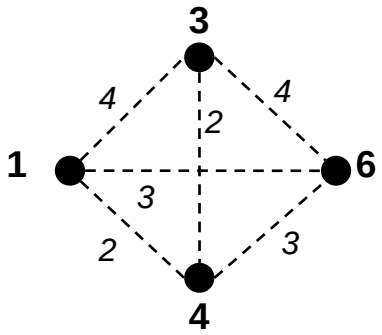
Construir grafo $G'=(V',E')$ con todos los vértices

$$V'=\{v\in V : d(v) \bmod 2 \neq 0\}, \text{ con } |V'| \text{ par}$$

$$E'=\{\text{Camino MIN } d(v_i,v_j) : v_i,v_j\in V'\}$$

Encontrar un Matching de MIN costo en G'.
Repetir las aristas de los caminos MIN del matching anterior.
Encontrar un circuito euleriano en G, y con ello, un circuito del cartero de mínimo coste en G.





Costo total = 29

V' = {1, 3, 4, 6}

Matching	Costo
(1,3) – (4,6)	4 + 3 = 7
(1,4) – (3,6)	2 + 4 = 6
(1,6) – (3,4)	3 + 2 = 5

Repetir las aristas de los Caminos Mínimos (1,6) y (3,4)

Costo del circuito del cartero: 29 + 5 = 34

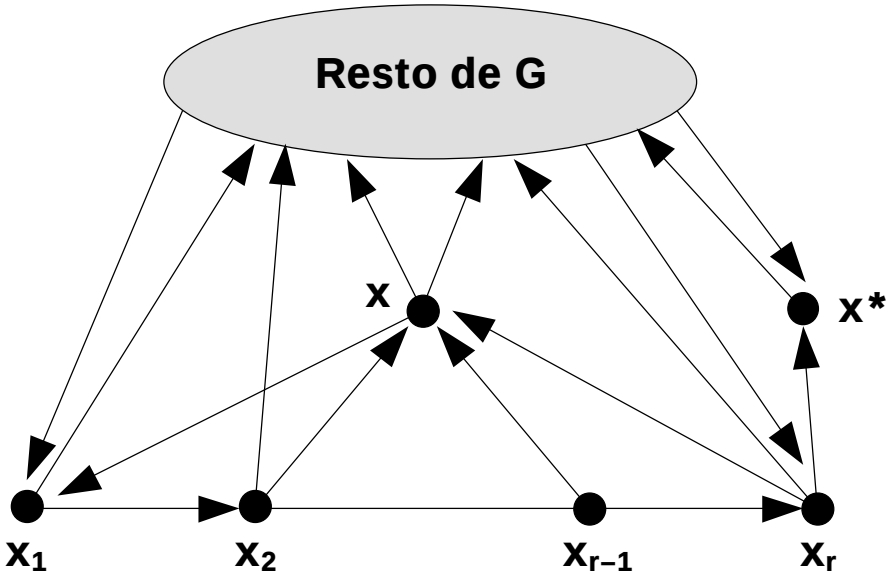


Figura 6: Mejora a) del método.

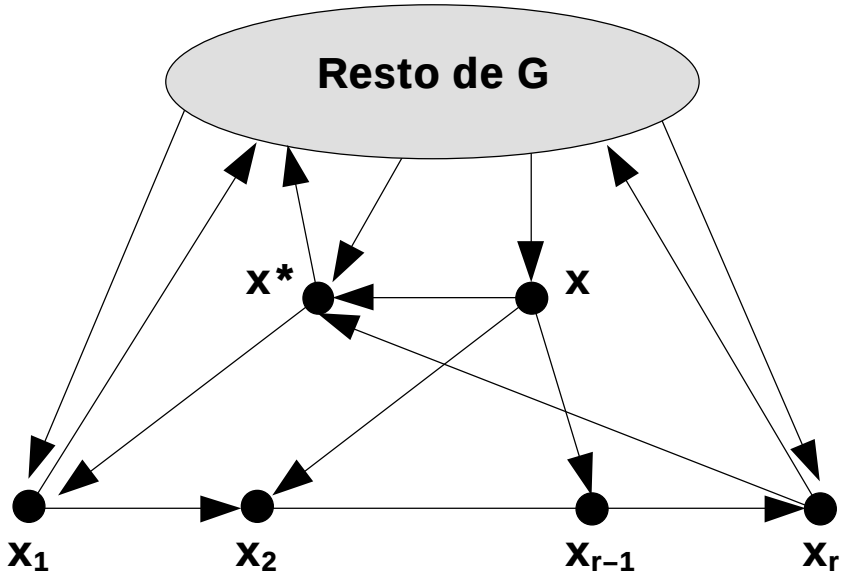


Figura 7: Mejora b) del método.

Algoritmo

```
Construir la matriz M_KxN = [m_ij]
Escoger un vértice inicial x_1
S ← S ∪ {x_1}
Marcado[x_1] ← TRUE
x ← x_1
mientras S ≠ ∅ hacer
   { mientras (exista algún elemento y en la columna x)
                y (|S| ≠ N) hacer
         { si Marcado[y] = FALSE entonces
              { S ← S ∪ {y}
                Marcado[y] ← TRUE
                x ← y
              }
         }
     si |S| = N entonces
         { si existe una arista del tipo (x, x_1) entonces
              { S ← S ∪ {x_1}
                Mostrar S en la salida
                si sólo se desea un circuito
                   hamiltoniano entonces
                   PARAR
              }
         }
     Siendo S = {x_1, x_2,..., x_r-1, x_r} hacer S ← S − {x_r}
     Marcado[x_r] ← FALSE
     x ← x_r-1
   }
```

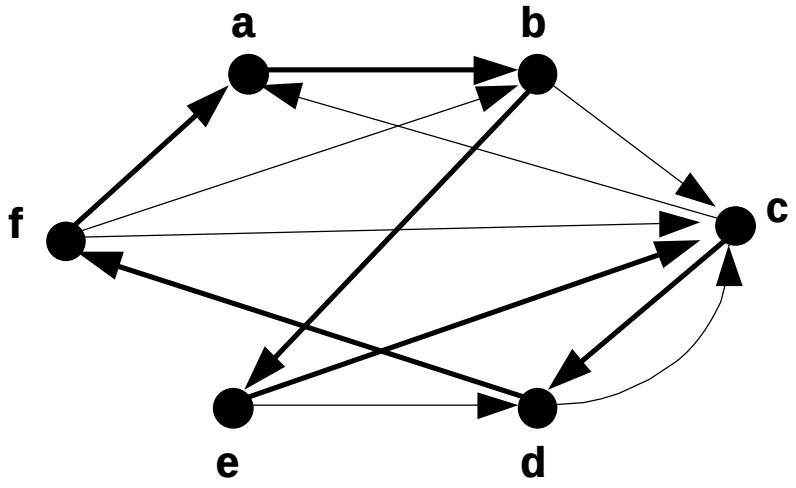


Figura 8: Grafo dirigido y su circuito hamiltoniano.

	a	b	c	d	e	f
1	b	c	a	c	c	a
2	—	e	d	f	d	b
3	—	—	—	—	—	c

Tabla 2: Matriz M del grafo anterior.

S	Vértice actual x	Siguiente vértice y	Marcado[y]
∅	—	a	FALSE
{a}	a	b	FALSE
{a, b}	b	c	FALSE
{a, b, c}	c	a	TRUE
{a, b, c}	c	d	FALSE
{a, b, c, d}	d	c	TRUE
{a, b, c, d}	d	f	FALSE
{a, b, c, d, f}	f	a, b, c	TRUE
{a, b, c, d}	d	—	—
{a, b, c}	c	—	—
{a, b}	b	e	FALSE
{a, b, e}	e	c	FALSE
{a, b, e, c}	c	a	TRUE
{a, b, e, c}	c	d	FALSE
{a, b, e, c, d}	d	c	TRUE
{a, b, e, c, d}	d	f	FALSE
{a, b, e, c, d, f}	f	—	—
{a, b, e, c, d, f, a}	—	—	—

Tabla 3: Traza del grafo anterior.

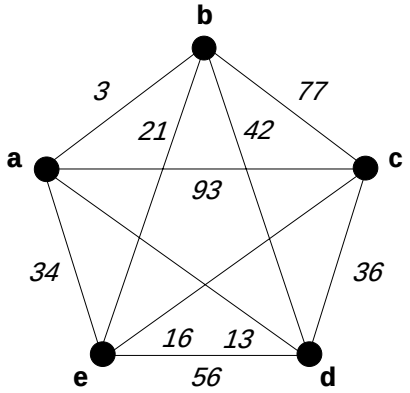


Figura 9: El TSP para cinco ciudades.

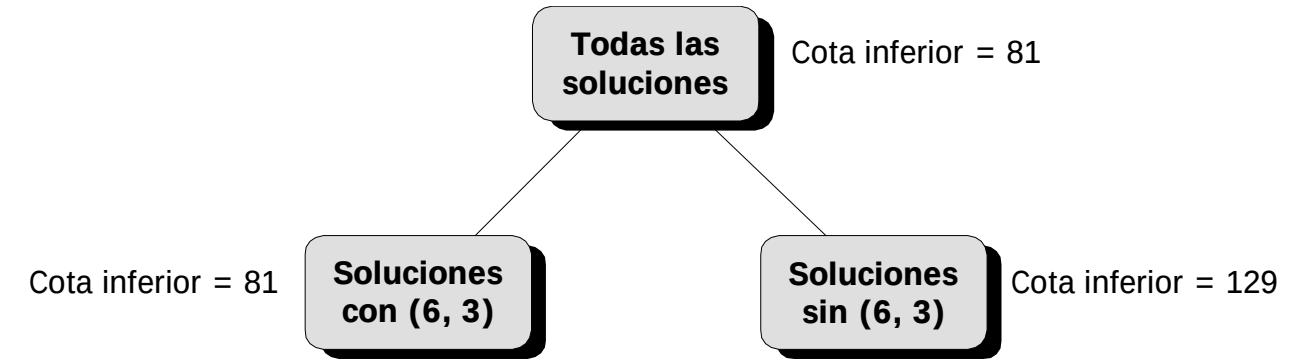
	1	2	3	4	5	6
1	∞	3	93	13	33	9
2	4	∞	77	42	21	16
3	45	17	∞	36	16	28
4	39	90	80	∞	56	7
5	28	46	88	33	∞	25
6	3	88	18	46	92	∞

(a)

	1	2	3	4	5	6
1	∞	0	75	2	30	6
2	0	∞	58	30	17	12
3	29	1	∞	12	0	12
4	32	83	58	∞	49	0
5	3	21	48	0	∞	0
6	0	85	0	35	89	∞

(b)

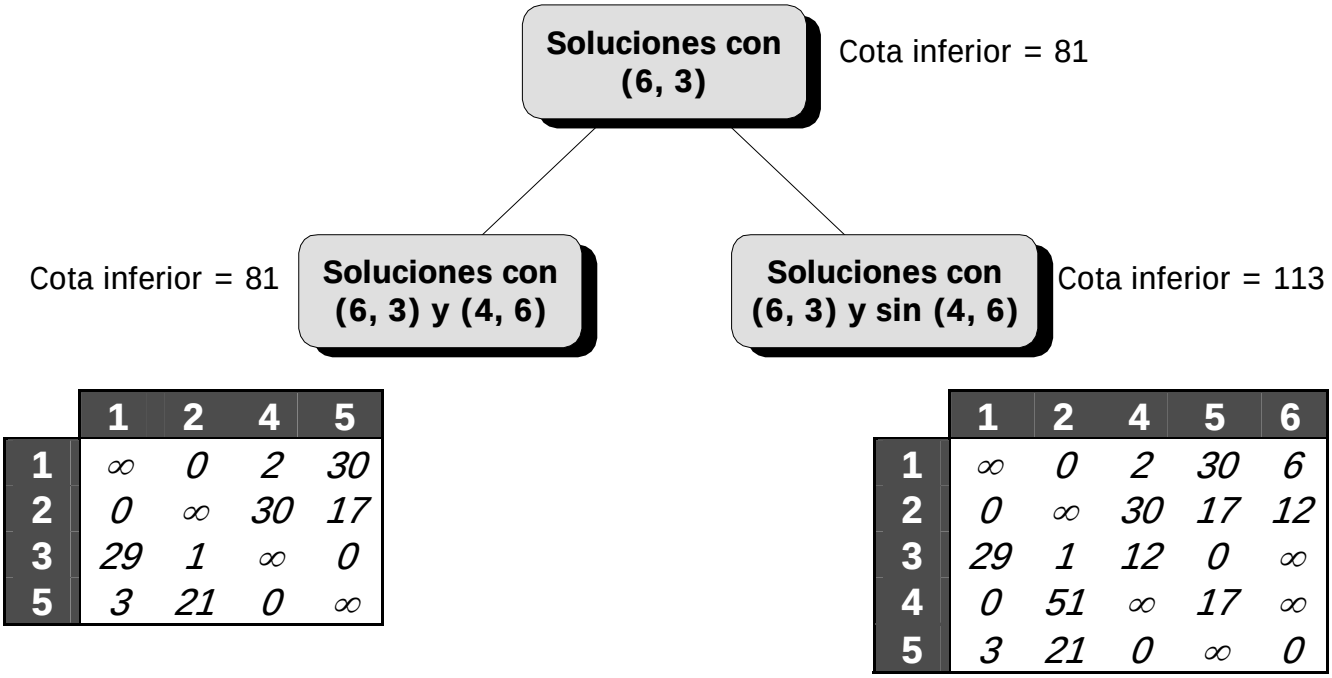
Tabla 4: (a) Matriz de costos inicial; (b) Matriz reducida.



	1	2	4	5	6
1	∞	0	2	30	6
2	0	∞	30	17	12
3	29	1	12	0	∞
4	32	83	∞	49	0
5	3	21	0	∞	0

	1	2	3	4	5	6
1	∞	0	27	2	30	6
2	0	∞	10	30	17	12
3	29	1	∞	12	0	12
4	32	83	10	∞	49	0
5	3	21	0	0	∞	0
6	0	85	∞	35	89	∞

Figura 10: División de las soluciones.



	1	2	4	5
1	∞	0	2	30
2	0	∞	30	17
3	29	1	∞	0
5	3	21	0	∞

	1	2	4	5	6
1	∞	0	2	30	6
2	0	∞	30	17	12
3	29	1	12	0	∞
4	0	51	∞	17	∞
5	3	21	0	∞	0

Figura 11: Árbol parcial con soluciones.

	2	4	5
1	∞	2	30
3	1	∞	0
5	21	0	∞

	2	4	5
1	∞	0	28
3	0	∞	0
5	20	0	∞

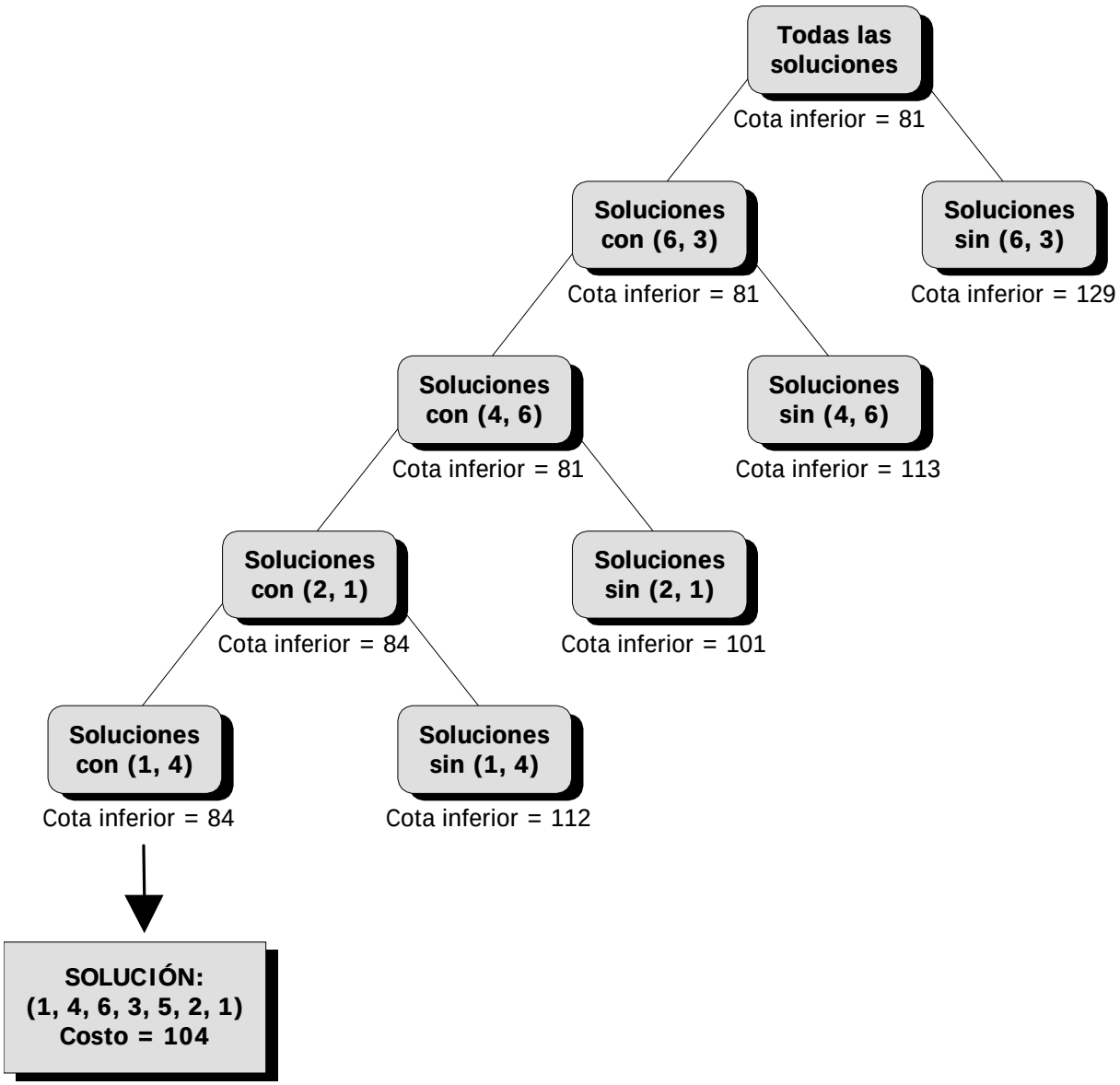


Figura 12: Árbol binario de ramificación y acotación.

	1	2	4	5
1	∞	0	2	30
2	∞	∞	13	0
3	26	1	∞	0
5	0	21	0	∞

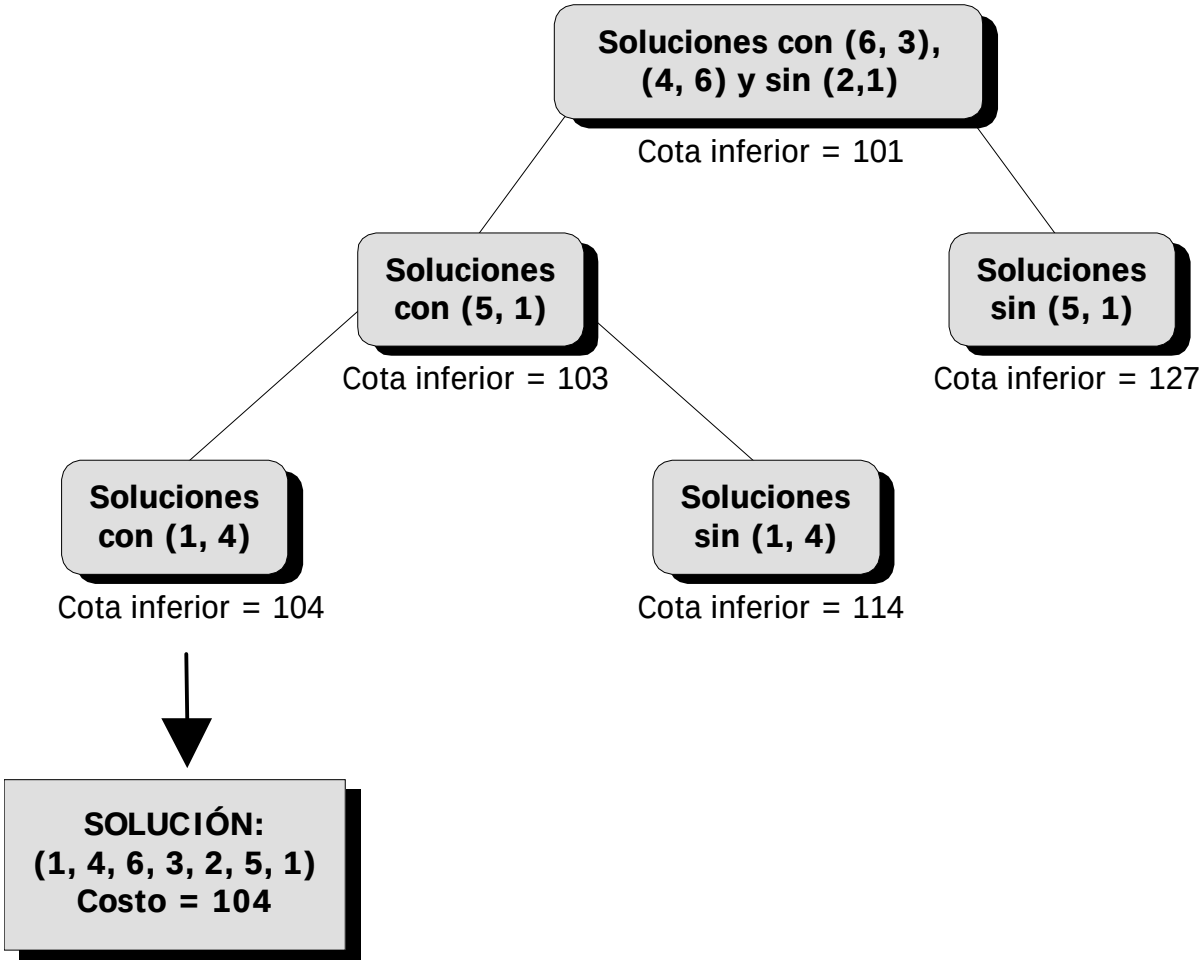


Figura 13: Expansión del nodo derecho con cota 101.

	w	x
u	0	∞
v	∞	0

	w	x
u	∞	0
v	0	∞

```
si A[1, 1] = ∞ entonces
   Añadir las aristas (u, x) y (v, w) al camino actual
en caso contrario
   Añadir las aristas (u, w) y (v, x) al camino actual
```

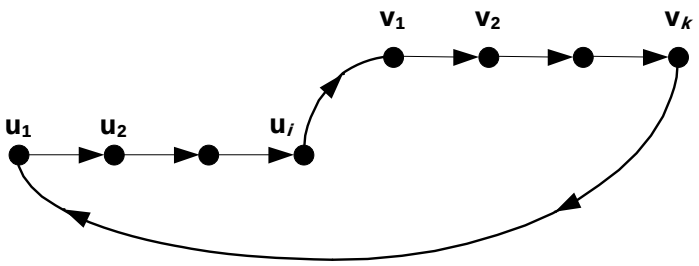


Figura 14: Ciclo corto.

Algoritmo

```
Costo_Min ← ∞

procedimiento TSP(Aristas, Costo, A)
{ Tamaño_A ← N − Aristas
  Valor_Red ← 0
  para i ← 1 hasta Tamaño_A hacer
     { Red_Fila[i] ← Elemento más pequeño en la fila i
       si Red_Fila[i] > 0 entonces
          { Sustraer Red_Fila[i] de cada elemento
            finito de la fila i
            Valor_Red ← Valor_Red + Red_Fila[i]
          }
     }

  para j ← 1 hasta Tamaño_A hacer
     { Red_Col[j] ← Elemento mínimo en la columna j
         si Red_Col[j] > 0 entonces
            { Sustraer Red_Col[j] de cada elemento
              finito de la columna j
              Valor_Red ← Valor_Red + Red_Col[j]
            }
     }

  Costo ← Costo + Valor_Red
  si Costo < Costo_Min entonces
     { si Aristas = (N − 2) entonces
          { Añadir las dos últimas aristas
            Costo_Min ← Costo
            Almacenar la nueva solución
          }
```

```
        en otro caso
            { Max ← −∞
              para i ← 1 hasta Tamaño_A hacer
                 { para j ← 1 hasta Tamaño_A hacer
                      { si a_ij = 0 entonces
                           { Min_Fila ← Mínimo en la fila
                                        i, distinto a a_ij
                             Min_Col ← Mínimo en la
                                 columna j, distinto a a_ij
                             Total ← Min_Fila + Min_Col
                             si Total > Max entonces
                                { Max ← Total
                                  f ← i
                                  c ← j
                                }
                           }
                      }
                 }
              Cota_Inferior ← Costo + Max
              Prevenir ciclos cortos
              Nueva_A ← A − fila f − columna c
              TSP(Aristas + 1, Costo, Nueva_A)
              Restaurar A añadiendo las fila f y
              la columna c
              si Cota_Inferior < Costo_Min entonces
                 { a_fc ← ∞
                   TSP(Aristas, Costo, A)
                   a_fc ← 0
                 }
            }
      }
  Volver a añadir a la matriz los valor de los vectores
  Fila_Red y Col_Red
}
```

Algoritmo de la inserción más alejada

$V_T \leftarrow \{s\}$
$E_T \leftarrow \{(s, s)\}$
$w_{ss} \leftarrow 0$
Costo_Total $\leftarrow 0$
para cada nodo $u \in (V - V_T)$ **hacer**
  Dist(u) $\leftarrow w_{su}$
mientras $|V_T| < N$ **hacer**
  { f $\leftarrow$ nodo en $V - V_T$ con el mayor valor de Dist(f)

NOTA: Para la inserción más cercana cambiar en el paso anterior mayor por menor

  para cada arista $(i, j) \in E_T$ **hacer**
    $c_{ij} \leftarrow w_{if} + w_{fj} - w_{ij}$
  $(t, h) \leftarrow$ arista en E_T con el menor valor de c_{th}
  $E_T \leftarrow E_T \cup \{(t, f), (f, h)\} - \{(t, h)\}$
  $V_T \leftarrow V_T \cup \{f\}$
  Costo_Total $\leftarrow$ Costo_Total $+ c_{th}$
  para todos los $x \in (V - V_T)$ **hacer**
    Dist(x) $\leftarrow \min\{Dist(x), w_{fx}\}$
}

Iter.	f	c_{ij}	Tour	Costo Total	Dist	Ciclo
Inicio	1	—	(1, 1)	0	(–, 3, 93, 13, 33, 9)	(1, 0, 0, 0, 0, 0)
1	3	c_{11}=138*	(1, 3, 1)	138	(–, 3, –, 13, 16, 9)	(3, 0, 1, 0, 0, 0)
2	5	c_{13}=28 c_{31}=–1*	(1, 3, 5, 1)	137	(–, 3, –, 13, –, 9)	(3, 0, 5, 0, 1, 0)
3	4	c_{13}=0* c_{35}=76 c_{51}=44	(1, 4, 3, 5, 1)	137	(–, 3, –, –, –, 7)	(4, 0, 5, 3, 1, 0)
4	6	c_{14}=42 c_{43}=–55* c_{35}=104 c_{51}=0	(1, 4, 6, 3, 5, 1)	82	(–, 3, –, –, –, –)	(4, 0, 5, 6, 1, 3)
5	2	c_{14}=32 c_{46}=99 c_{63}=147 c_{35}=22* c_{51}=22*	(1, 4, 6, 3, 5, 2, 1)	104	(–, –, –, –, –, –)	(4, 1, 5, 6, 2, 3)

Tabla 5: Inserción más lejana sobre el problema de las seis ciudades.

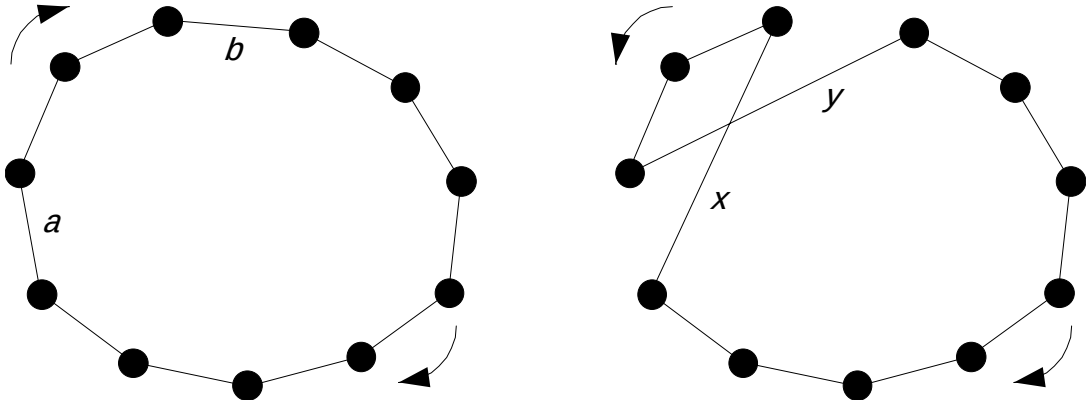


Figura 15: Un 2–intercambio.

Algoritmo de la 2–optimalidad

```
Calcular un tour inicial, por ejemplo, con la
heurística de la inserción más alejada
Sea H ← {x1, x2,...,xN} el conjunto de aristas del tour
inicial
Sea Costo_Total el costo del tour inicial
repetir
{ δmax ← 0
  para i ← 1 hasta (N - 2) hacer
     { si i = 1 entonces Límite ← N - 1
       en otro caso      Límite ← N
       para j ← (i + 2) hasta Límite hacer
          { δ ← (w(xi) + w(xj)) - (w(yp) + w(yq))
            si δ > δmax entonces
               { δmax ← δ
                 Guardar las aristas xi, xj, yp, yq
               }
          }
     }
  si δmax > 0 entonces
     { H ← H - {xi, xj} ∪ {yp, yq}
       Costo_Total ← Costo_Total - δmax
     }
} hasta que δmax = 0
```

(i, j)	1	2	3	4	5	6	7	8	9	10
1	–	10	55	78	93	79	48	49	76	11
2	10	–	77	46	64	86	37	8	29	35
3	55	77	–	79	9	82	13	54	80	84
4	78	46	79	–	34	29	9	35	98	91
5	93	64	9	34	–	74	83	44	67	68
6	79	86	82	29	74	–	42	98	85	8
7	48	37	13	9	83	42	–	84	5	2
8	49	8	54	35	44	98	84	–	98	28
9	76	29	80	98	67	85	5	98	–	48
10	11	35	84	91	68	8	2	28	48	–

Tabla 6: Matriz de costos W del ejemplo.

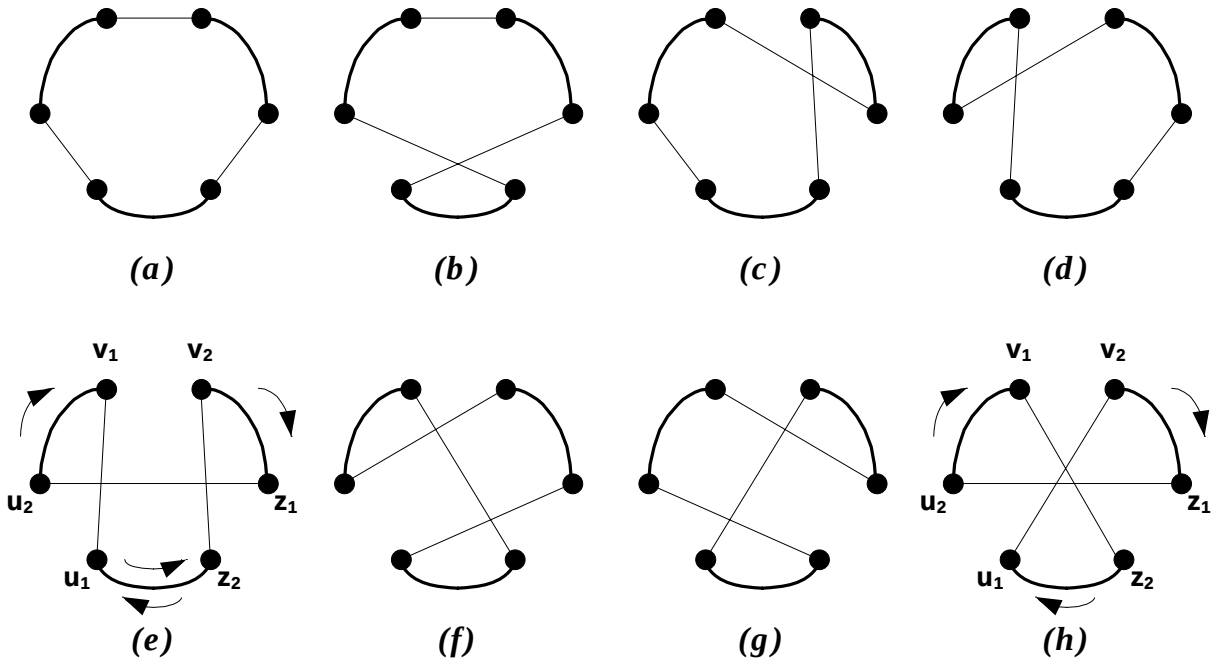


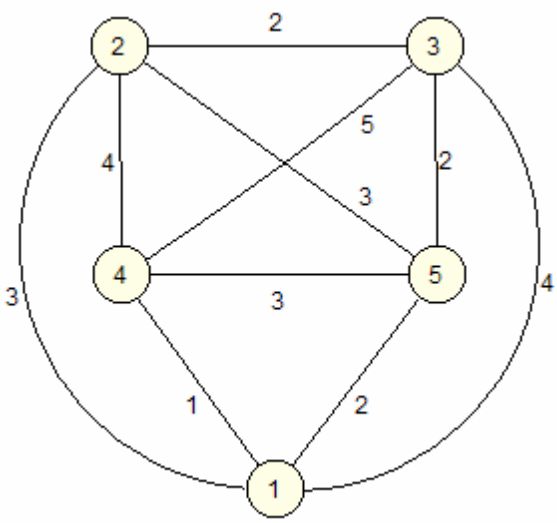
Figura 16: Todos los posibles intercambios de tres caminos. Líneas = caminos, Puntos = nuevas aristas.

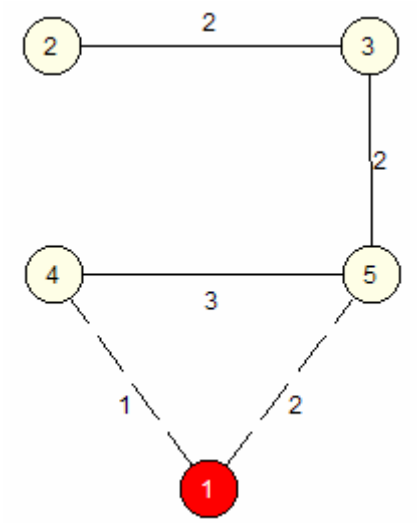
Algoritmo de la 3–optimalidad

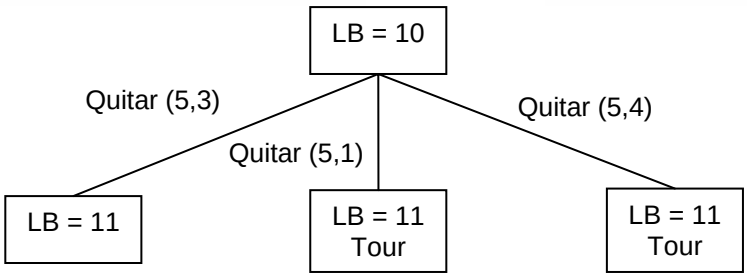
```
Calcular un tour inicial, por ejemplo, con la
heurística de la inserción más alejada
Sea H ← {x1, x2,...,xN} el conjunto de aristas del tour
inicial
Sea Costo_Total el costo del tour inicial
repetir
{ δmax ← 0
  para i ← 1 hasta N hacer
     { para j ← 2 hasta (N − 3) hacer
          { para k ← (j + 2) hasta (N − 1) hacer
               { Calcular para el caso e):
                    δe ← (w(xi) + w(xj) + w(xk)) −
                         (w(yp) + w(yq) + w(yr))
                 si δe > δmax entonces
                    { δmax ← δe
                      Guardar las aristas xi, xj, xk,
                                          yp, yq, yr
                    }
                 Calcular para el caso h):
                    δh ← (w(xi) + w(xj) + w(xk)) −
                         (w(ys) + w(yt) + w(yu))
                 si δh > δmax entonces
                    { δmax ← δh
                      Guardar las aristas xi, xj, xk,
                                          ys, yt, yu
                    }
               }
          }
     }
  si δmax > 0 entonces
     { H ← H − {xi, xj, xk}
       si es el caso e) entonces H ← H ∪ {yp, yq, yr}
       en otro caso               H ← H ∪ {ys, yt, yu}
       Costo_Total ← Costo_Total − δmax
    }
} hasta que δmax = 0
```

Algoritmo del TSP simétrico

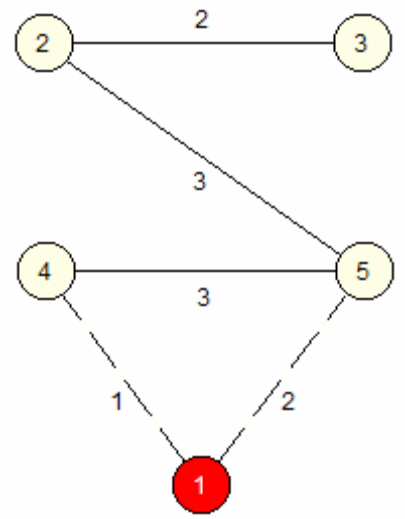
Buscar un cota inferior (LB) quitando un vértice x del grafo y calculando un árbol generador de mínimo coste (MST) con las n-2 aristas restantes.
Añadir las 2 aristas de menor coste del vértice x. El costo total es una LB.
Si todos los nodos tienen grado 2 => Tour encontrado.
Si algún nodo tiene grado > 2, entonces alguna de sus aristas debe ser eliminada. Se ramifica quitando cada una de esas aristas y obteniendo nuevas LB.
Volver a calcular el MST sin nodo x y sin la arista considerada, hasta que todas las LB >= valor óptimo.



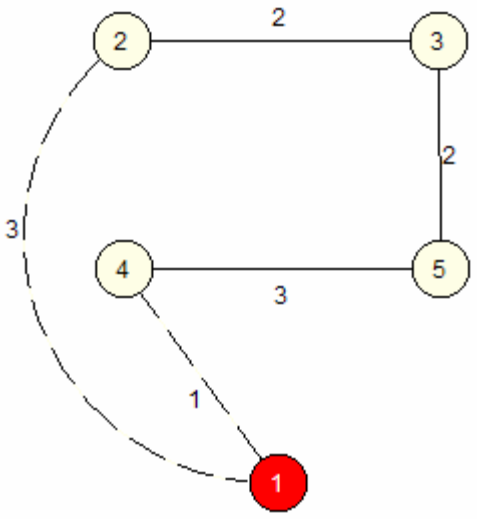




a) Quitar (5,3)



b) Quitar (5,1)



c) Quitar (5,4)