Mathematical Programming in Advanced Analytics

Project 2 Deliverables

Group 2 :
B063241
B060210
B128088
B082983

UNIVERSITY OF EDINBURGH
Business School

Abstract

ABC Inc. is faced with the problem of assigning 50 of their customers in the EH3 postcode are to one vehicle, and since the service cost and time are assumed to be proportional to distance, they wish to find the shortest route which starts at the warehouse and returns to the warehouse, visiting each customer in the cluster once and only once. A graphical representation of how these customers are scattered around the area is shown below.

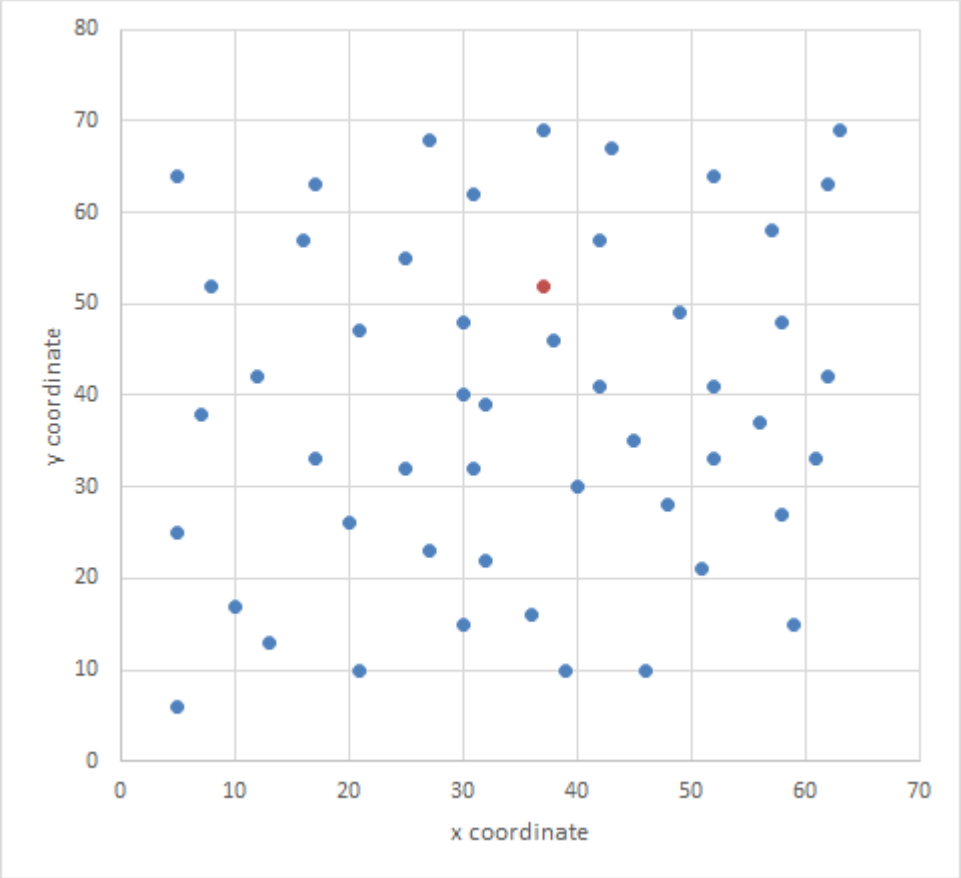

Figure 1: Scatter of the customers (blue) around the warehouse (red)

Task 1

Matrix formulation using DFJ sub-tour breaking constraints

The matrix format of the problem where S is the set of sub-tour breaking constraints proposed by Dantzig, Fulkerson and Johnson, 1954, is given by

$$\begin{aligned} \text{Minimize z} = \quad & \left[c^T_{1\times n^2}\right]\left[x_{n^2\times 1}\right] & & \\ \text{s.t.} \quad & \begin{bmatrix} A^1_{n\times n^2} \\ A^2_{n\times n^2} \\ A^3_{n_{sbt}\times n^2} \end{bmatrix} \left[\, x_{n^2\times 1}\right] & = & \begin{bmatrix} 1_{n\times 1} \\ 1_{n\times 1} \\ (|R|-1)_{n_{sbt}\times 1} \end{bmatrix} \\ & x = 0, 1 & \leq & \end{aligned} \tag{1}$$

where $c^T_{1\times n^2}$ is a vector of distances between every pair of customers, R is the set of customers in the sub-tour, and n_{sbt} is the number of sub-tour breaking constraints.

Comment on structures of the constraints matrix

A^1 denotes the matrix of coefficients of the first set of degree constraints:

$$A^1_{n\times n^2} = (\underbrace{I^1_{n\times n}\ I^2_{n\times n} \dots\ I^n_{n\times n}}_{n \text{ times}})$$

where $I^k_{n\times n}$ is an $n \times n$ identity matrix.

A^2 denotes the matrix of coefficients of the second set of degree constraints:

$$A^2_{n\times n^2} = (\underbrace{J^1_{n\times n}\ J^2_{n\times n} \dots\ J^n_{n\times n}}_{n \text{ times}})^T$$

where $J^k_{n\times n}$ is a row vector of zeros except for the entries in the range $[(k-1)n+1, kn]$, which are ones.

A^3 denotes the matrix of coefficients of the DFJ sub-tour breaking constraints:

$$A^3_{n_{sbt}\times n^2} = (\underbrace{L^1_{n\times n}\ L^2_{n\times n} \dots\ L^{n_{sbt}}_{n\times n}}_{n_{sbt} \text{ times}})^T$$

where $L^k_{n\times n}$ is a row vector of zeros except for the entries which lie in the sub-tour, which are ones.

Pseudo-code for generating constraints matrix

Algorithm 1 Generate Constraints Matrix

procedure TSP (DFJ Formulation) Constraints Matrix

Declare A^{temp} as an empty matrix
FOR $i = 1$ to n
{
Set A^{temp} = horizontal concatenation of A^{temp} and $\mathbb{I}_{n \times n}$
}
$A^1_{n \times n^2} = A^{temp}$

Declare J^{temp} as an empty matrix
Declare A^{temp} as an empty matrix
FOR $i = 1$ to n
{
Set J^{temp} to a zero matrix of size $n \times n$ except for the i^{th} row which is a vector of ones
Set A^{temp} = horizontal concatenation of A^{temp} and J^{temp}
}
$A^2_{n \times n^2} = A^{temp}$

Declare L^{temp} as an empty matrix
Declare A^{temp} as an empty matrix
FOR $i = 1$ to n_{sbt}
{
Set L^{temp} to a zero matrix of size $1 \times n^2$ except for entries which correspond to all paths which could be in the i^{th} sub-tour, which are ones
Set A^{temp} = vertical concatenation of A^{temp} and L^{temp}
}
$A^3_{n_{sbt} \times n^2} = A^{temp}$

Matrix formulation using connectivity constraints

The matrix format of the problem where S is the set of connectivity constraints is given by

$$\begin{aligned} \text{Minimize z} = \quad & \left[c^T_{1\times n^2}\right]\left[x_{n^2\times 1}\right] & & \\ \text{s.t.} \quad & \begin{bmatrix} A^1_{n\times n^2} \\ A^2_{n\times n^2} \\ A^3_{n_{sbt}\times n^2} \end{bmatrix}\left[\ x_{n^2\times 1}\right] & = & \begin{bmatrix} 1_{n\times 1} \\ 1_{n\times 1} \\ 1_{n_{sbt}\times 1} \end{bmatrix} \\ & x = 0, 1 & \geq & \end{aligned} \tag{2}$$

where $c^T_{1\times n^2}$ is a vector of distances between every pair of customers, n_{sbt} is the number of connectivity constraints.

Comment on structures of the constraints matrix

A^1 denotes the matrix of coefficients of the first set of degree constraints:

$$A^1_{n\times n^2} = (\underbrace{I^1_{n\times n}\ I^2_{n\times n} \dots\ I^n_{n\times n}}_{n \text{ times}})$$

where $I^k_{n\times n}$ represents the $k^{th} n \times n$ identity matrix.

A^2 denotes the matrix of coefficients of the second set of degree constraints:

$$A^2_{n\times n^2} = (\underbrace{J^1_{1\times n^2}\ J^2_{1\times n^2} \dots\ J^n_{1\times n^2}}_{n \text{ times}})^T$$

where $J^k_{1\times n^2}$ is a row vector of zeros except for the entries in the range $[(k-1)n+1, kn]$, which are ones.

A^3 denotes the matrix of the set of connectivity constraints:

$$A^3_{n_{sbt}\times n^2} = (\underbrace{L^1_{1\times n^2}\ L^2_{1\times n^2} \dots\ L^{n_{sbt}}_{1\times n^2}}_{n_{sbt} \text{ times}})^T$$

where $L^k_{1\times n^2}$ is a row vector of zeros except for the entries corresponding to the paths which go from a customer that is in the sub-tour to a customer that is not in the sub-tour, which are ones.

Pseudo-code for generating constraints matrix

Algorithm 2 Generate Constraints Matrix

1: **procedure** TSP (CC Formulation) Constraints Matrix
2:
3: Declare A^{temp} as an empty matrix
4: **FOR** $i = 1$ to n
5: {
6: Set A^{temp} = horizontal concatenation of A^{temp} and $\mathbb{I}_{n \times n}$
7: }
8: $A^1_{n \times n^2} = A^{temp}$
9:
10: Declare J^{temp} as an empty matrix
11: Declare A^{temp} as an empty matrix
12: **FOR** $i = 1$ to n
13: {
14: Set J^{temp} to a zero matrix of size $n \times n$ except for the i^{th} row which is a vector of ones
15: Set A^{temp} = horizontal concatenation of A^{temp} and J^{temp}
16: }
17: $A^2_{n \times n^2} = A^{temp}$
18:
19: Declare L^{temp} as an empty matrix
20: Declare A^{temp} as an empty matrix
21: **FOR** $i = 1$ to n_{sbt}
22: {
23: Set L^{temp} to a zero matrix of size $1 \times n^2$ except for entries which correspond to all paths which go from a customer that could be in the i^{th} sub-tour to a customer that isn't in the i^{th} sub-tour, which are ones
24: Set A^{temp} = vertical concatenation of A^{temp} and L^{temp}
25: }
26: $A^3_{n_{sbt} \times n^2} = A^{temp}$

Matrix formulation using MTZ sub-tour breaking constraints

The matrix format of the problem where S is the set of sub-tour breaking constraints proposed by Miller, Tucker, and Zimlin, 1960, is given by

$$\begin{aligned} \text{Minimize z} = \quad & \left[c^T_{1\times n^2}\right]\left[x_{n^2\times 1}\right] \\ \text{s.t.} \quad & \begin{bmatrix} A^1_{n\times(n^2+n)} \\ A^2_{n\times(n^2+n)} \\ A^3_{n_{sbt}\times(n^2+n)} \end{bmatrix} \begin{bmatrix} x_{n^2\times 1} \\ u_{n\times 1} \end{bmatrix} \begin{matrix} = \\ = \\ \leq \end{matrix} \begin{bmatrix} 1_{n\times 1} \\ 1_{n\times 1} \\ (\text{n - 2})_{n_{sbt}\times 1} \end{bmatrix} \\ & x = 0, 1 \end{aligned} \tag{3}$$

where $c^T_{1\times n^2}$ is a vector of distances between every pair of customers, n_{sbt} is the number of MTZ sub-tour breaking constraints.

Comment on structures of the constraints matrix

A^1 denotes the matrix of coefficients of the first set of degree constraints:

$$A^1_{n\times(n^2+n)} = (\underbrace{I^1_{n\times n}\ I^2_{n\times n} \dots\ I^n_{n\times n}}_{n \text{ times}} Z_{n\times n})$$

where $I^k_{n\times n}$ represents the $k^{th} n \times n$ identity matrix.

A^2 denotes the matrix of coefficients of the second set of degree constraints:

$$A^2_{n\times(n^2+n)} = (\underbrace{(J^1_{1\times n^2}\ J^2_{1\times n^2} \dots\ J^n_{1\times n^2})^T}_{n \text{ times}} Z_{n\times n})$$

where $J^k_{1\times n^2}$ is a row vector of zeros except for the entries in the range $[(k-1)n+1, kn]$, which are ones.

A^3 denotes the matrix of constraints:

$$A^3_{n_{sbt}\times n^2} = (\underbrace{(L^1_{1\times n^2}\ L^2_{1\times n^2} \dots\ L^{n_{sbt}}_{1\times n^2})^T}_{n_{sbt} \text{ times}} \underbrace{(U^1_{1\times n}\ U^2_{1\times n} \dots\ U^{n_{sbt}}_{1\times n})^T}_{n_{sbt} \text{ times}})$$

where $L^k_{1\times n^2}$ is a row vector of zeros, apart from the entry corresponding to x_{ij} in the decision vector, which takes the value $(n-1)$. $U^k_{1\times n}$ is a row vector of zeros except for the i^{th} entry which is one, and the j^{th} entry which is minus one. At no point can $i = j$.

Pseudo-code for generating constraints matrix

Algorithm 3 Generate Constraints Matrix

procedure TSP (MTZ Formulation) Constraints Matrix

Declare A^{temp} as an empty matrix
FOR $i = 1$ to n
{
Set A^{temp} = horizontal concatenation of A^{temp} and $\mathbb{I}_{n \times n}$
}
Set A^{temp} = horizontal concatenation of A^{temp} and a matrix of zeros with dimension $n \times n$
$A^{1}_{n \times (n^2+n)} = A^{temp}$

Declare J^{temp} as an empty matrix
Declare A^{temp} as an empty matrix
FOR $i = 1$ to n
{
Set J^{temp} to a zero matrix of size $n \times n$ except for the i^{th} row which is a vector of ones
Set A^{temp} = horizontal concatenation of A^{temp} and J^{temp}
}
Set A^{temp} = horizontal concatenation of A^{temp} and a matrix of zeros with dimension $n \times n$
$A^{2}_{n \times (n^2+n)} = A^{temp}$

Declare L^{temp} as an empty matrix
Declare U^{temp} as an empty matrix
Declare A^{temp} as an empty matrix
FOR $i = 2$ to n
{
FOR $j = 2$ to n, $j \neq i$
{
Set L^{temp1} to a zero matrix of size $1 \times n^2$ apart from the entries corresponding to element x_{ij} which take the value $(n-1)$.
Set L^{temp} = vertical concatenation of L^{temp} and L^{temp1}
Set U^{temp1} to a zero matrix of size $1 \times n$ apart from entry i which is 1 and j which is -1
Set U^{temp} = vertical concatenation of U^{temp} and U^{temp1}
}
}
Set A^{temp} = horizontal concatenation of L^{temp} and U^{temp}
$A^{3}_{n_{sbt} \times (n^2+n)} = A^{temp}$

Task 2

Performance of sub-tour selection rules

Constructing these three variations of the A^3 constraints matrix can be very computationally expensive - the number of constraints in the matrix grows exponentially with the problem size. Therefore when implementing them it is convenient to only generate a constraint when needed. This is achieved by an iterative procedure with the following steps:

- The relaxation of the assignment problem is solved
- If the solution contains sub-tours, then move onto the iterative step
- Iterative step:
  - Choose a sub-tour from the solution and generate the corresponding sub-tour breaking constraint
  - Add the constraint to the constraints matrix
  - Solve the problem again
  - If there are still sub-tours, return to the start of the iterative step and repeat, else you have the optimal solution

The choice of which constraint to break should be guided by a rule, several different rules for selecting which sub-tour to break were tested and the results compared with each other. Namely, these rules were: shortest (distance) sub-tour; longest (distance) sub-tour; smallest (fewest stops) sub-tour; and largest (most stops) sub-tour.

Solution to assignment based formulation relaxation

Shown in the Figure 2 is a graph illustrating the solution to the relaxation of the assignment based formulation. The relaxation problem is simply of the form

$$\begin{aligned} \text{Minimize z} = \quad & \left[c^T_{1\times n^2}\right]\left[x_{n^2\times 1}\right] & & \\ \text{s.t.} \quad & \begin{bmatrix} A^1_{n\times n^2} \\ A^2_{n\times n^2} \end{bmatrix} \left[\ x_{n^2\times 1}\right] & = & \begin{bmatrix} 1_{n\times 1} \\ 1_{n\times 1} \end{bmatrix} \\ & x = 0, 1 & & \end{aligned} \tag{4}$$

where A^1, A^2 have the same structure as described for the DFJ formulation. Quite clearly this solution contains many sub-tours, and therefore isn't a feasible solution to the problem as the deliveries aren't being made in one

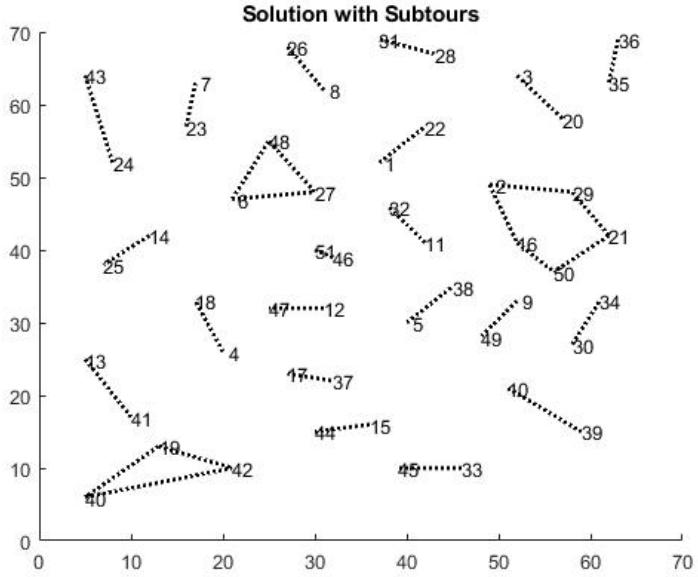

Figure 2: Solution to relaxation of assignment problem

round trip, or tour. To transform this into a feasible and in fact optimal solution, the iterative process detailed in the previous section is applied for each type of sub-tour breaking constraint.

Solutions to the different formulations

DFJ formulation

Once a sub-tour has been chosen to break by one of the rules, the DFJ constraint will remove this sub-tour by not allowing a tour to take place linking only a combination of the customers in the sub-tour. For example, suppose a sub-tour going from i to j to k has been chosen to be broken, the DFJ constraint says "the number of paths between customers i, j, k has to be less than or equal to 2", and so a sub-tour can't take place linking these three customers.

The iterative procedure to solving the DFJ formulation of the problem was carried out four times, each time using a different rule to select which sub-tour to break. It quickly became apparent that both the 'longest' and 'largest' rules were far more computationally expensive compared to the 'shortest' and 'smallest' rules, and so these procedures were stopped. When solving the problem using both the 'shortest' and 'smallest' rules, the iterative pro-

cedure ran 34 times before reaching a feasible, optimal solution. This means that 34 different sub-tour breaking constraints were concatenated onto the constraints matrix in order to reach the solution. When the algorithms were ran and timed, it seemed as though the algorithm which performed best was the one choosing which sub-tour to break using the 'shortest' sub-tour rule, with the average run times (taken from 8 runs, alternating between rule each time) of the respective algorithms being 16.145 seconds and 16.277 seconds.

'Shortest' run number	Time (seconds)	'Smallest' run number	Time (seconds)
1	15.971	1	16.252
2	16.438	2	18.422
3	17.521	3	16.314
4	16.182	4	15.755
5	16.151	5	16.230
6	15.719	6	15.685
7	15.721	7	15.790
8	15.456	8	15.771
Avg.	16.145	Avg.	16.277

Connectivity constraints formulation

Once a sub-tour has been chosen to break by one of the rules, the CC constraint will remove this sub-tour by forcing the path to go from one of the customers that is in the sub-tour to one that is not, breaking the sub-tour. For example, suppose a sub-tour going from i to j to k has been chosen to be broken, the CC constraint says "the number of paths between customers i, j, k and any customer p who is not in the current sub-tour has to be greater than or equal to 1", and so a sub-tour can't take place linking these initial three customers.

Again, the iterative procedure to solving the cc formulation of the problem was carried out four times, each time using a different rule to select which sub-tour to break. It quickly became apparent that both the 'longest' and 'largest' rules were again far more computationally expensive compared to the 'shortest' and 'smallest' rules, and so these procedures were stopped. When solving the problem using both the 'shortest' and 'smallest' rules, the iterative procedure ran 35 times before reaching a feasible, optimal solution. This means that 35 different sub-tour breaking constraints were concatenated onto the constraints matrix in order to reach the solution. When the algorithms

were ran and timed, it seemed as though the algorithm which performed best was the one choosing which sub-tour to break using the 'shortest' sub-tour rule, with the average run times (taken from 8 runs, alternating between rule each time) of the respective algorithms being 19.717 seconds and 19.804.

'Shortest' run number	Time (seconds)	'Smallest' run number	Time (seconds)
1	21.397	1	22.239
2	16.515	2	16.603
3	20.142	3	19.151
4	16.594	4	19.309
5	21.995	5	21.967
6	21.078	6	22.060
7	20.354	7	20.484
8	19.660	8	16.617
Avg.	19.717	Avg.	19.804

MTZ formulation

Where the DFJ and CC constraints break the sub-tours in fairly similar ways, the MTZ constraints use the introduction of additional decision variables, u, which are used to break the sub-tour in each iteration. The implementation of this formulation did not reach an optimal solution in MATLAB, please see the code for details.

Computational performance of the three formulations

The computational performance of the use of the DFJ and CC types of sub-tour breaking constraint in the iterative process can be measured in a number of ways, but perhaps the two measures which are easiest to interpret are the number of constraints added (iterations) required to reach optimality, and the time taken for the program to reach optimality. Using these measures, the formulation with the DFJ constraints can be seen to reach the optimal solution by using fewer constraints and in a shorter time. The optimal solution found is shown in Figure 3. The total distance of this path, that starts at the warehouse and travels through each of the customers visiting each one only once, is 428.8718 units of distance.

MATLAB CODE

Please find the MATLAB code used in Project 2 Task 2 in the appropriate folder on the memory stick.

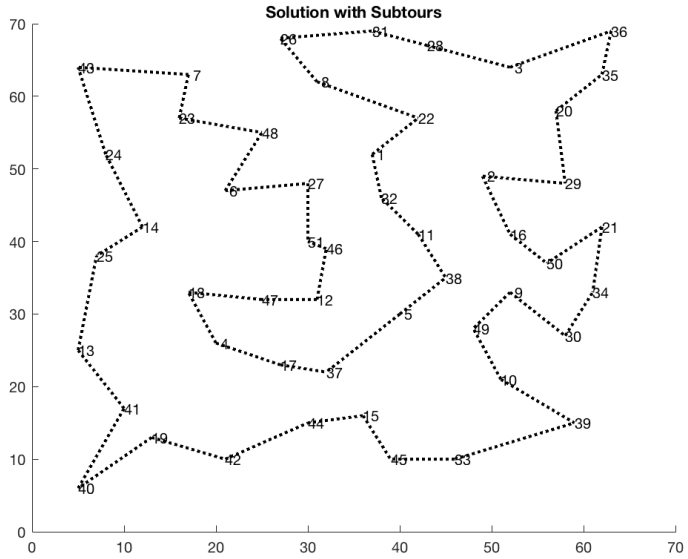

Figure 3: Optimal single tour solution

Task 3

To assess the performance of the solution procedure in figure 1 and its implementation for different types of sub-tour breaking constraints, the TSP provided was solved using the assignment-based formulation using flow variables and this solution used as a benchmark.

Matrix representation of assignment based formulation with flow variables

The matrix format of the assignment-based formulation with flow variables is given by

$$\begin{aligned} \text{Minimize z} = \quad & \left[c^T_{1\times n^2}\right]\left[x_{n^2\times 1}\right] & & \\ \text{s.t.} \quad & \begin{bmatrix} A^1_{n\times 2n^2} \\ A^2_{n\times 2n^2} \\ A^3_{n\times 2n^2} \\ A^4_{n^2\times 2n^2} \end{bmatrix} \begin{bmatrix} x_{n^2\times 1} \\ y_{n^2\times 1} \end{bmatrix} & = & \begin{bmatrix} 1_{n\times 1} \\ 1_{n\times 1} \\ -1_{n\times 1} \\ 0_{n^2\times 1} \end{bmatrix} \\ & \mathbf{x} = 0, 1, \mathbf{y} \geq 0 & \leq & \end{aligned} \tag{5}$$

where $c^T_{1\times n^2}$ is a vector of distances between every pair of customers, and $\mathbf{y}$ is a set of flow variables.

Pseudo-code for generating the constraints matrix with flow variables

Algorithm 4 Generate Constraints Matrix

procedure TSP (MTZ Formulation) Constraints Matrix

Declare A^{temp} as an empty matrix
FOR $i = 1$ to n
{
Set A^{temp} = horizontal concatenation of A^{temp} and $\mathbb{I}_{n \times n}$
}
FOR $i = 1$ to n
{
Set A^{temp} = horizontal concatenation of A^{temp} and a zero matrix of size $n \times n$
}
$A^1_{n \times 2n^2} = A^{temp}$

Declare J^{temp} as an empty matrix
Declare A^{temp} as an empty matrix
FOR $i = 1$ to n
{
Set J^{temp} to a zero matrix of size $n \times n$ except for the i^{th} row which is a vector of ones
Set A^{temp} = horizontal concatenation of A^{temp} and J^{temp}
}
FOR $i = 1$ to n
{
Set A^{temp} = horizontal concatenation of A^{temp} and a zero matrix of size $n \times n$
}
$A^2_{n \times 2n^2} = A^{temp}$

Declare Y^{temp} as an empty matrix
Declare A^{temp} as an empty matrix
FOR $i = 1$ to n
{
Set A^{temp} = horizontal concatenation of A^{temp} and a zero matrix of size $n \times n$
}
FOR $i = 2$ to n
{
FOR $j = 1$ to n
{
Declare Y^{temp1} as a zero matrix of dimension 1 $\times n^2$ apart from the entries corresponding to y_{ij} in the decision vector which take the value 1, and the entries corresponding to y_{ji} in the decision vector
}
Set Y^{temp} = vertical concatenation of Y^{temp1}
}
Set $A^3_{n \times 2n^2}$ = horizontal concatenation of A^{temp} and Y^{temp}

Set U $\geq$ n-1 (for example n)
Declare AZ an identity matrix of size $n^2 \times n^2$
A^4 = Horizontally concatenate -U*AZ and AZ

Solution to assignment based formulation with flow variables

Solution was not reached with the implementation in MATLAB, please see the code for details.

MATLAB CODE

Please find the MATLAB code used in Project 2 Task 2 in the appropriate folder on the memory stick.