Models and Methods for Operation Research

Solving The Traveling Salesman Problem (TSP)

Guillermo Villanueva Benito

December 2021

Contents

1 TSP data

We are considering the solution of the symmetric version of the TSP over a complete graph $G = (V, E)$ with $n = |V|$ nodes. More specifically, our graph have 12 nodes, $n = 12$. Moreover, since it is a complete graph the number of edges is

$$|E| = \frac{n(n-1)}{2} = 66 \tag{1}$$

The graph is non-directed and each edge (i, j) has its associated cost $c_{i,j}$ satisfying

$$c_{i,j} = c_{j,i} \tag{2}$$

in order to compute the solution of the symmetric TSP.

We consider the following cost matrix C

$$C = \begin{bmatrix} 0. & 0.723 & 1.046 & 0.811 & 0.588 & 0.547 & 0.97 & 0.452 & 0.361 & 0.444 & 0.605 & 0.359 \\ 0.723 & 0. & 0.844 & 0.618 & 0.455 & 0.186 & 0.276 & 0.534 & 0.467 & 0.306 & 0.707 & 0.39 \\ 1.046 & 0.844 & 0. & 0.255 & 0.498 & 0.791 & 0.773 & 0.597 & 0.716 & 0.954 & 0.463 & 0.975 \\ 0.811 & 0.618 & 0.255 & 0. & 0.244 & 0.543 & 0.609 & 0.359 & 0.468 & 0.701 & 0.288 & 0.72 \\ 0.588 & 0.455 & 0.498 & 0.244 & 0. & 0.33 & 0.551 & 0.156 & 0.231 & 0.467 & 0.255 & 0.479 \\ 0.547 & 0.186 & 0.791 & 0.543 & 0.33 & 0. & 0.425 & 0.369 & 0.285 & 0.185 & 0.562 & 0.248 \\ 0.97 & 0.276 & 0.773 & 0.609 & 0.551 & 0.425 & 0. & 0.681 & 0.661 & 0.578 & 0.798 & 0.658 \\ 0.452 & 0.534 & 0.597 & 0.359 & 0.156 & 0.369 & 0.681 & 0. & 0.134 & 0.448 & 0.212 & 0.431 \\ 0.361 & 0.467 & 0.716 & 0.468 & 0.231 & 0.285 & 0.661 & 0.134 & 0. & 0.324 & 0.344 & 0.299 \\ 0.444 & 0.306 & 0.954 & 0.701 & 0.467 & 0.185 & 0.578 & 0.448 & 0.324 & 0. & 0.659 & 0.088 \\ 0.605 & 0.707 & 0.463 & 0.288 & 0.255 & 0.562 & 0.798 & 0.212 & 0.344 & 0.659 & 0. & 0.642 \\ 0.359 & 0.39 & 0.975 & 0.72 & 0.479 & 0.248 & 0.658 & 0.431 & 0.299 & 0.088 & 0.642 & 0. \end{bmatrix}$$

(3)

In order to compute cost matrix C, we have used a simple python code, Listing (1). We have randomly generated 12 points in the region $[0, 1] \times [0, 1] \subset \mathbb{R}^2$ associated with the position of the 12 nodes of the graph. For each node $v_i \in V$ it position p_i is

$$p_i = (x_i, y_i) \tag{4}$$

Then, the cost matrix C is the distance matrix. More specifically, for each pair of nodes (v_i, v_j) its associated cost $c_{i,j}$ is given by

$$c_{i,j} = \|p_i - p_j\| = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{5}$$

A graphical representation of this graph is shown in Figure (1).

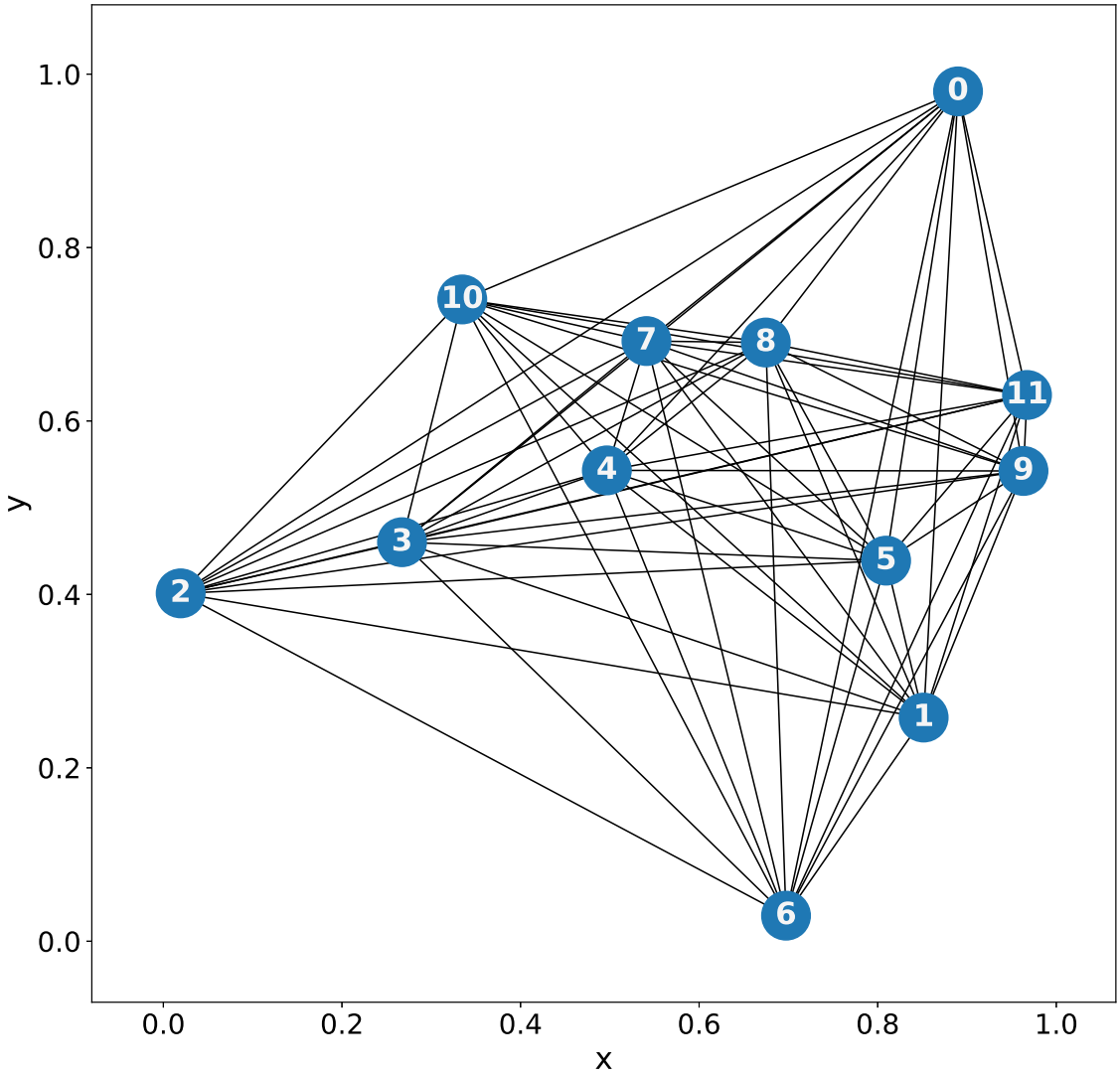

Figure 1: Graph representation

Listing 1: Code section 1

```
import numpy as np
import matplotlib.pyplot as plt
import networkx
```

```
import random as rn

# Create random coordinates
A = np.random.random((12,2))
x = A[:,0];
y = A[:,1];

# Compute cost matrix C (distance matrix)
C = np.zeros((12,12));
for i in range(len(x)):
    for j in range(len(y)):
        C[i,j] = np.round(np.sqrt((x[i]-x[j])**2+(y[i]-y[j])**2),3)

# Create complete graph with n = 12
G = networkx.complete_graph(12)

# Set the position of nodes
pos = networkx.spring_layout(G)
for i in range(12):
  pos[i] = A[i,:]

# Plot the graph
fig, ax = plt.subplots(figsize=(10, 10))
networkx.draw(G, pos, with_labels=True,font_color="whitesmoke",ax=ax,node_size
 ↪  = 1000,font_weight ='bold',font_size = 20)
limits=plt.axis('on') # turns on axis
plt.ylabel('y',fontsize=20)
plt.xlabel('x',fontsize=20)
ax.tick_params(left=True, bottom=True, labelleft=True,
 ↪  labelbottom=True,labelsize=18)
```

2 Obtaining an upper bound

In this section we program an heuristic for getting a feasible solution for the symmetric TSP (STSP). The heuristic will consist of two different parts: one constructive part, in which a greedy procedure is applied, and a second part, which consists of an improvement phase.

For the constructive part we will use the Nearest Neighbor primal heuristic.

Nearest Neighbor (NN) heuristic: Start at an arbitrary node i_1 and construct a path $i_1, i_2, \ldots, i_j, i_{j+1}, \ldots, i_n$ where $i_{j+1} = \arg(\min(c_{i,k} : k \in V \backslash \{i_1, i_2, \ldots, i_j\}))$.

For the improvement phase, we will apply the 2-Interchange local search heuristic to obtain better upper bounds for the original TSP.

2-Interchange local search heuristic: Start with a given tour (feasible solution) and replace each pair of non-adjacent edges in the tour with the unique pair of replacement edges if the resulting tour has a lower weight.

The main results of this section are shown in Figure (2). Here, we have applied the 2-Interchange local search heuristic to 12 initial tours. Each tour is the output solution for the NN heuristic taking a different initial arbitrary node. For instance, the red curve (NN(3)) shows the different improving tours obtained by the 2-Interchange local search heuristic taking as initial tour, the tour solution for the NN heuristic with initial node, node number 3. The initial upper bound is the upper bound obtained using only the NN heuristic.

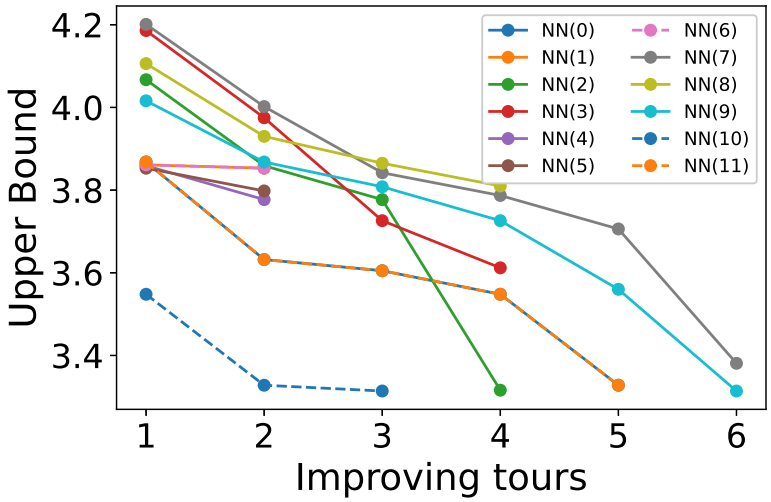

Figure 2: Upper bounds as a function of the different improving tours obtained by the 2-Interchange heuristic. The first upper bound is the upper bound obtained using the NN heuristic with initial node given by the color legend.

As expected, the local search heuristic improves the upper bound obtained by the greedy heuristic. We notice that the best upper bound has been obtained with NN(9) and NN(10). In both cases, the obtained upper bound is

$$\overline{z^*} = 3.314 \tag{6}$$

and the resulting tour is shown in Figure (3).

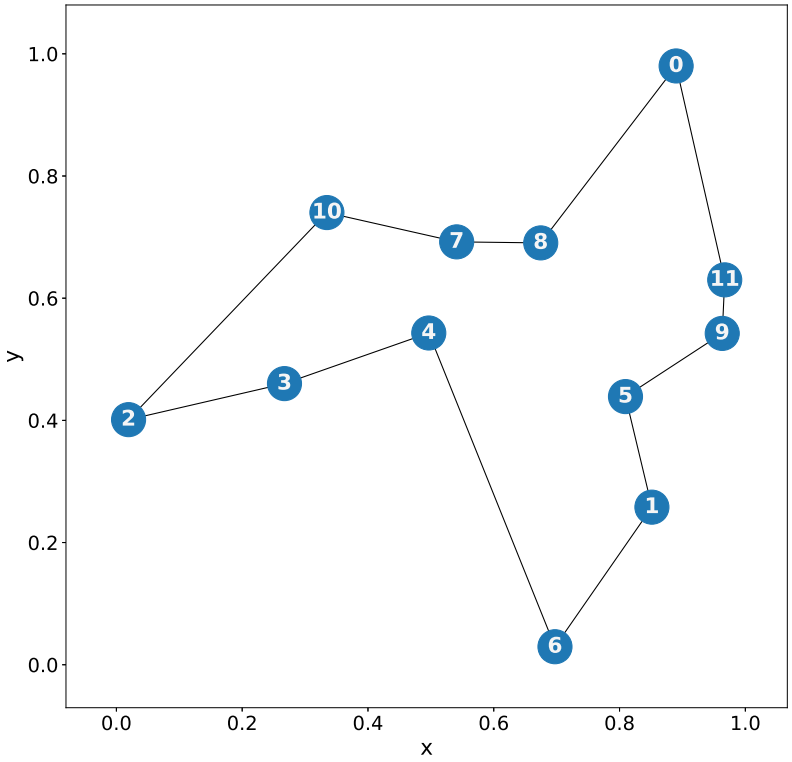

Figure 3: Tour associated with the best upper bound.

The complete list of successive tours with its associated cost is given in the Appendix. The python code used to implement the heuristic is shown below, Listing (2).

Listing 2: Code section 2

```
import numpy as np

# Save results in file test.txt
a = open('test.txt', 'w')
# NN heuristic for each initial node
for k in np.arange(0,12,1):

    # mark used node
    nodes = np.zeros(12);
    nodes[k] = 1;
    # tour
```

```
    tour = [k];
    # Initialize
    ok = len(np.where(nodes==0)[0]);
    i0 = k;
    while ok > 0:
        pos = np.where(nodes==0)[0];
        # Find nearest neighbor
        i1 = pos[np.where(D[pos,i0]== min(D[pos,i0]))[0]];
        # Add nearest neighbor
        tour.append(i1[0])
        # Update cost
        cost[k] = cost[k] + D[i0,i1];
        # mark used node
        nodes[i1] = 1;
        # Updates
        i0 = i1;
        ok = len(np.where(nodes==0)[0]);

    # Add final edge to close tour
    cost[k] = cost[k]+ D[i1,k];
    tour.append(k)
    # Save results
    a.write('NN(%s'%k+'):\n'+'%s' % tour +' -- z* = %s\n' % cost[k])
    # 2-Interchange heuristic for each NN tour solution
    for i in np.arange(0,10,1):

        for j in np.arange(i+2,12,1):
            d1 = D[tour[i],tour[i+1]] + D[tour[j],tour[j+1]];
            d2 = D[tour[i],tour[j]] + D[tour[i+1],tour[j+1]];

            # Do swap if interchange improves upper bound
            if d2 < d1:
                tour[i+1:j+1] = np.flip(tour[i+1:j+1])
                cost[k] = cost[k]-d1+d2;
                a.write('%s' % tour+' -- z* = %s\n' % cost[k])
a.close()
```

3 Obtaining lower bounds

Our goal is to solve the Symmetric Travelling Salesman Problem (P) over the complete graph built in the previous section

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{i,j}x_{i,j} \\ \text{s.t.} \quad & \sum_{i<j} x_{i,j} + \sum_{j<i} x_{j,i} = 2 \quad \forall i \in V \\ & x_{i,j} \in \{0,1\} \quad \forall (i,j) \in E \end{aligned} \tag{P}$$

In previous section, we obtained several upper bounds for the problem by applying an heuristic procedure. In this section, we will solve some relaxations of problem (P) in order to obtain lower bounds.

In order to find lower bounds we will first solve the following linear relaxation (RP1) of the problem in which Subtour Elimination Constraints (SECs) are not taken into account.

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{i,j}x_{i,j} \\ \text{s.t.} \quad & \sum_{i<j} x_{i,j} + \sum_{j<i} x_{j,i} = 2 \quad \forall i \in V \\ & 0 \leq x_{i,j} \leq 1 \end{aligned} \tag{RP1}$$

Afterwards, we will firstly add to problem (RP1) a violated SEC and solve the new relaxed problem (RP2). Secondly, we will add to problem (RP2) a violated valid inequality (different from SECs) and solve the new relaxed problem (RP3). Thirdly, we will add to problem (RP3) a gomory cut and solve the new relaxed problem (RP4). Finally, we will add to problem (RP4) a new violated SEC and solve the new relaxed problem (RP5).

We notice that the problem could be solved by only adding SECs to the initial problem formulation (RP1). However, we have decided to apply three different procedures (SEC, valid inequality and gomory cut) to improve the lower bound so as to have a better understanding of those procedures.

In order to solve the different relaxed problems we have used AMPL-CPLEX. However, in order to compute the gomory cut for problem (RP4) we have used the Python API for CPLEX.

The following solution has been obtained for (RP1),

$$x^{1*} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad z^{1*} = 3.249 \tag{7}$$

Figure (4) shows the graphical solution of (RP1). As it can be seen, the solution of problem (RP1) is not a solution for the original TSP problem.

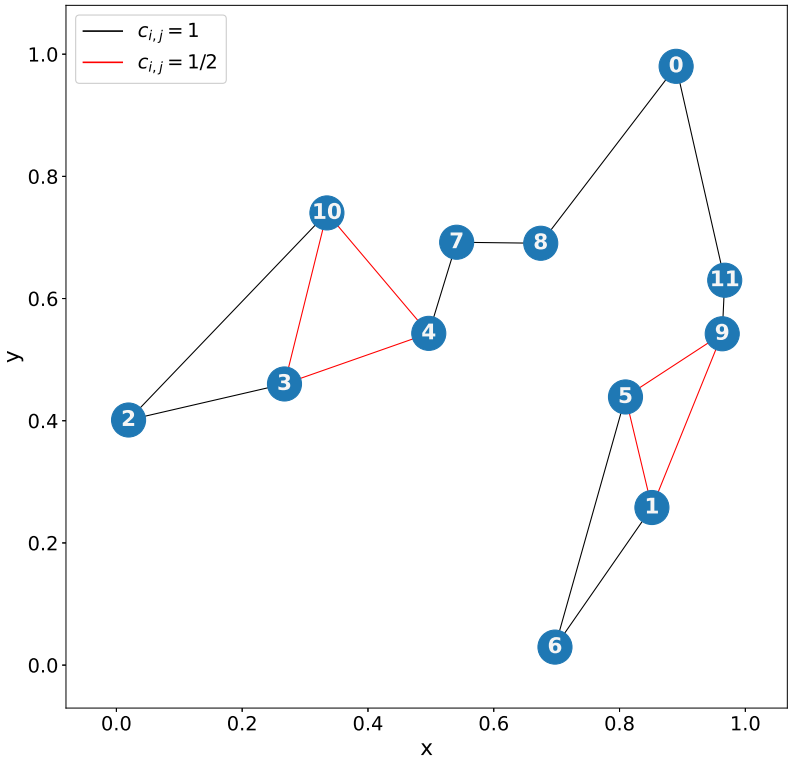

Figure 4: (RP1) solution

3.1 First Subtour Elimination Constraint (SEC-1)

We look for a Subtour Elimination Constraint (SEC) violated by x^{1*} and add it to the formulation of (RP1). In order to find it we will use the SEC identification heuristic method.

SEC identification heuristic method: Let x^* be the current solution and $G(x^*)$ the graph induced by x^*.

1. If there exist (i,j) such that $x^*_{i,j} > 1$, then the nodes involved in (i,j) define the SEC violated by x^*.
2. Else if $x^*_{i,j} < 1, \forall (i,j) \in G(x^*)$, then there are not violated SEC in the graph detected by the heuristic method.
3. Else if there exists (i,j) such that $x^*_{i,j} = 1$ then we generate a pseudo-node $\hat{i}_{i,j}$ that results from aggregating i and j and
   - If for some node k, $x^*_{i,k} > 0$ and $x^*_{j,k} > 0$ then we create an edge $(\hat{i}_{i,j}, k)$ with value $x^*_{\hat{i}_{i,j},k} = x^*_{i,k} + x^*_{j,k}$.
   - If for some node k, $x^*_{i,k} > 0$ or $x^*_{j,k} > 0$ then we create an edge $(\hat{i}_{i,j}, k)$ with value $x^*_{\hat{i}_{i,j},k} = x^*_{i,k}$ or $x^*_{\hat{i},k} = x^*_{j,k}$.

Applying the SEC identification heuristic method to $G(x^{1*})$ we observe that the subset $S_1 = \{2,3,10\} \subset V$ define a SEC violated by x^{1*}: if we consider the pseudo-node resulting from aggregating nodes 2 and 3, $\hat{i}_{2,3}$, we observe that the edge $(\hat{i}_{2,3}, 10)$ has $x^*_{\hat{i}_{2,3},10} = 1.5 > 1$.

Therefore, the SEC violated by x^{1*} and given by S_1 (SEC-1) is

$$\sum_{\substack{(i,j)\in S_1 \\ i<j}} x_{i,j} \leq |S_1| - 1 \qquad \text{(SEC-1)}$$

We add (SEC-1) to the formulation of (RP1) obtaining (RP2)

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{i,j} x_{i,j} \\ \text{s.t.} \quad & \sum_{i<j} x_{i,j} + \sum_{j<i} x_{j,i} = 2 \quad \forall i \in V \\ & \text{(SEC-1)} \\ & 0 \leq x_{i,j} \leq 1 \end{aligned} \qquad \text{(RP2)}$$

The following solution has been obtained for (RP2),

$$x^{2*} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 1 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad z^{2*} = 3.255 \tag{8}$$

Figure (5) shows the graphical solution of (RP2).

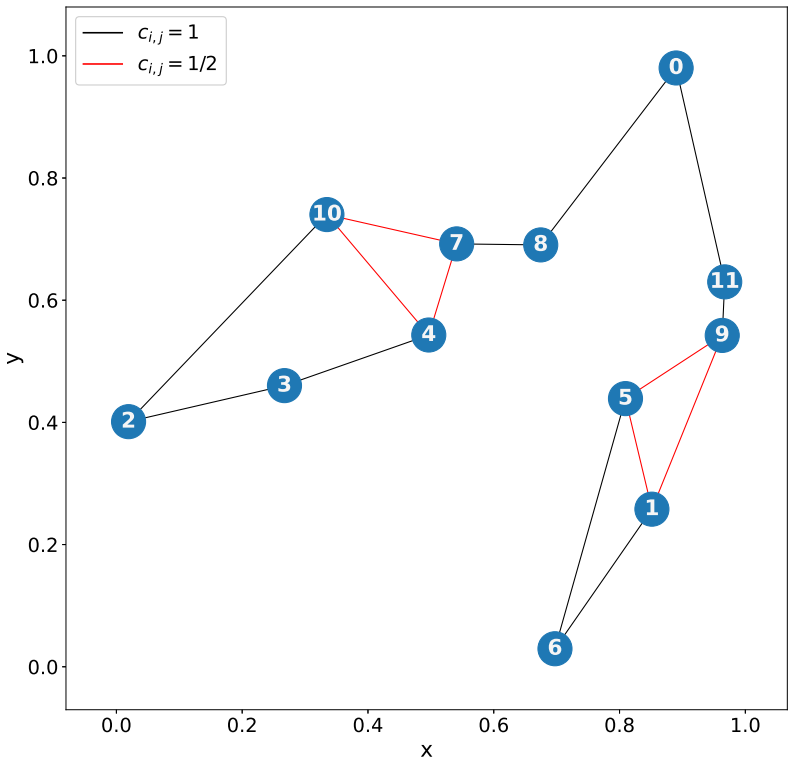

Figure 5: (RP2) solution

Again, solution to (RP2) is still not being a solution to the original TSP problem.

3.2 Valid Inequality (different from SECs)

Now we look for some valid inequality for the TSP (different from the SEC's) which be violated by x^{2*}. In order to find it, we will compute a Chvátal-Gomory inequality violated by x^{2*}.

Step 1: We sum up the degree equations for the set of nodes in $H = \{1, 5, 9\}$. Let $c(i)$ be the degree equation associated with node i, that is

$$c(i) : \sum_{i<j} x_{ij} + \sum_{j<i} x_{ji} = 2 \tag{9}$$

Then, we have

$$c(1) + c(5) + c(9) = 2 \times 3 \tag{10}$$

Step 2: Since, all variables x_{ij} satisfy $0 \leq x_{ij} \leq 1$, then from Eq. (10) we have the inequality

$$2 \sum_{\substack{(i,j)\in H \\ i<j}} x_{i,j} + x_{1,6} + x_{5,6} + x_{9,11} \leq 2 \times 3 \tag{11}$$

Step 3: We sum to Eq. (11) the inequalities (upper bounds)

$$x_{1,6} \leq 1, \quad x_{5,6} \leq 1, \quad x_{9,11} \leq 1 \tag{12}$$

and we obtain the following inequality

$$2 \sum_{\substack{(i,j)\in H \\ i<j}} x_{i,j} + 2(x_{1,6} + x_{5,6} + x_{9,11}) \leq 2 \times 3 + 3 \tag{13}$$

Multiplying Eq. (13) by $1/2$ and since

$$\sum_{\substack{(i,j)\in H \\ i<j}} x_{i,j} = x_{1,5} + x_{1,9} + x_{5,9} \tag{14}$$

we obtain

$$x_{1,5} + x_{1,9} + x_{5,9} + x_{1,6} + x_{5,6} + x_{9,11} \leq 3 + 3/2 \tag{15}$$

Now, since the the left hand side for the TSP should be integer-valued we can round down the right hand side to obtain a valid inequality for the STP, which in fact is violated by x_{2*}

$$x_{1,5} + x_{1,9} + x_{5,9} + x_{1,6} + x_{5,6} + x_{9,11} \leq 4 \tag{INEQ-1}$$

Therefore, we add (INEQ-1) to the formulation of (RP2) in order to obtain (RP3). We notice that (INEQ-1) belongs to a type of inequalities known as *Comb inequalities*.

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in E} c_{i,j}x_{i,j} \\
\text{s.t.} \quad & \sum_{i<j} x_{i,j} + \sum_{j<i} x_{j,i} = 2 \quad \forall i \in V \\
& \text{(SEC-1)} \\
& \text{(INEQ-1)} \\
& 0 \le x_{i,j} \le 1
\end{aligned}
\tag{RP3}
$$

The following solution has been obtained for (RP3),

$$
x^{3*} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1/2 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}, \quad z^{3*} = 3.2715 \tag{16}
$$

Figure (6) shows the graphical solution of (RP3).

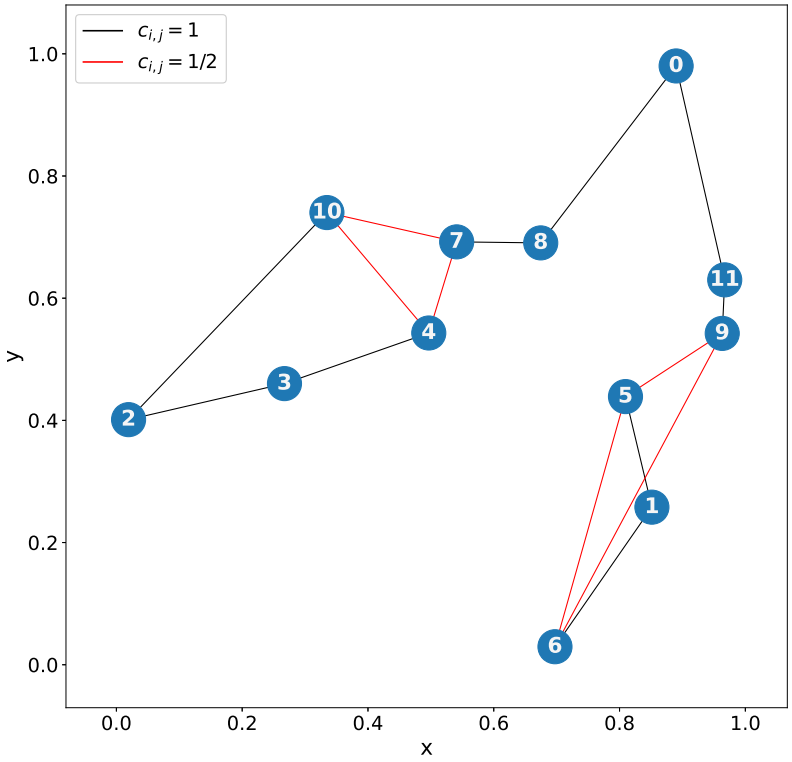

Figure 6: (RP3) solution

Again, solution to (RP3) is still not being a solution to the original TSP problem.

3.3 Gomory Cut

Now, we try to identify some Gomory cut violated by x^{3*}. In order to compute it, we have used the Python API for CPLEX and intance methods *binvarow()* and *basis.write()* which provide the $i-th$ row of the optimal LP tableau and basic variables information, respectively.

In order to use the Python API for CPLEX we have firstly export the AMPL model into a *.mps* format file (Appendix: tsp.mps). The code below, Listing (3) shows the python code used.

Listing 3: Code section 3 (Gomory cut)

```
1 import cplex
2 import numpy as np
3
```

```
# Load model( file .mps)
c = cplex.Cplex()
out = c.set_results_stream(None)
out = c.set_log_stream(None)
c.read("tsp.mps")

#Solve problem
c.solve()

# Compute Simplex Tableau
Tableau = np.zeros((14,132));
cont = 0;
for tableau_row in c.solution.advanced.binvarow():
    Tableau[cont,:] = tableau_row;
    cont = cont+1;

# Basis info
c.solution.basis.write("basis.mps")

#Solution
SOL = c.solution.get_values();
```

We will compute the gomory cut associated with the basic variable $x_{5,8}$ which has a non-integer value ($x^{3*}_{5,8} = 1/2$). The 8-row is the one associated with basic variable $x_{5,8}$ (Appendix: .col + basis.mps). Then, according to the Tableau, the following equality must be satisfied by any feasible solution (in particular by x^{3*})

$$x_{5,8} - \frac{1}{2}(x_{1,3} + x_{1,4} + x_{1,11} + x_{2,3} + x_{2,4} + x_{2,11} + x_{3,6} + x_{3,7} + x_{3,9} + x_{3,10} + x_{3,12} + x_{4,6} + x_{4,7} + \\ + x_{4,9} + x_{4,10} + x_{4,12} + x_{6,11} + x_{7,11} + x_{9,11} + x_{10,11} + x_{11,12}) + \\ + \frac{1}{2}(x_{1,5} + x_{1,8} + x_{2,5} + x_{2,8} + x_{5,6} + x_{5,7} + x_{5,9} + x_{5,10} + x_{5,12} + x_{6,8} + + x_{7,8} + x_{8,9} + x_{8,10} + x_{8,12}) = 1/2 \tag{17}$$

In order to compute the Gomory cut (GC-1), we notice that variables $x_{i,j}$ are integer and non-negative and we round down the right hand side and coefficients on the left hand side and we obtain,

$$x_{5,8} - \frac{1}{2}(x_{1,3} + x_{1,4} + x_{1,11} + x_{2,3} + x_{2,4} + x_{2,11} + x_{3,6} + x_{3,7} + x_{3,9} + x_{3,10} + x_{3,12} + x_{4,6} + x_{4,7} + \\ + x_{4,9} + x_{4,10} + x_{4,12} + x_{6,11} + x_{7,11} + x_{9,11} + x_{10,11} + x_{11,12}) \leq 0 \tag{18}$$

Therefore, we add (GC-1) to the formulation of (RP3) in order to obtain (RP4).

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{i,j}x_{i,j} \\ \text{s.t.} \quad & \sum_{i<j} x_{i,j} + \sum_{j<i} x_{j,i} = 2 \quad \forall i \in V \\ & \text{(SEC-1)} \\ & \text{(INEQ-1)} \\ & \text{(GC-1)} \\ & 0 \le x_{i,j} \le 1 \end{aligned} \tag{P3}$$

The following solution has been obtained to (RP4), (Appendix: .sol)

$$x^{4*} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1/3 & 0 & 0 & 0 & 2/3 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & & 1/3 & 0 & 0 & 2/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 & 2/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 2/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad z^{4*} = 3.29 \tag{19}$$

Figure (7) shows the graphical solution of (RP4).

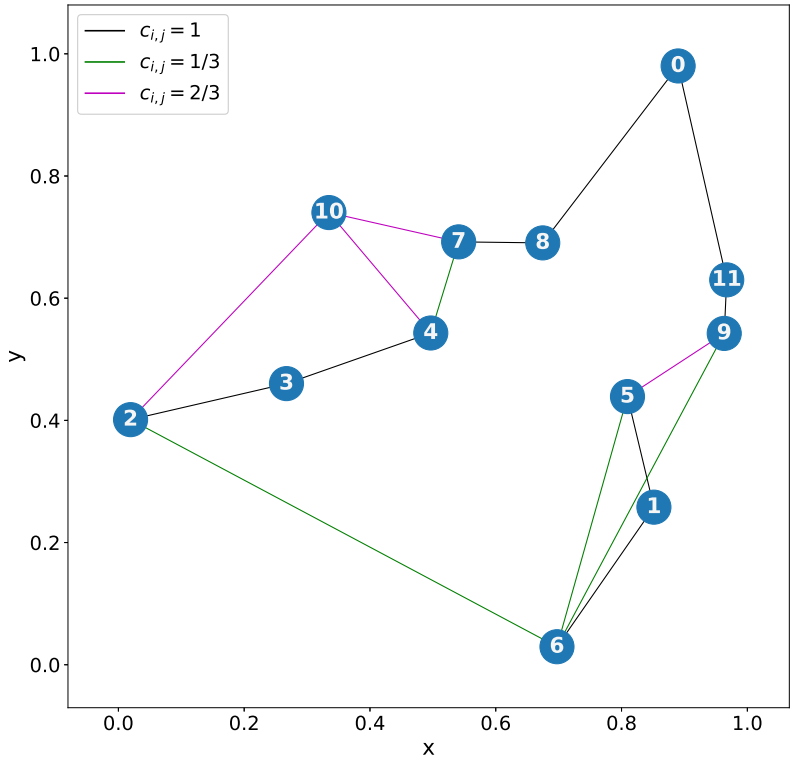

Figure 7: (RP4) solution

Again, solution to (RP4) is still not being a solution to the original TSP problem.

3.4 Second Subtour Elimination constraint (SEC-2)

We look for a Subtour Elimination Constraint (SEC) violated by x^{4*} and add it to the formulation of (RP4). In order to find it we will use the SEC identification heuristic method previously explained.

Applying the SEC identification heuristic method to $G(x^{4*})$ we observe that the subset $S_2 = \{1, 5, 6\} \subset V$ defines a SEC violated by x^{4*}: if we consider the pseudo-node resulting from aggregating nodes 1 and 6, $\hat{i}_{1,6}$, we observe that the edge $(\hat{i}_{1,6}, 5)$ has $x^*_{\hat{i}_{1,6},5} = 1 + 1/3 > 1$.

Therefore, the SEC violated by x^{4*} and given by S_2 (SEC-2) is

$$\sum_{\substack{(i,j)\in S_2 \\ i<j}} x_{i,j} \leq |S_2| - 1 \qquad \text{(SEC-2)}$$

We add (SEC-2) to the formulation of (RP4) obtaining (RP5)

$$\begin{aligned} \min \quad & \sum_{(i,j)\in E} c_{i,j}x_{i,j} \\ \text{s.t.} \quad & \sum_{i<j} x_{i,j} + \sum_{j<i} x_{j,i} = 2 \quad \forall i \in V \\ & \text{(SEC-1)} \\ & \text{(INEQ-1)} \\ & \text{(GC-1)} \\ & \text{(SEC-2)} \\ & 0 \leq x_{i,j} \leq 1 \end{aligned} \tag{RP5}$$

The following solution has been obtained for (RP5),

$$x^{5*} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad z^{5*} = 3.314 \tag{20}$$

Clearly, solution to (RP5) is the solution to the TSP as it is an integer solution and no SEC is violated. Moreover, for this solution the lower bound obtained equals the best upper bound obtained in the previous section

$$z^* = \overline{z^*} = \underline{z^*} \tag{21}$$

which shows optimality.

Figure (8) shows the corresponding lower bounds for the TSP obtained in the relaxed problems (RP1), (RP2), (RP3), (RP4) and (RP5) just solved.

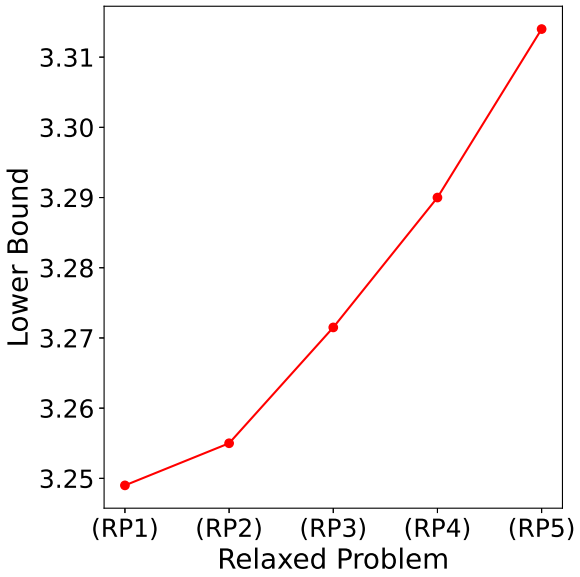

Figure 8: TSP Lower bounds for the different relaxed problems solved.

As expected, as the problem becomes more restrictive, lower bounds increase.

Below, we show the main AMPL-CPLEX files (.mod, .dat) used in this section.

Listing 4: Section 3 (.mod)

```
# parameters
param n;
set nodes ordered := {1..n};
set edges := {i in nodes, j in nodes};

set nodes1 ordered := {3,4,11};
set edges1 := {i in nodes1, j in nodes1};

set nodes2 ordered := {2,6,7};
set edges2 := {i in nodes2, j in nodes2};

param c {(i,j) in edges};
var X {(i,j) in edges} >=0, <=1;

#Objective function
minimize fobj : sum {(i,j) in edges} (c[i,j]*X[i,j]);

#Constraints
subject to cc1 {i in nodes}: sum{(i,j) in edges: i < j} (X[i,j]) + sum{(i,j) in edges: j < i}
    (X[j,i])= 2;
```

subject to cc2: **sum**{(i,j) **in** edges1:i < j} (X[i,j]) <= 2;
subject to cc3: X[2,6]+X[2,10]+X[6,10]+X[2,7]+X[6,7]+X[10,12] <= 4;
subject to cc4: X[5,8] −
X[1,3]−X[1,4]−X[1,11]−X[2,3]−X[2,4]−X[2,11]−X[3,6]−X[3,7]−X[3,9]−X[3,10]−X[3,12]−X[4,6]−
−X[4,7]−X[4,9]−X[4,10]−X[4,12]−X[6,11]−X[7,11]−X[9,11]−X[10,11]−X[11,12] <= 0;
subject to cc5: **sum**{(i,j) **in** edges2:i < j} (X[i,j]) <= 2;

Listing 5: Section 3 (.dat)

param n := 12;

param c: 1 2 3 4 5 6 7 8 9 10 11 12 :=

1 0. 0.723 1.046 0.811 0.588 0.547 0.97 0.452 0.361 0.444 0.605 0.359
2 0.723 0. 0.844 0.618 0.455 0.186 0.276 0.534 0.467 0.306 0.707 0.39
3 1.046 0.844 0. 0.255 0.498 0.791 0.773 0.597 0.716 0.954 0.463 0.975
4 0.811 0.618 0.255 0. 0.244 0.543 0.609 0.359 0.468 0.701 0.288 0.72
5 0.588 0.455 0.498 0.244 0. 0.33 0.551 0.156 0.231 0.467 0.255 0.479
6 0.547 0.186 0.791 0.543 0.33 0. 0.425 0.369 0.285 0.185 0.562 0.248
7 0.97 0.276 0.773 0.609 0.551 0.425 0. 0.681 0.661 0.578 0.798 0.658
8 0.452 0.534 0.597 0.359 0.156 0.369 0.681 0. 0.134 0.448 0.212 0.431
9 0.361 0.467 0.716 0.468 0.231 0.285 0.661 0.134 0. 0.324 0.344 0.299
10 0.444 0.306 0.954 0.701 0.467 0.185 0.578 0.448 0.324 0. 0.659 0.088
11 0.605 0.707 0.463 0.288 0.255 0.562 0.798 0.212 0.344 0.659 0. 0.642
12 0.359 0.39 0.975 0.72 0.479 0.248 0.658 0.431 0.299 0.088 0.642 0.
;

4 Appendix

4.1 Section 2

Complete list of successive tours with its associated cost obtained by the heuristic explained in section 2.

Listing 6: Heuristic method

```
NN(0):
[0, 11, 9, 5, 1, 6, 4, 7, 8, 10, 3, 2, 0] -- z* = 3.8679999999999994
[0, 11, 9, 5, 1, 6, 2, 3, 10, 8, 7, 4, 0] -- z* = 3.6319999999999997
[0, 11, 9, 5, 1, 6, 2, 3, 4, 7, 8, 10, 0] -- z* = 3.6049999999999995
[0, 11, 9, 5, 1, 6, 2, 3, 4, 8, 7, 10, 0] -- z* = 3.5479999999999996
[0, 11, 9, 5, 1, 6, 2, 3, 4, 10, 7, 8, 0] -- z* = 3.328
NN(1):
```

```
[1, 5, 9, 11, 8, 7, 4, 3, 2, 10, 0, 6, 1] -- z* = 3.8609999999999998
[1, 0, 10, 2, 3, 4, 7, 8, 11, 9, 5, 6, 1] -- z* = 3.8529999999999998
NN(2):
[2, 3, 4, 7, 8, 5, 9, 11, 0, 10, 1, 6, 2] -- z* = 4.066999999999999
[2, 3, 10, 0, 11, 9, 5, 8, 7, 4, 1, 6, 2] -- z* = 3.858999999999999
[2, 3, 10, 4, 7, 8, 5, 9, 11, 0, 1, 6, 2] -- z* = 3.7769999999999992
[2, 3, 10, 4, 7, 8, 0, 11, 9, 5, 1, 6, 2] -- z* = 3.315999999999999
NN(3):
[3, 4, 7, 8, 5, 9, 11, 0, 10, 2, 6, 1, 3] -- z* = 4.186000000000001
[3, 2, 10, 0, 11, 9, 5, 8, 7, 4, 6, 1, 3] -- z* = 3.975000000000001
[3, 2, 10, 0, 11, 9, 5, 1, 6, 4, 7, 8, 3] -- z* = 3.726000000000001
[3, 2, 10, 0, 11, 9, 5, 1, 6, 8, 7, 4, 3] -- z* = 3.612000000000001
NN(4):
[4, 7, 8, 5, 9, 11, 0, 10, 3, 2, 6, 1, 4] -- z* = 3.859
[4, 7, 8, 5, 9, 11, 0, 1, 6, 2, 3, 10, 4] -- z* = 3.777
NN(5):
[5, 9, 11, 8, 7, 4, 3, 2, 10, 0, 1, 6, 5] -- z* = 3.8529999999999998
[5, 0, 10, 2, 3, 4, 7, 8, 11, 9, 1, 6, 5] -- z* = 3.798
NN(6):
[6, 1, 5, 9, 11, 8, 7, 4, 3, 2, 10, 0, 6] -- z* = 3.8609999999999998
[6, 1, 0, 10, 2, 3, 4, 7, 8, 11, 9, 5, 6] -- z* = 3.8529999999999998
NN(7):
[7, 8, 4, 3, 2, 10, 5, 9, 11, 0, 1, 6, 7] -- z* = 4.201
[7, 10, 2, 3, 4, 8, 5, 9, 11, 0, 1, 6, 7] -- z* = 4.002
[7, 10, 2, 3, 6, 1, 0, 11, 9, 5, 8, 4, 7] -- z* = 3.842
[7, 10, 2, 3, 6, 1, 9, 11, 0, 5, 8, 4, 7] -- z* = 3.787
[7, 10, 2, 3, 6, 1, 5, 0, 11, 9, 8, 4, 7] -- z* = 3.7059999999999995
[7, 10, 2, 3, 6, 1, 5, 9, 11, 0, 8, 4, 7] -- z* = 3.3809999999999993
NN(8):
[8, 7, 4, 3, 2, 10, 5, 9, 11, 0, 1, 6, 8] -- z* = 4.106
[8, 7, 10, 2, 3, 4, 5, 9, 11, 0, 1, 6, 8] -- z* = 3.9299999999999997
[8, 7, 10, 2, 3, 6, 1, 0, 11, 9, 5, 4, 8] -- z* = 3.8649999999999993
[8, 7, 10, 2, 3, 6, 1, 9, 11, 0, 5, 4, 8] -- z* = 3.8099999999999996
NN(9):
[9, 11, 5, 1, 6, 4, 7, 8, 10, 3, 2, 0, 9] -- z* = 4.016
[9, 11, 0, 2, 3, 10, 8, 7, 4, 6, 1, 5, 9] -- z* = 3.868
[9, 11, 0, 3, 2, 10, 8, 7, 4, 6, 1, 5, 9] -- z* = 3.808
[9, 11, 0, 10, 2, 3, 8, 7, 4, 6, 1, 5, 9] -- z* = 3.7259999999999995
[9, 11, 0, 8, 3, 2, 10, 7, 4, 6, 1, 5, 9] -- z* = 3.5599999999999996
[9, 11, 0, 8, 7, 10, 2, 3, 4, 6, 1, 5, 9] -- z* = 3.3139999999999996
NN(10):
[10, 7, 8, 4, 3, 2, 6, 1, 5, 9, 11, 0, 10] -- z* = 3.548
[10, 7, 8, 0, 11, 9, 5, 1, 6, 2, 3, 4, 10] -- z* = 3.3280000000000003
[10, 7, 8, 0, 11, 9, 5, 1, 6, 4, 3, 2, 10] -- z* = 3.314
NN(11):
```

```
[11, 9, 5, 1, 6, 4, 7, 8, 10, 3, 2, 0, 11] -- z* = 3.8679999999999994
[11, 9, 5, 1, 6, 2, 3, 10, 8, 7, 4, 0, 11] -- z* = 3.6319999999999997
[11, 9, 5, 1, 6, 2, 3, 4, 7, 8, 10, 0, 11] -- z* = 3.6049999999999995
[11, 9, 5, 1, 6, 2, 3, 4, 8, 7, 10, 0, 11] -- z* = 3.5479999999999996
[11, 9, 5, 1, 6, 2, 3, 4, 10, 7, 8, 0, 11] -- z* = 3.328
```

4.2 Section 3

Listing 7: tsp.mps

```
NAME          tsp
ROWS
 E  R0001
 E  R0002
 E  R0003
 E  R0004
 E  R0005
 E  R0006
 E  R0007
 E  R0008
 E  R0009
 E  R0010
 E  R0011
 E  R0012
 L  R0013
 L  R0014
 N  R0015
COLUMNS
    C0001     R0001     1
    C0001     R0002     1
    C0001     R0015     0.723
    C0002     R0001     1
    C0002     R0003     1
    C0002     R0015     1.046
    C0003     R0001     1
    C0003     R0004     1
    C0003     R0015     0.811
    C0004     R0001     1
    C0004     R0005     1
    C0004     R0015     0.588
    C0005     R0001     1
    C0005     R0006     1
    C0005     R0015     0.547
    C0006     R0001     1
    C0006     R0007     1
```

```
C0006     R0015     0.97
C0007     R0001     1
C0007     R0008     1
C0007     R0015     0.452
C0008     R0001     1
C0008     R0009     1
C0008     R0015     0.361
C0009     R0001     1
C0009     R0010     1
C0009     R0015     0.444
C0010     R0001     1
C0010     R0011     1
C0010     R0015     0.605
C0011     R0001     1
C0011     R0012     1
C0011     R0015     0.359
C0012     R0015     0.723
C0013     R0002     1
C0013     R0003     1
C0013     R0015     0.844
C0014     R0002     1
C0014     R0004     1
C0014     R0015     0.618
C0015     R0002     1
C0015     R0005     1
C0015     R0015     0.455
C0016     R0002     1
C0016     R0006     1
C0016     R0014     1
C0016     R0015     0.186
C0017     R0002     1
C0017     R0007     1
C0017     R0014     1
C0017     R0015     0.276
C0018     R0002     1
C0018     R0008     1
C0018     R0015     0.534
C0019     R0002     1
C0019     R0009     1
C0019     R0015     0.467
C0020     R0002     1
C0020     R0010     1
C0020     R0014     1
C0020     R0015     0.306
C0021     R0002     1
```

C0021	R0011	1
C0021	R0015	0.707
C0022	R0002	1
C0022	R0012	1
C0022	R0015	0.39
C0023	R0015	1.046
C0024	R0015	0.844
C0025	R0003	1
C0025	R0004	1
C0025	R0013	1
C0025	R0015	0.255
C0026	R0003	1
C0026	R0005	1
C0026	R0015	0.498
C0027	R0003	1
C0027	R0006	1
C0027	R0015	0.791
C0028	R0003	1
C0028	R0007	1
C0028	R0015	0.773
C0029	R0003	1
C0029	R0008	1
C0029	R0015	0.597
C0030	R0003	1
C0030	R0009	1
C0030	R0015	0.716
C0031	R0003	1
C0031	R0010	1
C0031	R0015	0.954
C0032	R0003	1
C0032	R0011	1
C0032	R0013	1
C0032	R0015	0.463
C0033	R0003	1
C0033	R0012	1
C0033	R0015	0.975
C0034	R0015	0.811
C0035	R0015	0.618
C0036	R0015	0.255
C0037	R0004	1
C0037	R0005	1
C0037	R0015	0.244
C0038	R0004	1
C0038	R0006	1
C0038	R0015	0.543

C0039	R0004	1
C0039	R0007	1
C0039	R0015	0.609
C0040	R0004	1
C0040	R0008	1
C0040	R0015	0.359
C0041	R0004	1
C0041	R0009	1
C0041	R0015	0.468
C0042	R0004	1
C0042	R0010	1
C0042	R0015	0.701
C0043	R0004	1
C0043	R0011	1
C0043	R0013	1
C0043	R0015	0.288
C0044	R0004	1
C0044	R0012	1
C0044	R0015	0.72
C0045	R0015	0.588
C0046	R0015	0.455
C0047	R0015	0.498
C0048	R0015	0.244
C0049	R0005	1
C0049	R0006	1
C0049	R0015	0.33
C0050	R0005	1
C0050	R0007	1
C0050	R0015	0.551
C0051	R0005	1
C0051	R0008	1
C0051	R0015	0.156
C0052	R0005	1
C0052	R0009	1
C0052	R0015	0.231
C0053	R0005	1
C0053	R0010	1
C0053	R0015	0.467
C0054	R0005	1
C0054	R0011	1
C0054	R0015	0.255
C0055	R0005	1
C0055	R0012	1
C0055	R0015	0.479
C0056	R0015	0.547

C0057	R0015	0.186
C0058	R0015	0.791
C0059	R0015	0.543
C0060	R0015	0.33
C0061	R0006	1
C0061	R0007	1
C0061	R0014	1
C0061	R0015	0.425
C0062	R0006	1
C0062	R0008	1
C0062	R0015	0.369
C0063	R0006	1
C0063	R0009	1
C0063	R0015	0.285
C0064	R0006	1
C0064	R0010	1
C0064	R0014	1
C0064	R0015	0.185
C0065	R0006	1
C0065	R0011	1
C0065	R0015	0.562
C0066	R0006	1
C0066	R0012	1
C0066	R0015	0.248
C0067	R0015	0.97
C0068	R0015	0.276
C0069	R0015	0.773
C0070	R0015	0.609
C0071	R0015	0.551
C0072	R0015	0.425
C0073	R0007	1
C0073	R0008	1
C0073	R0015	0.681
C0074	R0007	1
C0074	R0009	1
C0074	R0015	0.661
C0075	R0007	1
C0075	R0010	1
C0075	R0015	0.578
C0076	R0007	1
C0076	R0011	1
C0076	R0015	0.798
C0077	R0007	1
C0077	R0012	1
C0077	R0015	0.658

C0078	R0015	0.452
C0079	R0015	0.534
C0080	R0015	0.597
C0081	R0015	0.359
C0082	R0015	0.156
C0083	R0015	0.369
C0084	R0015	0.681
C0085	R0008	1
C0085	R0009	1
C0085	R0015	0.134
C0086	R0008	1
C0086	R0010	1
C0086	R0015	0.448
C0087	R0008	1
C0087	R0011	1
C0087	R0015	0.212
C0088	R0008	1
C0088	R0012	1
C0088	R0015	0.431
C0089	R0015	0.361
C0090	R0015	0.467
C0091	R0015	0.716
C0092	R0015	0.468
C0093	R0015	0.231
C0094	R0015	0.285
C0095	R0015	0.661
C0096	R0015	0.134
C0097	R0009	1
C0097	R0010	1
C0097	R0015	0.324
C0098	R0009	1
C0098	R0011	1
C0098	R0015	0.344
C0099	R0009	1
C0099	R0012	1
C0099	R0015	0.299
C0100	R0015	0.444
C0101	R0015	0.306
C0102	R0015	0.954
C0103	R0015	0.701
C0104	R0015	0.467
C0105	R0015	0.185
C0106	R0015	0.578
C0107	R0015	0.448
C0108	R0015	0.324

```
    C0109     R0010     1
    C0109     R0011     1
    C0109     R0015     0.659
    C0110     R0010     1
    C0110     R0012     1
    C0110     R0014     1
    C0110     R0015     0.088
    C0111     R0015     0.605
    C0112     R0015     0.707
    C0113     R0015     0.463
    C0114     R0015     0.288
    C0115     R0015     0.255
    C0116     R0015     0.562
    C0117     R0015     0.798
    C0118     R0015     0.212
    C0119     R0015     0.344
    C0120     R0015     0.659
    C0121     R0011     1
    C0121     R0012     1
    C0121     R0015     0.642
    C0122     R0015     0.359
    C0123     R0015     0.39
    C0124     R0015     0.975
    C0125     R0015     0.72
    C0126     R0015     0.479
    C0127     R0015     0.248
    C0128     R0015     0.658
    C0129     R0015     0.431
    C0130     R0015     0.299
    C0131     R0015     0.088
    C0132     R0015     0.642
RHS
    B         R0001     2
    B         R0002     2
    B         R0003     2
    B         R0004     2
    B         R0005     2
    B         R0006     2
    B         R0007     2
    B         R0008     2
    B         R0009     2
    B         R0010     2
    B         R0011     2
    B         R0012     2
    B         R0013     2
```

```
    B         R0014     4
BOUNDS
 UP BOUND     C0001     1
 UP BOUND     C0002     1
 UP BOUND     C0003     1
 UP BOUND     C0004     1
 UP BOUND     C0005     1
 UP BOUND     C0006     1
 UP BOUND     C0007     1
 UP BOUND     C0008     1
 UP BOUND     C0009     1
 UP BOUND     C0010     1
 UP BOUND     C0011     1
 UP BOUND     C0012     1
 UP BOUND     C0013     1
 UP BOUND     C0014     1
 UP BOUND     C0015     1
 UP BOUND     C0016     1
 UP BOUND     C0017     1
 UP BOUND     C0018     1
 UP BOUND     C0019     1
 UP BOUND     C0020     1
 UP BOUND     C0021     1
 UP BOUND     C0022     1
 UP BOUND     C0023     1
 UP BOUND     C0024     1
 UP BOUND     C0025     1
 UP BOUND     C0026     1
 UP BOUND     C0027     1
 UP BOUND     C0028     1
 UP BOUND     C0029     1
 UP BOUND     C0030     1
 UP BOUND     C0031     1
 UP BOUND     C0032     1
 UP BOUND     C0033     1
 UP BOUND     C0034     1
 UP BOUND     C0035     1
 UP BOUND     C0036     1
 UP BOUND     C0037     1
 UP BOUND     C0038     1
 UP BOUND     C0039     1
 UP BOUND     C0040     1
 UP BOUND     C0041     1
 UP BOUND     C0042     1
 UP BOUND     C0043     1
```

```
UP BOUND     C0044     1
UP BOUND     C0045     1
UP BOUND     C0046     1
UP BOUND     C0047     1
UP BOUND     C0048     1
UP BOUND     C0049     1
UP BOUND     C0050     1
UP BOUND     C0051     1
UP BOUND     C0052     1
UP BOUND     C0053     1
UP BOUND     C0054     1
UP BOUND     C0055     1
UP BOUND     C0056     1
UP BOUND     C0057     1
UP BOUND     C0058     1
UP BOUND     C0059     1
UP BOUND     C0060     1
UP BOUND     C0061     1
UP BOUND     C0062     1
UP BOUND     C0063     1
UP BOUND     C0064     1
UP BOUND     C0065     1
UP BOUND     C0066     1
UP BOUND     C0067     1
UP BOUND     C0068     1
UP BOUND     C0069     1
UP BOUND     C0070     1
UP BOUND     C0071     1
UP BOUND     C0072     1
UP BOUND     C0073     1
UP BOUND     C0074     1
UP BOUND     C0075     1
UP BOUND     C0076     1
UP BOUND     C0077     1
UP BOUND     C0078     1
UP BOUND     C0079     1
UP BOUND     C0080     1
UP BOUND     C0081     1
UP BOUND     C0082     1
UP BOUND     C0083     1
UP BOUND     C0084     1
UP BOUND     C0085     1
UP BOUND     C0086     1
UP BOUND     C0087     1
UP BOUND     C0088     1
```

```
 UP BOUND     C0089     1
 UP BOUND     C0090     1
 UP BOUND     C0091     1
 UP BOUND     C0092     1
 UP BOUND     C0093     1
 UP BOUND     C0094     1
 UP BOUND     C0095     1
 UP BOUND     C0096     1
 UP BOUND     C0097     1
 UP BOUND     C0098     1
 UP BOUND     C0099     1
 UP BOUND     C0100     1
 UP BOUND     C0101     1
 UP BOUND     C0102     1
 UP BOUND     C0103     1
 UP BOUND     C0104     1
 UP BOUND     C0105     1
 UP BOUND     C0106     1
 UP BOUND     C0107     1
 UP BOUND     C0108     1
 UP BOUND     C0109     1
 UP BOUND     C0110     1
 UP BOUND     C0111     1
 UP BOUND     C0112     1
 UP BOUND     C0113     1
 UP BOUND     C0114     1
 UP BOUND     C0115     1
 UP BOUND     C0116     1
 UP BOUND     C0117     1
 UP BOUND     C0118     1
 UP BOUND     C0119     1
 UP BOUND     C0120     1
 UP BOUND     C0121     1
 UP BOUND     C0122     1
 UP BOUND     C0123     1
 UP BOUND     C0124     1
 UP BOUND     C0125     1
 UP BOUND     C0126     1
 UP BOUND     C0127     1
 UP BOUND     C0128     1
 UP BOUND     C0129     1
 UP BOUND     C0130     1
 UP BOUND     C0131     1
 UP BOUND     C0132     1
ENDATA
```

Listing 8: .col

```
X[1,2]
X[1,3]
X[1,4]
X[1,5]
X[1,6]
X[1,7]
X[1,8]
X[1,9]
X[1,10]
X[1,11]
X[1,12]
X[2,1]
X[2,3]
X[2,4]
X[2,5]
X[2,6]
X[2,7]
X[2,8]
X[2,9]
X[2,10]
X[2,11]
X[2,12]
X[3,1]
X[3,2]
X[3,4]
X[3,5]
X[3,6]
X[3,7]
X[3,8]
X[3,9]
X[3,10]
X[3,11]
X[3,12]
X[4,1]
X[4,2]
X[4,3]
X[4,5]
X[4,6]
X[4,7]
X[4,8]
X[4,9]
X[4,10]
```

X[4,11]
X[4,12]
X[5,1]
X[5,2]
X[5,3]
X[5,4]
X[5,6]
X[5,7]
X[5,8]
X[5,9]
X[5,10]
X[5,11]
X[5,12]
X[6,1]
X[6,2]
X[6,3]
X[6,4]
X[6,5]
X[6,7]
X[6,8]
X[6,9]
X[6,10]
X[6,11]
X[6,12]
X[7,1]
X[7,2]
X[7,3]
X[7,4]
X[7,5]
X[7,6]
X[7,8]
X[7,9]
X[7,10]
X[7,11]
X[7,12]
X[8,1]
X[8,2]
X[8,3]
X[8,4]
X[8,5]
X[8,6]
X[8,7]
X[8,9]
X[8,10]
X[8,11]

X[8,12]
X[9,1]
X[9,2]
X[9,3]
X[9,4]
X[9,5]
X[9,6]
X[9,7]
X[9,8]
X[9,10]
X[9,11]
X[9,12]
X[10,1]
X[10,2]
X[10,3]
X[10,4]
X[10,5]
X[10,6]
X[10,7]
X[10,8]
X[10,9]
X[10,11]
X[10,12]
X[11,1]
X[11,2]
X[11,3]
X[11,4]
X[11,5]
X[11,6]
X[11,7]
X[11,8]
X[11,9]
X[11,10]
X[11,12]
X[12,1]
X[12,2]
X[12,3]
X[12,4]
X[12,5]
X[12,6]
X[12,7]
X[12,8]
X[12,9]
X[12,10]
X[12,11]

Listing 9: .row

```
cc1[1]
cc1[2]
cc1[3]
cc1[4]
cc1[5]
cc1[6]
cc1[7]
cc1[8]
cc1[9]
cc1[10]
cc1[11]
cc1[12]
cc2
cc3
fobj
```

Listing 10: basis.mps

```
* ENCODING=ISO-8859-1
NAME          tsp.mps  Iterations 26  Rows 14  Cols 132
 XL C0008     R0001
 XL C0009     R0002
 XL C0016     R0003
 XL C0020     R0004
 XL C0032     R0005
 XL C0037     R0006
 XL C0043     R0007
 XL C0051     R0008
 XL C0054     R0009
 XL C0061     R0010
 XL C0064     R0011
 XL C0066     R0012
 XL C0075     R0013
 XL C0087     R0014
 UL C0011
 UL C0017
 UL C0025
 UL C0085
 UL C0110
ENDATA
```

Listing 11: .sol

```
<?xml version = "1.0" encoding="UTF-8" standalone="yes"?>
<CPLEXSolution version="1.2">
 <header
   problemName="tsp.mps"
   objectiveValue="3.2714999999999996"
   solutionTypeValue="1"
   solutionTypeString="basic"
   solutionStatusValue="1"
   solutionStatusString="optimal"
   solutionMethodString="dual"
   primalFeasible="1"
   dualFeasible="1"
   simplexIterations="26"
   writeLevel="1"/>
 <quality
   epRHS="9.9999999999999995e-07"
   epOpt="9.9999999999999995e-07"
   maxPrimalInfeas="0"
   maxDualInfeas="0"
   maxPrimalResidual="0"
   maxDualResidual="2.7755575615628914e-17"
   maxX="1"
   maxPi="0.40899999999999997"
   maxSlack="0"
   maxRedCost="1.046"
   kappa="10.5"/>
 <linearConstraints>
  <constraint name="R0001" index="0" status="LL" slack="0" dual="0.27500000000000002"/>
  <constraint name="R0002" index="1" status="LL" slack="0" dual="0.16999999999999998"/>
  <constraint name="R0003" index="2" status="LL" slack="0" dual="0.31950000000000001"/>
  <constraint name="R0004" index="3" status="LL" slack="0" dual="0.14449999999999999"/>
  <constraint name="R0005" index="4" status="LL" slack="0" dual="0.099500000000000005"/>
  <constraint name="R0006" index="5" status="LL" slack="0" dual="0.048999999999999988"/>
  <constraint name="R0007" index="6" status="LL" slack="0" dual="0.40899999999999997"/>
  <constraint name="R0008" index="7" status="LL" slack="0" dual="0.056499999999999995"/>
  <constraint name="R0009" index="8" status="LL" slack="0" dual="0.085999999999999965"/>
  <constraint name="R0010" index="9" status="LL" slack="0" dual="0.16899999999999998"/>
  <constraint name="R0011" index="10" status="LL" slack="0" dual="0.1555"/>
  <constraint name="R0012" index="11" status="LL" slack="0" dual="0.19900000000000001"/>
  <constraint name="R0013" index="12" status="LL" slack="0" dual="-0.012000000000000011"/>
  <constraint name="R0014" index="13" status="LL" slack="0" dual="-0.032999999999999974"/>
 </linearConstraints>
 <variables>
  <variable name="C0001" index="0" status="LL" value="0" reducedCost="0.27799999999999997"/>
  <variable name="C0002" index="1" status="LL" value="0" reducedCost="0.45150000000000001"/>
```

<variable name="C0003" index="2" status="LL" value="0" reducedCost="0.39150000000000007"/>
<variable name="C0004" index="3" status="LL" value="0" reducedCost="0.21349999999999994"/>
<variable name="C0005" index="4" status="LL" value="0" reducedCost="0.22300000000000003"/>
<variable name="C0006" index="5" status="LL" value="0" reducedCost="0.28599999999999998"/>
<variable name="C0007" index="6" status="LL" value="0" reducedCost="0.1205"/>
<variable name="C0008" index="7" status="BS" value="1" reducedCost="0"/>
<variable name="C0009" index="8" status="BS" value="0" reducedCost="0"/>
<variable name="C0010" index="9" status="LL" value="0" reducedCost="0.17449999999999996"/>
<variable name="C0011" index="10" status="UL" value="1" reducedCost="-0.11500000000000005"/>
<variable name="C0012" index="11" status="LL" value="0" reducedCost="0.72299999999999998"/>
<variable name="C0013" index="12" status="LL" value="0" reducedCost="0.35449999999999993"/>
<variable name="C0014" index="13" status="LL" value="0" reducedCost="0.30349999999999999"/>
<variable name="C0015" index="14" status="LL" value="0" reducedCost="0.18550000000000003"/>
<variable name="C0016" index="15" status="BS" value="1" reducedCost="0"/>
<variable name="C0017" index="16" status="UL" value="1" reducedCost="-0.26999999999999996"/>
<variable name="C0018" index="17" status="LL" value="0" reducedCost="0.30750000000000005"/>
<variable name="C0019" index="18" status="LL" value="0" reducedCost="0.21100000000000008"/>
<variable name="C0020" index="19" status="BS" value="0" reducedCost="0"/>
<variable name="C0021" index="20" status="LL" value="0" reducedCost="0.38149999999999995"/>
<variable name="C0022" index="21" status="LL" value="0" reducedCost="0.021000000000000019"/>
<variable name="C0023" index="22" status="LL" value="0" reducedCost="1.046"/>
<variable name="C0024" index="23" status="LL" value="0" reducedCost="0.84399999999999997"/>
<variable name="C0025" index="24" status="UL" value="1" reducedCost="-0.19699999999999998"/>
<variable name="C0026" index="25" status="LL" value="0" reducedCost="0.078999999999999987"/>
<variable name="C0027" index="26" status="LL" value="0" reducedCost="0.42250000000000004"/>
<variable name="C0028" index="27" status="LL" value="0" reducedCost="0.04450000000000004"/>
<variable name="C0029" index="28" status="LL" value="0" reducedCost="0.22099999999999997"/>
<variable name="C0030" index="29" status="LL" value="0" reducedCost="0.3105"/>
<variable name="C0031" index="30" status="LL" value="0" reducedCost="0.46549999999999997"/>
<variable name="C0032" index="31" status="BS" value="1" reducedCost="0"/>
<variable name="C0033" index="32" status="LL" value="0" reducedCost="0.45649999999999996"/>
<variable name="C0034" index="33" status="LL" value="0" reducedCost="0.81100000000000005"/>
<variable name="C0035" index="34" status="LL" value="0" reducedCost="0.61799999999999999"/>
<variable name="C0036" index="35" status="LL" value="0" reducedCost="0.255"/>
<variable name="C0037" index="36" status="BS" value="1" reducedCost="0"/>
<variable name="C0038" index="37" status="LL" value="0" reducedCost="0.34950000000000009"/>
<variable name="C0039" index="38" status="LL" value="0" reducedCost="0.055500000000000049"/>
<variable name="C0040" index="39" status="LL" value="0" reducedCost="0.158"/>
<variable name="C0041" index="40" status="LL" value="0" reducedCost="0.23750000000000004"/>
<variable name="C0042" index="41" status="LL" value="0" reducedCost="0.38750000000000001"/>
<variable name="C0043" index="42" status="BS" value="0" reducedCost="0"/>
<variable name="C0044" index="43" status="LL" value="0" reducedCost="0.3765"/>
<variable name="C0045" index="44" status="LL" value="0" reducedCost="0.58799999999999997"/>
<variable name="C0046" index="45" status="LL" value="0" reducedCost="0.45500000000000002"/>
<variable name="C0047" index="46" status="LL" value="0" reducedCost="0.498"/>

<variable name="C0048" index="47" status="LL" value="0" reducedCost="0.24399999999999999"/>
<variable name="C0049" index="48" status="LL" value="0" reducedCost="0.18150000000000002"/>
<variable name="C0050" index="49" status="LL" value="0" reducedCost="0.042500000000000038"/>
<variable name="C0051" index="50" status="BS" value="0.5" reducedCost="0"/>
<variable name="C0052" index="51" status="LL" value="0" reducedCost="0.04550000000000004"/>
<variable name="C0053" index="52" status="LL" value="0" reducedCost="0.19850000000000007"/>
<variable name="C0054" index="53" status="BS" value="0.5" reducedCost="0"/>
<variable name="C0055" index="54" status="LL" value="0" reducedCost="0.18049999999999994"/>
<variable name="C0056" index="55" status="LL" value="0" reducedCost="0.54700000000000004"/>
<variable name="C0057" index="56" status="LL" value="0" reducedCost="0.186"/>
<variable name="C0058" index="57" status="LL" value="0" reducedCost="0.79100000000000004"/>
<variable name="C0059" index="58" status="LL" value="0" reducedCost="0.54300000000000004"/>
<variable name="C0060" index="59" status="LL" value="0" reducedCost="0.33000000000000002"/>
<variable name="C0061" index="60" status="BS" value="0.5" reducedCost="0"/>
<variable name="C0062" index="61" status="LL" value="0" reducedCost="0.26350000000000001"/>
<variable name="C0063" index="62" status="LL" value="0" reducedCost="0.15000000000000002"/>
<variable name="C0064" index="63" status="BS" value="0.5" reducedCost="0"/>
<variable name="C0065" index="64" status="LL" value="0" reducedCost="0.35750000000000015"/>
<variable name="C0066" index="65" status="BS" value="0" reducedCost="0"/>
<variable name="C0067" index="66" status="LL" value="0" reducedCost="0.96999999999999997"/>
<variable name="C0068" index="67" status="LL" value="0" reducedCost="0.27600000000000002"/>
<variable name="C0069" index="68" status="LL" value="0" reducedCost="0.77300000000000002"/>
<variable name="C0070" index="69" status="LL" value="0" reducedCost="0.60899999999999999"/>
<variable name="C0071" index="70" status="LL" value="0" reducedCost="0.55100000000000005"/>
<variable name="C0072" index="71" status="LL" value="0" reducedCost="0.42499999999999999"/>
<variable name="C0073" index="72" status="LL" value="0" reducedCost="0.21550000000000008"/>
<variable name="C0074" index="73" status="LL" value="0" reducedCost="0.16600000000000009"/>
<variable name="C0075" index="74" status="BS" value="0.5" reducedCost="0"/>
<variable name="C0076" index="75" status="LL" value="0" reducedCost="0.23350000000000007"/>
<variable name="C0077" index="76" status="LL" value="0" reducedCost="0.050000000000000044"/>
<variable name="C0078" index="77" status="LL" value="0" reducedCost="0.45200000000000001"/>
<variable name="C0079" index="78" status="LL" value="0" reducedCost="0.53400000000000003"/>
<variable name="C0080" index="79" status="LL" value="0" reducedCost="0.59699999999999998"/>
<variable name="C0081" index="80" status="LL" value="0" reducedCost="0.35899999999999999"/>
<variable name="C0082" index="81" status="LL" value="0" reducedCost="0.156"/>
<variable name="C0083" index="82" status="LL" value="0" reducedCost="0.36899999999999999"/>
<variable name="C0084" index="83" status="LL" value="0" reducedCost="0.68100000000000005"/>
<variable name="C0085" index="84" status="UL" value="1" reducedCost="-0.008499999999999952"/>
<variable name="C0086" index="85" status="LL" value="0" reducedCost="0.22250000000000003"/>
<variable name="C0087" index="86" status="BS" value="0.5" reducedCost="0"/>
<variable name="C0088" index="87" status="LL" value="0" reducedCost="0.17549999999999999"/>
<variable name="C0089" index="88" status="LL" value="0" reducedCost="0.36099999999999999"/>
<variable name="C0090" index="89" status="LL" value="0" reducedCost="0.46700000000000003"/>
<variable name="C0091" index="90" status="LL" value="0" reducedCost="0.71599999999999997"/>
<variable name="C0092" index="91" status="LL" value="0" reducedCost="0.46800000000000003"/>

```
  <variable name="C0093" index="92" status="LL" value="0" reducedCost="0.23100000000000001"/>
  <variable name="C0094" index="93" status="LL" value="0" reducedCost="0.28499999999999998"/>
  <variable name="C0095" index="94" status="LL" value="0" reducedCost="0.66100000000000003"/>
  <variable name="C0096" index="95" status="LL" value="0" reducedCost="0.13400000000000001"/>
  <variable name="C0097" index="96" status="LL" value="0" reducedCost="0.069000000000000061"/>
  <variable name="C0098" index="97" status="LL" value="0" reducedCost="0.10250000000000001"/>
  <variable name="C0099" index="98" status="LL" value="0" reducedCost="0.014000000000000012"/>
  <variable name="C0100" index="99" status="LL" value="0" reducedCost="0.44400000000000001"/>
  <variable name="C0101" index="100" status="LL" value="0" reducedCost="0.30599999999999999"/>
  <variable name="C0102" index="101" status="LL" value="0" reducedCost="0.95399999999999996"/>
  <variable name="C0103" index="102" status="LL" value="0" reducedCost="0.70099999999999996"/>
  <variable name="C0104" index="103" status="LL" value="0" reducedCost="0.46700000000000003"/>
  <variable name="C0105" index="104" status="LL" value="0" reducedCost="0.185"/>
  <variable name="C0106" index="105" status="LL" value="0" reducedCost="0.57799999999999996"/>
  <variable name="C0107" index="106" status="LL" value="0" reducedCost="0.44800000000000001"/>
  <variable name="C0108" index="107" status="LL" value="0" reducedCost="0.32400000000000001"/>
  <variable name="C0109" index="108" status="LL" value="0" reducedCost="0.33450000000000002"/>
  <variable name="C0110" index="109" status="UL" value="1" reducedCost="-0.24700000000000005"/>
  <variable name="C0111" index="110" status="LL" value="0" reducedCost="0.60499999999999998"/>
  <variable name="C0112" index="111" status="LL" value="0" reducedCost="0.70699999999999996"/>
  <variable name="C0113" index="112" status="LL" value="0" reducedCost="0.46300000000000002"/>
  <variable name="C0114" index="113" status="LL" value="0" reducedCost="0.28799999999999998"/>
  <variable name="C0115" index="114" status="LL" value="0" reducedCost="0.255"/>
  <variable name="C0116" index="115" status="LL" value="0" reducedCost="0.56200000000000006"/>
  <variable name="C0117" index="116" status="LL" value="0" reducedCost="0.79800000000000004"/>
  <variable name="C0118" index="117" status="LL" value="0" reducedCost="0.21199999999999999"/>
  <variable name="C0119" index="118" status="LL" value="0" reducedCost="0.34399999999999997"/>
  <variable name="C0120" index="119" status="LL" value="0" reducedCost="0.65900000000000003"/>
  <variable name="C0121" index="120" status="LL" value="0" reducedCost="0.28750000000000003"/>
  <variable name="C0122" index="121" status="LL" value="0" reducedCost="0.35899999999999999"/>
  <variable name="C0123" index="122" status="LL" value="0" reducedCost="0.39000000000000001"/>
  <variable name="C0124" index="123" status="LL" value="0" reducedCost="0.97499999999999998"/>
  <variable name="C0125" index="124" status="LL" value="0" reducedCost="0.71999999999999997"/>
  <variable name="C0126" index="125" status="LL" value="0" reducedCost="0.47899999999999998"/>
  <variable name="C0127" index="126" status="LL" value="0" reducedCost="0.248"/>
  <variable name="C0128" index="127" status="LL" value="0" reducedCost="0.65800000000000003"/>
  <variable name="C0129" index="128" status="LL" value="0" reducedCost="0.43099999999999999"/>
  <variable name="C0130" index="129" status="LL" value="0" reducedCost="0.29899999999999999"/>
  <variable name="C0131" index="130" status="LL" value="0" reducedCost="0.087999999999999995"/>
  <variable name="C0132" index="131" status="LL" value="0" reducedCost="0.64200000000000002"/>
 </variables>
 <objectiveValues>
  <objective index="0" name="R0015" value="3.2714999999999996"/>
 </objectiveValues>
</CPLEXSolution>
```

References

[1] Laurence A. Wolsey, George L. Nemhauser, *Integer and Combinatorial Optimization.*

[2] IBM, *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual*