Branch: master **03MAIR-Algoritmos-de-optimizacion** / AG3 / **Isabel_Vazquez_AG3.ipynb**

Find file Copy path

404isabel Creado con Colaboratory e7f414c 3 minutes ago

1 contributor

1001 lines (1001 sloc) 97.1 KB

Raw Blame History

Open in Colab

AG-Actividad Guiada 3

Nombre: Isabel Vázquez Trigás

https://github.com/404isabel/03MAIR-Algoritmos-de-optimizacion/tree/master/AG3

In [94]:
```
import urllib.request
file="swiss42.tsp"
urllib.request.urlretrieve("http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/swiss42.tsp",file
)
```

Out[94]:
```
('swiss42.tsp', <http.client.HTTPMessage at 0x7fce62cccc18>)
```

In [95]:
```
!pip install tsplib95
```

```
Requirement already satisfied: tsplib95 in /usr/local/lib/python3.6/dist-packages (0.3.2)
Requirement already satisfied: Click>=6.0 in /usr/local/lib/python3.6/dist-packages (from tsplib
95) (7.0)
Requirement already satisfied: networkx==2.1 in /usr/local/lib/python3.6/dist-packages (from tsp
lib95) (2.1)
Requirement already satisfied: decorator>=4.1.0 in /usr/local/lib/python3.6/dist-packages (from
networkx==2.1->tsplib95) (4.3.2)
```

In [0]:
```
import tsplib95
import random
from math import e

problem = tsplib95.load_problem(file)

#Nodos
Nodos = list(problem.get_nodes())

#Aristas
Aristas = list(problem.get_edges())

#print("Nodos",Nodos)
```

```
In [103]: print("Nodos",Nodos)
          print("Aristas",Aristas)
```

```
Nodos [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41]
Aristas [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4), (0, 5), (0, 6), (0, 7), (0, 8), (0, 9), (0, 1
0), (0, 11), (0, 12), (0, 13), (0, 14), (0, 15), (0, 16), (0, 17), (0, 18), (0, 19), (0, 20),
(0, 21), (0, 22), (0, 23), (0, 24), (0, 25), (0, 26), (0, 27), (0, 28), (0, 29), (0, 30), (0, 3
1), (0, 32), (0, 33), (0, 34), (0, 35), (0, 36), (0, 37), (0, 38), (0, 39), (0, 40), (0, 41),
(1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (1, 8), (1, 9), (1, 10), (1, 1
1), (1, 12), (1, 13), (1, 14), (1, 15), (1, 16), (1, 17), (1, 18), (1, 19), (1, 20), (1, 21),
(1, 22), (1, 23), (1, 24), (1, 25), (1, 26), (1, 27), (1, 28), (1, 29), (1, 30), (1, 31), (1, 3
2), (1, 33), (1, 34), (1, 35), (1, 36), (1, 37), (1, 38), (1, 39), (1, 40), (1, 41), (2, 0), (2,
1), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (2, 8), (2, 9), (2, 10), (2, 11), (2, 12),
(2, 13), (2, 14), (2, 15), (2, 16), (2, 17), (2, 18), (2, 19), (2, 20), (2, 21), (2, 22), (2, 2
3), (2, 24), (2, 25), (2, 26), (2, 27), (2, 28), (2, 29), (2, 30), (2, 31), (2, 32), (2, 33),
(2, 34), (2, 35), (2, 36), (2, 37), (2, 38), (2, 39), (2, 40), (2, 41), (3, 0), (3, 1), (3, 2),
(3, 3), (3, 4), (3, 5), (3, 6), (3, 7), (3, 8), (3, 9), (3, 10), (3, 11), (3, 12), (3, 13), (3,
14), (3, 15), (3, 16), (3, 17), (3, 18), (3, 19), (3, 20), (3, 21), (3, 22), (3, 23), (3, 24),
(3, 25), (3, 26), (3, 27), (3, 28), (3, 29), (3, 30), (3, 31), (3, 32), (3, 33), (3, 34), (3, 3
5), (3, 36), (3, 37), (3, 38), (3, 39), (3, 40), (3, 41), (4, 0), (4, 1), (4, 2), (4, 3), (4,
4), (4, 5), (4, 6), (4, 7), (4, 8), (4, 9), (4, 10), (4, 11), (4, 12), (4, 13), (4, 14), (4, 1
5), (4, 16), (4, 17), (4, 18), (4, 19), (4, 20), (4, 21), (4, 22), (4, 23), (4, 24), (4, 25),
(4, 26), (4, 27), (4, 28), (4, 29), (4, 30), (4, 31), (4, 32), (4, 33), (4, 34), (4, 35), (4, 3
6), (4, 37), (4, 38), (4, 39), (4, 40), (4, 41), (5, 0), (5, 1), (5, 2), (5, 3), (5, 4), (5, 5),
(5, 6), (5, 7), (5, 8), (5, 9), (5, 10), (5, 11), (5, 12), (5, 13), (5, 14), (5, 15), (5, 16),
(5, 17), (5, 18), (5, 19), (5, 20), (5, 21), (5, 22), (5, 23), (5, 24), (5, 25), (5, 26), (5, 2
7), (5, 28), (5, 29), (5, 30), (5, 31), (5, 32), (5, 33), (5, 34), (5, 35), (5, 36), (5, 37),
(5, 38), (5, 39), (5, 40), (5, 41), (6, 0), (6, 1), (6, 2), (6, 3), (6, 4), (6, 5), (6, 6), (6,
7), (6, 8), (6, 9), (6, 10), (6, 11), (6, 12), (6, 13), (6, 14), (6, 15), (6, 16), (6, 17), (6,
18), (6, 19), (6, 20), (6, 21), (6, 22), (6, 23), (6, 24), (6, 25), (6, 26), (6, 27), (6, 28),
(6, 29), (6, 30), (6, 31), (6, 32), (6, 33), (6, 34), (6, 35), (6, 36), (6, 37), (6, 38), (6, 3
9), (6, 40), (6, 41), (7, 0), (7, 1), (7, 2), (7, 3), (7, 4), (7, 5), (7, 6), (7, 7), (7, 8),
(7, 9), (7, 10), (7, 11), (7, 12), (7, 13), (7, 14), (7, 15), (7, 16), (7, 17), (7, 18), (7, 1
9), (7, 20), (7, 21), (7, 22), (7, 23), (7, 24), (7, 25), (7, 26), (7, 27), (7, 28), (7, 29),
(7, 30), (7, 31), (7, 32), (7, 33), (7, 34), (7, 35), (7, 36), (7, 37), (7, 38), (7, 39), (7, 4
0), (7, 41), (8, 0), (8, 1), (8, 2), (8, 3), (8, 4), (8, 5), (8, 6), (8, 7), (8, 8), (8, 9), (8,
10), (8, 11), (8, 12), (8, 13), (8, 14), (8, 15), (8, 16), (8, 17), (8, 18), (8, 19), (8, 20),
(8, 21), (8, 22), (8, 23), (8, 24), (8, 25), (8, 26), (8, 27), (8, 28), (8, 29), (8, 30), (8, 3
```

```
1), (8, 32), (8, 33), (8, 34), (8, 35), (8, 36), (8, 37), (8, 38), (8, 39), (8, 40), (8, 41),
(9, 0), (9, 1), (9, 2), (9, 3), (9, 4), (9, 5), (9, 6), (9, 7), (9, 8), (9, 9), (9, 10), (9, 1
1), (9, 12), (9, 13), (9, 14), (9, 15), (9, 16), (9, 17), (9, 18), (9, 19), (9, 20), (9, 21),
(9, 22), (9, 23), (9, 24), (9, 25), (9, 26), (9, 27), (9, 28), (9, 29), (9, 30), (9, 31), (9, 3
2), (9, 33), (9, 34), (9, 35), (9, 36), (9, 37), (9, 38), (9, 39), (9, 40), (9, 41), (10, 0), (1
0, 1), (10, 2), (10, 3), (10, 4), (10, 5), (10, 6), (10, 7), (10, 8), (10, 9), (10, 10), (10, 1
1), (10, 12), (10, 13), (10, 14), (10, 15), (10, 16), (10, 17), (10, 18), (10, 19), (10, 20), (1
0, 21), (10, 22), (10, 23), (10, 24), (10, 25), (10, 26), (10, 27), (10, 28), (10, 29), (10, 3
0), (10, 31), (10, 32), (10, 33), (10, 34), (10, 35), (10, 36), (10, 37), (10, 38), (10, 39), (1
0, 40), (10, 41), (11, 0), (11, 1), (11, 2), (11, 3), (11, 4), (11, 5), (11, 6), (11, 7), (11,
8), (11, 9), (11, 10), (11, 11), (11, 12), (11, 13), (11, 14), (11, 15), (11, 16), (11, 17), (1
1, 18), (11, 19), (11, 20), (11, 21), (11, 22), (11, 23), (11, 24), (11, 25), (11, 26), (11, 2
7), (11, 28), (11, 29), (11, 30), (11, 31), (11, 32), (11, 33), (11, 34), (11, 35), (11, 36), (1
1, 37), (11, 38), (11, 39), (11, 40), (11, 41), (12, 0), (12, 1), (12, 2), (12, 3), (12, 4), (1
2, 5), (12, 6), (12, 7), (12, 8), (12, 9), (12, 10), (12, 11), (12, 12), (12, 13), (12, 14), (1
2, 15), (12, 16), (12, 17), (12, 18), (12, 19), (12, 20), (12, 21), (12, 22), (12, 23), (12, 2
4), (12, 25), (12, 26), (12, 27), (12, 28), (12, 29), (12, 30), (12, 31), (12, 32), (12, 33), (1
2, 34), (12, 35), (12, 36), (12, 37), (12, 38), (12, 39), (12, 40), (12, 41), (13, 0), (13, 1),
(13, 2), (13, 3), (13, 4), (13, 5), (13, 6), (13, 7), (13, 8), (13, 9), (13, 10), (13, 11), (13,
12), (13, 13), (13, 14), (13, 15), (13, 16), (13, 17), (13, 18), (13, 19), (13, 20), (13, 21),
(13, 22), (13, 23), (13, 24), (13, 25), (13, 26), (13, 27), (13, 28), (13, 29), (13, 30), (13, 3
1), (13, 32), (13, 33), (13, 34), (13, 35), (13, 36), (13, 37), (13, 38), (13, 39), (13, 40), (1
3, 41), (14, 0), (14, 1), (14, 2), (14, 3), (14, 4), (14, 5), (14, 6), (14, 7), (14, 8), (14,
9), (14, 10), (14, 11), (14, 12), (14, 13), (14, 14), (14, 15), (14, 16), (14, 17), (14, 18), (1
4, 19), (14, 20), (14, 21), (14, 22), (14, 23), (14, 24), (14, 25), (14, 26), (14, 27), (14, 2
8), (14, 29), (14, 30), (14, 31), (14, 32), (14, 33), (14, 34), (14, 35), (14, 36), (14, 37), (1
4, 38), (14, 39), (14, 40), (14, 41), (15, 0), (15, 1), (15, 2), (15, 3), (15, 4), (15, 5), (15,
6), (15, 7), (15, 8), (15, 9), (15, 10), (15, 11), (15, 12), (15, 13), (15, 14), (15, 15), (15,
16), (15, 17), (15, 18), (15, 19), (15, 20), (15, 21), (15, 22), (15, 23), (15, 24), (15, 25),
(15, 26), (15, 27), (15, 28), (15, 29), (15, 30), (15, 31), (15, 32), (15, 33), (15, 34), (15, 3
5), (15, 36), (15, 37), (15, 38), (15, 39), (15, 40), (15, 41), (16, 0), (16, 1), (16, 2), (16,
3), (16, 4), (16, 5), (16, 6), (16, 7), (16, 8), (16, 9), (16, 10), (16, 11), (16, 12), (16, 1
3), (16, 14), (16, 15), (16, 16), (16, 17), (16, 18), (16, 19), (16, 20), (16, 21), (16, 22), (1
6, 23), (16, 24), (16, 25), (16, 26), (16, 27), (16, 28), (16, 29), (16, 30), (16, 31), (16, 3
2), (16, 33), (16, 34), (16, 35), (16, 36), (16, 37), (16, 38), (16, 39), (16, 40), (16, 41), (1
7, 0), (17, 1), (17, 2), (17, 3), (17, 4), (17, 5), (17, 6), (17, 7), (17, 8), (17, 9), (17, 1
0), (17, 11), (17, 12), (17, 13), (17, 14), (17, 15), (17, 16), (17, 17), (17, 18), (17, 19), (1
7, 20), (17, 21), (17, 22), (17, 23), (17, 24), (17, 25), (17, 26), (17, 27), (17, 28), (17, 2
9), (17, 30), (17, 31), (17, 32), (17, 33), (17, 34), (17, 35), (17, 36), (17, 37), (17, 38), (1
7, 39), (17, 40), (17, 41), (18, 0), (18, 1), (18, 2), (18, 3), (18, 4), (18, 5), (18, 6), (18,
```

7), (18, 8), (18, 9), (18, 10), (18, 11), (18, 12), (18, 13), (18, 14), (18, 15), (18, 16), (18, 17), (18, 18), (18, 19), (18, 20), (18, 21), (18, 22), (18, 23), (18, 24), (18, 25), (18, 26), (18, 27), (18, 28), (18, 29), (18, 30), (18, 31), (18, 32), (18, 33), (18, 34), (18, 35), (18, 36), (18, 37), (18, 38), (18, 39), (18, 40), (18, 41), (19, 0), (19, 1), (19, 2), (19, 3), (19, 4), (19, 5), (19, 6), (19, 7), (19, 8), (19, 9), (19, 10), (19, 11), (19, 12), (19, 13), (19, 14), (19, 15), (19, 16), (19, 17), (19, 18), (19, 19), (19, 20), (19, 21), (19, 22), (19, 23), (19, 24), (19, 25), (19, 26), (19, 27), (19, 28), (19, 29), (19, 30), (19, 31), (19, 32), (19, 33), (19, 34), (19, 35), (19, 36), (19, 37), (19, 38), (19, 39), (19, 40), (19, 41), (20, 0), (20, 1), (20, 2), (20, 3), (20, 4), (20, 5), (20, 6), (20, 7), (20, 8), (20, 9), (20, 10), (20, 11), (20, 12), (20, 13), (20, 14), (20, 15), (20, 16), (20, 17), (20, 18), (20, 19), (20, 20), (20, 21), (20, 22), (20, 23), (20, 24), (20, 25), (20, 26), (20, 27), (20, 28), (20, 29), (20, 30), (20, 31), (20, 32), (20, 33), (20, 34), (20, 35), (20, 36), (20, 37), (20, 38), (20, 39), (20, 40), (20, 41), (21, 0), (21, 1), (21, 2), (21, 3), (21, 4), (21, 5), (21, 6), (21, 7), (21, 8), (21, 9), (21, 10), (21, 11), (21, 12), (21, 13), (21, 14), (21, 15), (21, 16), (21, 17), (21, 18), (21, 19), (21, 20), (21, 21), (21, 22), (21, 23), (21, 24), (21, 25), (21, 26), (21, 27), (21, 28), (21, 29), (21, 30), (21, 31), (21, 32), (21, 33), (21, 34), (21, 35), (21, 36), (21, 37), (21, 38), (21, 39), (21, 40), (21, 41), (22, 0), (22, 1), (22, 2), (22, 3), (22, 4), (22, 5), (22, 6), (22, 7), (22, 8), (22, 9), (22, 10), (22, 11), (22, 12), (22, 13), (22, 14), (22, 15), (22, 16), (22, 17), (22, 18), (22, 19), (22, 20), (22, 21), (22, 22), (22, 23), (22, 24), (22, 25), (22, 26), (22, 27), (22, 28), (22, 29), (22, 30), (22, 31), (22, 32), (22, 33), (22, 34), (22, 35), (22, 36), (22, 37), (22, 38), (22, 39), (22, 40), (22, 41), (23, 0), (23, 1), (23, 2), (23, 3), (23, 4), (23, 5), (23, 6), (23, 7), (23, 8), (23, 9), (23, 10), (23, 11), (23, 12), (23, 13), (23, 14), (23, 15), (23, 16), (23, 17), (23, 18), (23, 19), (23, 20), (23, 21), (23, 22), (23, 23), (23, 24), (23, 25), (23, 26), (23, 27), (23, 28), (23, 29), (23, 30), (23, 31), (23, 32), (23, 33), (23, 34), (23, 35), (23, 36), (23, 37), (23, 38), (23, 39), (23, 40), (23, 41), (24, 0), (24, 1), (24, 2), (24, 3), (24, 4), (24, 5), (24, 6), (24, 7), (24, 8), (24, 9), (24, 10), (24, 11), (24, 12), (24, 13), (24, 14), (24, 15), (24, 16), (24, 17), (24, 18), (24, 19), (24, 20), (24, 21), (24, 22), (24, 23), (24, 24), (24, 25), (24, 26), (24, 27), (24, 28), (24, 29), (24, 30), (24, 31), (24, 32), (24, 33), (24, 34), (24, 35), (24, 36), (24, 37), (24, 38), (24, 39), (24, 40), (24, 41), (25, 0), (25, 1), (25, 2), (25, 3), (25, 4), (25, 5), (25, 6), (25, 7), (25, 8), (25, 9), (25, 10), (25, 11), (25, 12), (25, 13), (25, 14), (25, 15), (25, 16), (25, 17), (25, 18), (25, 19), (25, 20), (25, 21), (25, 22), (25, 23), (25, 24), (25, 25), (25, 26), (25, 27), (25, 28), (25, 29), (25, 30), (25, 31), (25, 32), (25, 33), (25, 34), (25, 35), (25, 36), (25, 37), (25, 38), (25, 39), (25, 40), (25, 41), (26, 0), (26, 1), (26, 2), (26, 3), (26, 4), (26, 5), (26, 6), (26, 7), (26, 8), (26, 9), (26, 10), (26, 11), (26, 12), (26, 13), (26, 14), (26, 15), (26, 16), (26, 17), (26, 18), (26, 19), (26, 20), (26, 21), (26, 22), (26, 23), (26, 24), (26, 25), (26, 26), (26, 27), (26, 28), (26, 29), (26, 30), (26, 31), (26, 32), (26, 33), (26, 34), (26, 35), (26, 36), (26, 37), (26, 38), (26, 39), (26, 40), (26, 41), (27, 0), (27, 1), (27, 2), (27, 3), (27, 4), (27, 5), (27, 6), (27, 7), (27, 8), (27, 9), (27, 10), (27, 11), (27, 12), (27, 13), (27, 14), (27, 15), (27, 16), (27, 17), (27, 18), (27, 19), (2

```
7, 20), (27, 21), (27, 22), (27, 23), (27, 24), (27, 25), (27, 26), (27, 27), (27, 28), (27, 2
9), (27, 30), (27, 31), (27, 32), (27, 33), (27, 34), (27, 35), (27, 36), (27, 37), (27, 38), (2
7, 39), (27, 40), (27, 41), (28, 0), (28, 1), (28, 2), (28, 3), (28, 4), (28, 5), (28, 6), (28,
7), (28, 8), (28, 9), (28, 10), (28, 11), (28, 12), (28, 13), (28, 14), (28, 15), (28, 16), (28,
17), (28, 18), (28, 19), (28, 20), (28, 21), (28, 22), (28, 23), (28, 24), (28, 25), (28, 26),
(28, 27), (28, 28), (28, 29), (28, 30), (28, 31), (28, 32), (28, 33), (28, 34), (28, 35), (28, 3
6), (28, 37), (28, 38), (28, 39), (28, 40), (28, 41), (29, 0), (29, 1), (29, 2), (29, 3), (29,
4), (29, 5), (29, 6), (29, 7), (29, 8), (29, 9), (29, 10), (29, 11), (29, 12), (29, 13), (29, 1
4), (29, 15), (29, 16), (29, 17), (29, 18), (29, 19), (29, 20), (29, 21), (29, 22), (29, 23), (2
9, 24), (29, 25), (29, 26), (29, 27), (29, 28), (29, 29), (29, 30), (29, 31), (29, 32), (29, 3
3), (29, 34), (29, 35), (29, 36), (29, 37), (29, 38), (29, 39), (29, 40), (29, 41), (30, 0), (3
0, 1), (30, 2), (30, 3), (30, 4), (30, 5), (30, 6), (30, 7), (30, 8), (30, 9), (30, 10), (30, 1
1), (30, 12), (30, 13), (30, 14), (30, 15), (30, 16), (30, 17), (30, 18), (30, 19), (30, 20), (3
0, 21), (30, 22), (30, 23), (30, 24), (30, 25), (30, 26), (30, 27), (30, 28), (30, 29), (30, 3
0), (30, 31), (30, 32), (30, 33), (30, 34), (30, 35), (30, 36), (30, 37), (30, 38), (30, 39), (3
0, 40), (30, 41), (31, 0), (31, 1), (31, 2), (31, 3), (31, 4), (31, 5), (31, 6), (31, 7), (31,
8), (31, 9), (31, 10), (31, 11), (31, 12), (31, 13), (31, 14), (31, 15), (31, 16), (31, 17), (3
1, 18), (31, 19), (31, 20), (31, 21), (31, 22), (31, 23), (31, 24), (31, 25), (31, 26), (31, 2
7), (31, 28), (31, 29), (31, 30), (31, 31), (31, 32), (31, 33), (31, 34), (31, 35), (31, 36), (3
1, 37), (31, 38), (31, 39), (31, 40), (31, 41), (32, 0), (32, 1), (32, 2), (32, 3), (32, 4), (3
2, 5), (32, 6), (32, 7), (32, 8), (32, 9), (32, 10), (32, 11), (32, 12), (32, 13), (32, 14), (3
2, 15), (32, 16), (32, 17), (32, 18), (32, 19), (32, 20), (32, 21), (32, 22), (32, 23), (32, 2
4), (32, 25), (32, 26), (32, 27), (32, 28), (32, 29), (32, 30), (32, 31), (32, 32), (32, 33), (3
2, 34), (32, 35), (32, 36), (32, 37), (32, 38), (32, 39), (32, 40), (32, 41), (33, 0), (33, 1),
(33, 2), (33, 3), (33, 4), (33, 5), (33, 6), (33, 7), (33, 8), (33, 9), (33, 10), (33, 11), (33,
12), (33, 13), (33, 14), (33, 15), (33, 16), (33, 17), (33, 18), (33, 19), (33, 20), (33, 21),
(33, 22), (33, 23), (33, 24), (33, 25), (33, 26), (33, 27), (33, 28), (33, 29), (33, 30), (33, 3
1), (33, 32), (33, 33), (33, 34), (33, 35), (33, 36), (33, 37), (33, 38), (33, 39), (33, 40), (3
3, 41), (34, 0), (34, 1), (34, 2), (34, 3), (34, 4), (34, 5), (34, 6), (34, 7), (34, 8), (34,
9), (34, 10), (34, 11), (34, 12), (34, 13), (34, 14), (34, 15), (34, 16), (34, 17), (34, 18), (3
4, 19), (34, 20), (34, 21), (34, 22), (34, 23), (34, 24), (34, 25), (34, 26), (34, 27), (34, 2
8), (34, 29), (34, 30), (34, 31), (34, 32), (34, 33), (34, 34), (34, 35), (34, 36), (34, 37), (3
4, 38), (34, 39), (34, 40), (34, 41), (35, 0), (35, 1), (35, 2), (35, 3), (35, 4), (35, 5), (35,
6), (35, 7), (35, 8), (35, 9), (35, 10), (35, 11), (35, 12), (35, 13), (35, 14), (35, 15), (35,
16), (35, 17), (35, 18), (35, 19), (35, 20), (35, 21), (35, 22), (35, 23), (35, 24), (35, 25),
(35, 26), (35, 27), (35, 28), (35, 29), (35, 30), (35, 31), (35, 32), (35, 33), (35, 34), (35, 3
5), (35, 36), (35, 37), (35, 38), (35, 39), (35, 40), (35, 41), (36, 0), (36, 1), (36, 2), (36,
3), (36, 4), (36, 5), (36, 6), (36, 7), (36, 8), (36, 9), (36, 10), (36, 11), (36, 12), (36, 1
3), (36, 14), (36, 15), (36, 16), (36, 17), (36, 18), (36, 19), (36, 20), (36, 21), (36, 22), (3
6, 23), (36, 24), (36, 25), (36, 26), (36, 27), (36, 28), (36, 29), (36, 30), (36, 31), (36, 3
```

```
2), (36, 33), (36, 34), (36, 35), (36, 36), (36, 37), (36, 38), (36, 39), (36, 40), (36, 41), (3
7, 0), (37, 1), (37, 2), (37, 3), (37, 4), (37, 5), (37, 6), (37, 7), (37, 8), (37, 9), (37, 1
0), (37, 11), (37, 12), (37, 13), (37, 14), (37, 15), (37, 16), (37, 17), (37, 18), (37, 19), (3
7, 20), (37, 21), (37, 22), (37, 23), (37, 24), (37, 25), (37, 26), (37, 27), (37, 28), (37, 2
9), (37, 30), (37, 31), (37, 32), (37, 33), (37, 34), (37, 35), (37, 36), (37, 37), (37, 38), (3
7, 39), (37, 40), (37, 41), (38, 0), (38, 1), (38, 2), (38, 3), (38, 4), (38, 5), (38, 6), (38,
7), (38, 8), (38, 9), (38, 10), (38, 11), (38, 12), (38, 13), (38, 14), (38, 15), (38, 16), (38,
17), (38, 18), (38, 19), (38, 20), (38, 21), (38, 22), (38, 23), (38, 24), (38, 25), (38, 26),
(38, 27), (38, 28), (38, 29), (38, 30), (38, 31), (38, 32), (38, 33), (38, 34), (38, 35), (38, 3
6), (38, 37), (38, 38), (38, 39), (38, 40), (38, 41), (39, 0), (39, 1), (39, 2), (39, 3), (39,
4), (39, 5), (39, 6), (39, 7), (39, 8), (39, 9), (39, 10), (39, 11), (39, 12), (39, 13), (39, 1
4), (39, 15), (39, 16), (39, 17), (39, 18), (39, 19), (39, 20), (39, 21), (39, 22), (39, 23), (3
9, 24), (39, 25), (39, 26), (39, 27), (39, 28), (39, 29), (39, 30), (39, 31), (39, 32), (39, 3
3), (39, 34), (39, 35), (39, 36), (39, 37), (39, 38), (39, 39), (39, 40), (39, 41), (40, 0), (4
0, 1), (40, 2), (40, 3), (40, 4), (40, 5), (40, 6), (40, 7), (40, 8), (40, 9), (40, 10), (40, 1
1), (40, 12), (40, 13), (40, 14), (40, 15), (40, 16), (40, 17), (40, 18), (40, 19), (40, 20), (4
0, 21), (40, 22), (40, 23), (40, 24), (40, 25), (40, 26), (40, 27), (40, 28), (40, 29), (40, 3
0), (40, 31), (40, 32), (40, 33), (40, 34), (40, 35), (40, 36), (40, 37), (40, 38), (40, 39), (4
0, 40), (40, 41), (41, 0), (41, 1), (41, 2), (41, 3), (41, 4), (41, 5), (41, 6), (41, 7), (41,
8), (41, 9), (41, 10), (41, 11), (41, 12), (41, 13), (41, 14), (41, 15), (41, 16), (41, 17), (4
1, 18), (41, 19), (41, 20), (41, 21), (41, 22), (41, 23), (41, 24), (41, 25), (41, 26), (41, 2
7), (41, 28), (41, 29), (41, 30), (41, 31), (41, 32), (41, 33), (41, 34), (41, 35), (41, 36), (4
1, 37), (41, 38), (41, 39), (41, 40), (41, 41)]
```

In [0]:

```python
#Devuelve el factorial de un numero
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

In [105]:

```python
#Se genera una solucion aleatoria con comienzo en en el nodo 0
def crear_solucion(Nodos):
  solucion = [0]
  for i in range(len(Nodos)-1):
    solucion = solucion + [random.choice(list(set(Nodos) - set({0}) - set(solucion)))]
  return solucion
```

```
#Devuelve la distancia entre dos nodos
def distancia(a,b, problem):
  return problem.wfunc(a,b)

#distancia(0,1,problem)

#Devuelve la distancia total de una trayectoria
def distancia_total(solucion, problem):
  distancia_total = 0
  for i in range(len(solucion)-1):
    distancia_total += distancia(solucion[i] ,solucion[i+1] ,  problem)
  return distancia_total + distancia(solucion[len(solucion)-1] ,solucion[0], problem)

solucion=crear_solucion(Nodos)
#print(solucion)
distancia_total(solucion,problem)
```

Out[105]: 4987

Búsqueda aleatoria

In [106]:

```
def busquedaAleatoria(problem, N):

  Nodos = list(problem.get_nodes())

  mejor_solucion = []
  mejor_distancia = 10e100

  for i in range(N):
    solucion=crear_solucion(Nodos)

    distancia = distancia_total(solucion,problem)

    if distancia < mejor_distancia:

      mejor_solucion = solucion

      mejor_distancia = distancia
```

```
  print("Mejor solución :",mejor_solucion)
  print("Mejor distancia :",mejor_distancia)

  return mejor_solucion


sol=busquedaAleatoria(problem, 100)
```

```
Mejor solución : [0, 32, 10, 8, 26, 23, 21, 39, 40, 13, 16, 41, 20, 19, 31, 27, 30, 4, 3, 18, 1
2, 6, 37, 11, 36, 17, 1, 7, 14, 35, 34, 24, 29, 33, 22, 28, 9, 5, 15, 2, 38, 25]
Mejor distancia : 4072
```

Búsqueda local

In [107]:
```
def genera_vecina(solucion):
  #Generador de soluciones vecinas: 2-opt (intercambiar 2 nodos) Si hay N nodos se generan (N-
1)x(N-2)/2 soluciones
  #print(solucion)
  mejor_solucion = []
  #mejor_distancia = 10e100
  mejor_distancia = float("infinity") #Modificación
  for i in range(1,len(solucion)-1):
    for j in range(i+1, len(solucion)):
      vecina = solucion[:i] + [solucion[j]] + solucion[i+1:j] + [solucion[i]] + solucion[j+1:]
      distancia_vecina = distancia_total(vecina, problem)
      if distancia_vecina <= mejor_distancia:
        mejor_distancia = distancia_vecina
        mejor_solucion = vecina
  return mejor_solucion

solucion=crear_solucion(Nodos)
print(solucion)

nueva_solucion = genera_vecina(solucion) #Se ve cómo se han intercambiado 2 nodos
print(nueva_solucion)
```

```
[0, 29, 22, 33, 4, 1, 2, 21, 13, 28, 15, 35, 20, 41, 19, 37, 39, 34, 6, 3, 12, 32, 23, 9, 38, 3
```

```
1, 7, 11, 17, 25, 18, 36, 27, 14, 24, 30, 40, 5, 16, 10, 8, 26]
[0, 29, 22, 33, 4, 1, 2, 21, 13, 28, 15, 35, 20, 17, 19, 37, 39, 34, 6, 3, 12, 32, 23, 9, 38, 3
1, 7, 11, 41, 25, 18, 36, 27, 14, 24, 30, 40, 5, 16, 10, 8, 26]
```

In [111]:

```
def busqueda_local(problem, N):
  mejor_solucion = []
  mejor_distancia = float("infinity") #Modificación

  Nodos = list(problem.get_nodes())

  solucion_referencia = crear_solucion(Nodos)

  for i in range(N):
    vecina = genera_vecina(solucion_referencia)
    distancia_vecina = distancia_total(vecina, problem)

    if distancia_vecina < mejor_distancia:
      mejor_solucion = vecina
      mejor_distancia = distancia_vecina

    solucion_referencia = vecina

  print("Mejor solución:" , mejor_solucion)
  print("Distancia     :" , mejor_distancia)
  return mejor_solucion



#Modificación: Le cambio el nombre a la variable para después usar esta solución en el gráfico
solLocal = busqueda_local(problem, 100)
```

```
Mejor solución: [0, 31, 35, 36, 17, 7, 1, 3, 2, 4, 26, 5, 13, 19, 6, 27, 28, 29, 39, 21, 24, 40,
23, 41, 9, 8, 10, 18, 14, 16, 15, 37, 20, 33, 34, 12, 11, 25, 22, 38, 30, 32]
Distancia     : 1833
```

Recocido simulado

In [0]:

```
def genera_vecina_aleatorio(solucion):
  #Generador de 1 solucion vecina 2-opt (intercambiar 2 nodos)
  #Se puede mejorar haciendo que la elección no se uniforme sino entre las que estén más proxim
```

```
  #Se puede mejorar haciendo que la elección no se uniforme sino entre las que estén más próxim
as
  i = random.choice(range(1, len(solucion)) )
  j = random.choice(list(set(range(1, len(solucion))) - {i}))
  vecina = solucion[:i] + [solucion[j]] + solucion[i+1:j] + [solucion[i]] + solucion[j+1:]
  return vecina


def probabilidad(T,d):
  r=random.random()
  return r <= (e**(-1*d)/(T*1.0)) #Modificación para poner todo en 1 línea
  #if(r <= (e**(-1*d)/(T*1.0))):
  #  return True
  #else:
  #  return False

def bajar_temperatura(T):
  return T-1
```

In [113]:
```
def recocido_simulado(problem, TEMPERATURA):
  #problem = datos del problema
  #T = Temperatura

  solucion_referencia = crear_solucion(Nodos)
  distancia_referencia = distancia_total(solucion_referencia, problem)

  mejor_solucion = []
  mejor_distancia = float("infinity") #Modificación

  while TEMPERATURA > 0:
    #Genera una solución vecina(aleatoria)
    vecina = genera_vecina_aleatorio(solucion_referencia)
    #vecina = genera_vecina(solucion_referencia)#Mejores soluciones

    #Calcula su valor(distancia)
    distancia_vecina = distancia_total(vecina, problem)

    #Si es la mejor solución de todas se guarda
    if distancia_vecina < mejor_distancia:
        mejor_solucion = vecina
```

```
        mejor_distancia = distancia_vecina

    #Si la nueva vecina es mejor, se cambia y si es peor se cambia según una probabilidad depen
diente de T y de |distancia_referencia - distancia_vecina|
    if distancia_vecina < distancia_referencia or probabilidad(TEMPERATURA, abs(distancia_refer
encia - distancia_vecina) ) :
      solucion_referencia = vecina
      distancia_referencia = distancia_vecina

    TEMPERATURA = bajar_temperatura(TEMPERATURA)

  print("La mejor solución encontrada es " , end="")
  print(mejor_solucion)
  print("con una distancia total de " , end="")
  print(mejor_distancia)
  return mejor_solucion

sol = recocido_simulado(problem, 10000)
```

```
La mejor solución encontrada es [0, 27, 28, 29, 8, 41, 25, 11, 13, 19, 14, 16, 15, 37, 36, 35, 3
1, 17, 7, 1, 30, 39, 21, 24, 40, 23, 26, 5, 6, 2, 32, 34, 20, 33, 38, 22, 9, 10, 12, 18, 4, 3]
con una distancia total de 1732
```

Colonia de hormigas

In [0]:
```
import math
#Colonia de hormigas
def Add_Nodo(problem, H ,T ) : #H (hormiga):recorrido parcial      T:fermomona ---> mejorar est
e método, no se está teniendo en cuenta T (jugar con los dos parámetros, distancia y feromona)
  #Establecer una una funcion de probabilidad para
  # añadir un nuevo nodo dependiendo de los nodos mas cercanos y de las feromonas depositadas
  #print(H)
  Nodos = list(problem.get_nodes())
  return random.choice(   list(set(range(1,len(Nodos))) - set(H) )  ) #añade un nodo de modo al
eatorio, debería ser en base a la feromona T

#Modificación: creo nuevo método para elegir el siguiente nodo, mediante una función de probabi
lidad que tenga en cuenta y las feromonas
def Add_Nodo_Mod(problem, H ,T, actual ) :
```

```
alfa=1.0 #Peso que se da al rastro
beta=2.5 #Peso que se da a la distancia

Nodos = list(problem.get_nodes())

listaValores  = []
listaValoresPesos = []

for i in range(len(Nodos)):

  if len(T)>actual and i<len(T[actual]) :

    feromona  = math.pow((T[actual][i]), alfa)

    #Distancia entre el nodo i y el último nodo de la hormiga
    hormiga=H[actual]
    nodohormiga=hormiga[len(hormiga)-1]
    dist=(distancia(nodohormiga,Nodos[i],problem))
    #Divido entre 1 porque a menor distancia, mayor peso le quiero dar al nodo
    # Lo multiplico *100 y posteriormente lo convertiré a entero para que funcione correctamente la función de calculo de probabilidad en base a pesos)
    if dist!=0:
      dist=1+(1/dist)*100

    peso = math.pow(dist, beta) * feromona

    #Sólo añado el nodo si no existe previamente
    s=set(H[actual])
    if Nodos[i] not in s:
      elem=(Nodos[i],peso)
      listaValoresPesos.append(elem)


if(len(listaValoresPesos)>0):
  listadoPesos = [nodo for nodo, peso in listaValoresPesos for i in range(int(peso))]
  valor= random.choice(listadoPesos)
  return valor
else:
  return random.choice(list(set(range(1,len(Nodos))) - set(H[actual]) )  )
```

```
def Incrementa_Feromona(problem, T, H):
  #Incrementar segun la calidad de la solución. Añadir una cantidad inversamente proporcional a
 la distancia total
  for i in range(len(H)-1):
    T[H[i]][H[i+1]] += 1000/distancia_total(H, problem) #más feromonas a las distancias más peq
ueñas
  return T

def Evaporar_Feromonas(T):
  #Podemos elegir diferentes funciones de evaporación dependiendo de la cantidad actual y de la
 suma total de feromonas depositadas,...
  #Evapora 0.3 el valor de la feromona, sin que baje de 1    --> mejorable, podría hacerse en b
ase al número de ciclos, etc...
  T = [[ max(T[i][j] - 0.3 , 1) for i in range(len(Nodos)) ] for j in range(len(Nodos))]
  return T

#Modificación: Intento de Función de evaporación de hormigas en función de los ciclos (Al final
 no la uso, porque no veo que mejore la función inicial)
def Evaporar_Feromonas_Mod(T,ciclos,N):

  #Evaporar en función de los ciclos (lo multiplico * 2 para corregir el efecto que se produce
 en los primeros ciclos, sobretodo cuando hay muchos agentes.
  #La evaporación era mínima y provocaba que el algoritmo no arrojase resultados tan óptimos):
  ratioEvaporacion=(ciclos/N)*2


  T = [[ max(T[i][j] - ratioEvaporacion , 1) for i in range(len(Nodos)) ] for j in range(len(No
dos))]
  return T
```

In [116]:

```
#problem = datos del problema
#N = Número de agentes(hormigas)
def hormigas(problem, N):
  #Nodos
  Nodos = list(problem.get_nodes())
  #Aristas
  Aristas = list(problem.get_edges())
```

```
#Inicializa las aristas con una cantidad inicial de feromonas:1 (vector de 2 dimensiones)
T = [[ 1 for _ in range(len(Nodos)) ] for _ in range(len(Nodos))]
#print(T)

#Se generan los agentes(hormigas) que serán estructuras de caminos desde 0
Hormiga = [[0] for _ in range(N)]

#Recorre cada agente construyendo la solución
ciclos=0
for h in range(N):
  #Para cada agente se construye un camino
  for i in range(len(Nodos)-1):

    #Elige el siguiente nodo
    #Nuevo_Nodo = Add_Nodo(problem, Hormiga[h] ,T )
    Nuevo_Nodo = Add_Nodo_Mod(problem, Hormiga ,T, h )

    Hormiga[h].append(Nuevo_Nodo)

  #Incrementa feromonas en esa arista
  T = Incrementa_Feromona(problem, T, Hormiga[h] )

  #Evapora Feromonas
  T = Evaporar_Feromonas(T)
  #T = Evaporar_Feromonas_Mod(T,ciclos,N)
  ciclos+=1

  #Seleccionamos el mejor agente
mejor_solucion = []
mejor_distancia = float("infinity")
for h in range(N):
  distancia_actual = distancia_total(Hormiga[h], problem)
  if distancia_actual < mejor_distancia:
    mejor_solucion = Hormiga[h]
    mejor_distancia =distancia_actual


print(mejor_solucion)
print(mejor_distancia)
```

```
solucionHormigas=hormigas(problem, 1000)
```

```
[0, 1, 8, 28, 36, 37, 15, 16, 14, 6, 27, 2, 30, 29, 9, 22, 38, 24, 18, 12, 10, 25, 11, 41, 23, 3
9, 40, 21, 3, 4, 5, 7, 34, 33, 20, 13, 19, 26, 17, 31, 35, 32]
2413
```

Pruebas con distinto número de agentes para el problema de las hormigas

In [77]:
```
#Con 10 agentes
VECES=5
for i in range(VECES):
  hormigas(problem, 10)
```

```
[0, 3, 27, 2, 1, 37, 14, 15, 16, 17, 31, 28, 4, 5, 35, 36, 34, 20, 33, 30, 24, 21, 32, 39, 41, 2
3, 26, 6, 7, 9, 8, 19, 13, 18, 25, 10, 29, 38, 22, 12, 11, 40]
2813
[0, 29, 30, 28, 3, 4, 5, 31, 15, 13, 19, 37, 1, 17, 40, 21, 9, 2, 27, 34, 38, 22, 39, 24, 10, 2
5, 41, 23, 8, 18, 6, 36, 16, 14, 7, 12, 11, 26, 32, 33, 20, 35]
2673
[0, 1, 7, 17, 6, 4, 3, 2, 28, 27, 25, 10, 12, 11, 26, 5, 14, 16, 15, 33, 34, 20, 29, 21, 9, 23,
41, 36, 35, 19, 13, 37, 31, 32, 30, 8, 39, 38, 22, 18, 40, 24]
2593
[0, 1, 3, 6, 7, 15, 14, 16, 37, 17, 9, 8, 27, 2, 32, 28, 12, 18, 26, 5, 13, 23, 4, 29, 22, 25, 1
0, 41, 11, 33, 34, 35, 31, 20, 36, 30, 24, 40, 21, 39, 38, 19]
2751
[0, 3, 1, 6, 4, 37, 17, 31, 18, 29, 23, 41, 25, 12, 15, 16, 14, 32, 20, 33, 30, 19, 5, 26, 10, 2
2, 38, 24, 2, 27, 28, 11, 13, 7, 8, 9, 39, 40, 21, 35, 36, 34]
2807
```

In [78]:
```
#Con 42 agentes tantos agentes como nodos
VECES=5
for i in range(VECES):
  hormigas(problem, 42)
```

```
[0, 27, 2, 3, 6, 25, 11, 14, 16, 15, 37, 18, 12, 26, 13, 19, 28, 17, 31, 33, 29, 10, 41, 23, 9,
22, 38, 40, 24, 39, 21, 8, 4, 1, 5, 7, 36, 35, 20, 34, 32, 30]
2245
[0, 3, 27, 2, 6, 7, 4, 18, 29, 8, 10, 12, 11, 30, 28, 1, 32, 33, 20, 34, 31, 17, 37, 14, 16, 15,
19, 26, 5, 13, 25, 22, 38, 9, 23, 41, 24, 21, 40, 39, 36, 35]
```

```
2208
[0, 1, 6, 4, 28, 2, 27, 3, 30, 29, 19, 5, 12, 10, 25, 23, 41, 9, 39, 38, 8, 18, 37, 16, 14, 15,
31, 26, 13, 33, 34, 32, 11, 40, 21, 24, 22, 17, 7, 20, 35, 36]
2566
[0, 5, 16, 15, 7, 37, 14, 12, 11, 18, 10, 25, 8, 41, 9, 2, 27, 3, 4, 29, 39, 21, 24, 40, 23, 22,
38, 6, 1, 13, 19, 17, 35, 20, 34, 30, 28, 31, 36, 32, 33, 26]
2375
[0, 1, 6, 14, 15, 19, 5, 2, 27, 4, 3, 28, 13, 26, 31, 35, 36, 29, 30, 39, 23, 21, 40, 9, 8, 17,
37, 16, 7, 10, 25, 11, 41, 24, 12, 18, 33, 22, 38, 32, 34, 20]
2642
```

In [79]:

```
#Con 1000 agentes
VECES=5
for i in range(VECES):
  hormigas(problem, 1000)
```

```
[0, 37, 15, 14, 16, 29, 30, 28, 1, 6, 26, 5, 4, 13, 19, 31, 12, 11, 25, 3, 27, 2, 41, 23, 8, 10,
39, 9, 40, 24, 21, 22, 7, 20, 33, 34, 32, 17, 36, 35, 38, 18]
2486
[0, 18, 19, 16, 14, 15, 37, 35, 36, 2, 27, 5, 26, 11, 12, 10, 25, 22, 39, 8, 29, 30, 32, 28, 9,
21, 40, 38, 13, 6, 3, 1, 31, 17, 7, 4, 23, 41, 34, 33, 20, 24]
2669
[0, 30, 27, 2, 32, 29, 35, 31, 34, 4, 3, 18, 25, 11, 12, 5, 19, 13, 14, 16, 15, 26, 22, 28, 6,
1, 7, 37, 36, 20, 33, 39, 24, 40, 21, 8, 23, 9, 41, 10, 38, 17]
2573
[0, 2, 3, 27, 13, 16, 14, 15, 37, 25, 12, 11, 21, 40, 23, 41, 10, 1, 26, 18, 5, 31, 17, 20, 33,
34, 32, 35, 36, 19, 28, 24, 22, 39, 9, 8, 30, 38, 7, 6, 4, 29]
2507
[0, 6, 1, 18, 26, 19, 9, 8, 23, 41, 25, 10, 4, 3, 2, 27, 28, 38, 22, 24, 21, 39, 40, 30, 29, 13,
15, 14, 16, 17, 33, 34, 32, 12, 11, 7, 31, 20, 5, 35, 37, 36]
2544
```

In [80]:

```
#Con 5000 agentes
VECES=5
for i in range(VECES):
  hormigas(problem, 5000)
```

```
[0, 1, 7, 19, 5, 17, 32, 10, 4, 26, 13, 23, 41, 12, 11, 6, 2, 3, 27, 28, 30, 8, 18, 9, 29, 38, 2
0, 33, 34, 31, 15, 14, 16, 37, 36, 35, 22, 39, 40, 21, 24, 25]
2483
```

```
[0, 33, 20, 34, 32, 28, 27, 2, 4, 26, 13, 14, 16, 15, 37, 17, 31, 40, 21, 39, 23, 41, 11, 12, 8,
10, 25, 18, 3, 1, 30, 29, 9, 7, 24, 35, 36, 19, 5, 6, 38, 22]
2603
[0, 30, 10, 8, 9, 26, 1, 4, 6, 5, 15, 16, 14, 17, 3, 31, 7, 11, 12, 25, 41, 23, 21, 24, 40, 29,
19, 13, 18, 33, 34, 28, 27, 2, 36, 35, 20, 32, 38, 22, 39, 37]
2602
[0, 7, 12, 10, 3, 1, 2, 27, 28, 4, 26, 13, 19, 6, 32, 34, 33, 5, 38, 22, 21, 24, 40, 36, 17, 31,
15, 14, 16, 37, 20, 30, 29, 39, 9, 41, 8, 25, 23, 18, 11, 35]
2715
[0, 1, 2, 3, 27, 18, 26, 19, 13, 36, 31, 17, 5, 11, 12, 10, 29, 30, 7, 16, 14, 15, 37, 35, 34, 3
3, 20, 32, 9, 8, 21, 40, 23, 41, 25, 4, 6, 28, 38, 39, 22, 24]
2309
```

En vista de los datos, se observa una mejora sustancial entre usar 10 agentes o 42 (tantos agentes/hormigas como nodos). A partir de ahí, no se ven mejoras en usar más número de agentes. Además, vemos que usando 42 agentes obtenemos tiempos de ejecución bastante razonables.

Aplicación del algoritmo de las hormigas a otros problemas:

El primer algoritmo de colonia de hormigas, tenía como objetivo resolver el problema del viajante.

Además, se ha visto que este algoritmo es útil en la resolución de los siguientes problemas:

- Problema de la mochila:

  ```
  El problema de la mochila consiste en seleccionar un conjunto de objetos de manera tal que el val
  os objetos sea máximo y no supere un límite establecido (mochila).

  Para resolver el problema de la mochila usando el algoritmo de las hormigas:

  Se inicializa cada objeto con una cantidad mínima de feromonas.

  Cada hormiga se coloca en una posición aleatoria y va seleccionando cada nodo de forma iterativa
  ando a la mochila. Los objetos totales no pueden sobrepasar la capacidad de la mochila.

  Por último se selecciona la mejor solución y se actualizan los rastros de las feromonas.
  ```

- Para Optimización de estructuras de hormigón armado. (Reforzando aquellos elementos que se han empleado para la construcción, calidad de la solución.)

- Para problemas de distribución en planta (Para buscar la mejor asignación espacial de estaciones o celdas de trabajo)
- Para detectar fallos de programación
- Tareas de supervisión de las máquinas de aprendizaje, encargadas de agrupar los grupos de objetos que son similares

Referencias:

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

https://www.agenciasinc.es/Noticias/Colonias-de-hormigas-detectan-fallos-de-programacion

http://www.academia.edu/3567732/ACHPM_ALGORITMO_DE_OPTIMIZACI%C3%93N_CON_COLONIA_DE_HORMIGAS_PARA_EL_PRO

https://victoryepes.blogs.upv.es/tag/colonia-de-hormigas/

https://revistas.udistrital.edu.co/ojs/index.php/visele/article/view/11897

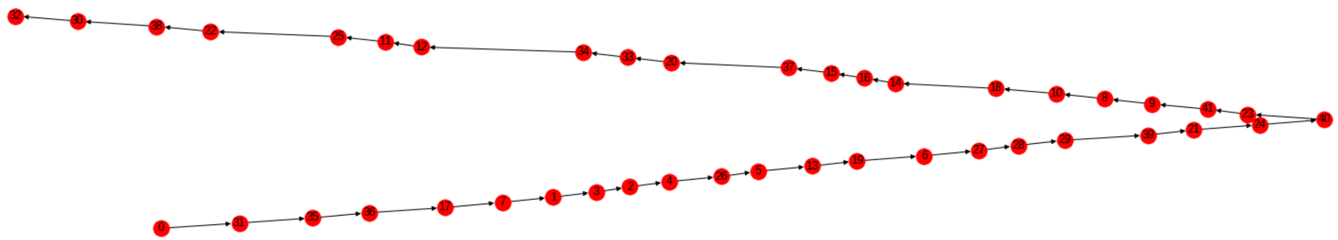
Gráfico para visualizar mejor camino y distancia entre los nodos.

El gráfico mostrado es el obtenido mediante el algoritmo de busquedaLocal

In [343]:
```
import matplotlib.pyplot as plt
import networkx as nx
import sys

#Gráfico para mostrar la solución, representando el camino, así como la distancia entre los nod
os
def dibujar(solucion):
  G = nx.DiGraph()
  G.clear()
  G.pos={}
  x=0
  y=0
  posiciones=[]
  tamanio=len(solucion)
  mitad=len(solucion)//2
  distancias=[]
  i=0
```

```
  for elem in solucion:

    #Posiciono el primer elemento en la coordenada (0,0)
    if(i==0):
        G.pos[elem]=(0,0)
        posiciones.append(0)
    #Para el resto de casos calculo la posición x en función de la distancia
    #Además voy sumando siempre el valor 30 para x e y para que una mejor visualización
    #y que no se queden los nodos apelotonados
    else:
      #Añado la arista del elemento anterior y este
      G.add_edge(solucion[i-1],solucion[i])
      #Calculo la distancia entre el nodo anterior y este
      dist=distancia(solucion[i],solucion[i-1],problem)
      #Añado la distancia al array
      distancias.append(dist)

      #Para una mejor visualización, divido el gráfico en 2 partes, ya que si no no se visualiz
a correctamente, al haber tantos nodos
      #Si es menor que la mitad, voy a visualizar el gráfico de forma ascendente hacia la derec
ha
      if i<=mitad:
        y+=30
        x+=30
        posicionAnterior=posiciones[i-1]
        posicionActual=posicionAnterior+dist
      #Si es menor que la mitad, hacia la izquierda y hacia arriba
      else:
        y+=30
        x-=30
        posicionAnterior=posiciones[i-1]
        posicionActual=posicionAnterior-dist


      #Establezco la posición del nodo actual, y añado la posición a array
      G.pos[elem]=((posicionActual)+x,y)
      posiciones.append(posicionActual)
    i+=1

  #Visualizo la figura
```

```
  plt.figure(figsize=(30, 5))
  pos = nx.spring_layout(G)
  nx.draw_networkx_edges(G, pos, arrows=True)
  nx.draw(G,G.pos, with_labels=True, size=1000,cmap=plt.cm.Reds_r)
  plt.show()

  #Pinto, además el camino y las distancias para que se pueda comparar con el gráfico
  print("Camino :"+str(solucion))
  print("Distancias :"+str(distancias))

#Pasamos al gráfico como parámetro la solución obtenida anteriormente mediante el algoritmo de
 búsqueda local
dibujar(solLocal)
```



```
Camino :[0, 31, 35, 36, 17, 7, 1, 3, 2, 4, 26, 5, 13, 19, 6, 27, 28, 29, 39, 21, 24, 40, 23, 41,
9, 8, 10, 18, 14, 16, 15, 37, 20, 33, 34, 12, 11, 25, 22, 38, 30, 32]
Distancias :[66, 60, 39, 63, 40, 32, 23, 11, 18, 34, 15, 36, 25, 52, 37, 18, 27, 72, 26, 51, 49,
64, 19, 38, 28, 29, 44, 93, 8, 11, 22, 113, 24, 24, 168, 14, 28, 126, 36, 67, 46]
```