Computational Methods for Industrial Engineering Term Project (TSP)

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

Our program is a popular topic in modeling, which is Travelling Salesman Problem. When we were assigned to the project our first approach was to find as many as possible viable similar programs. We have even watch lectures on the internet about Travelling Salesman Problem just to have a deeper understanding. After that, we have searched for similar code written in visual basic but we were not able to find many sources on that. Since we couldn't find a beneficial source on VBA we changed our approach to other languages. There were a lot of application of TSP in C/C++ and Python. From the information, we have gathered with our research we decided to apply an algorithm on Nearest Neighbor like this,

1. Find each node's distance to every other node.
2. Set your starting point to since it is random
3. Create a binary tableau based on whether that road is available to travel or not (true for traveled false for not traveled)
4. Find the minimum distance to the starting point. If it is an available destination then go, if not find the next minimum.
5. Update the Route Array
6. Update the Binary tableau, Set True every binary in the End Point column (It means you cannot travel to this node from anywhere)
7. If every node is travelled exit function. If not repeat from step 4

Computational Methods for Industrial Engineering Term Project (TSP)

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

```
1.
2.  Sub TSP()
3.      Dim Size As Integer
4.      Dim distance() As Single
5.      Dim traveled() As Boolean
6.      Dim Route() As Integer
7.      Dim totalDistance As Single
8.      Dim MatrixYcounter As Integer
9.      Dim StartPoint As Integer
10.     Dim EndPoint As Integer
11.     Dim minDistance As Single 'just to use in the distance loop
12.     Dim i As Integer, j As Integer, k As Integer, l As Integer
13.     Dim x1 As Integer, x2 As Integer, y1 As Integer, y2 As Integer
14.     Dim temp_dist As Single
15.     Dim coords As Range 'this is the table of coordinates
16.
17.     '2 opt variables
18.
19.     Dim Newroute() As Integer
20.     Dim Temproute1 As Integer
21.     Dim Temproute2 As Integer
22.     Dim Newroutedistance As Single
23.     Dim Iterationcounter As Integer
24.     Dim Iterationflag As Boolean
25.
26.
27.
28.
29.     ' determining how many cities are there?
30.     Size = Range(Range("a1").Offset(1, 0), Range("a1").Offset(1, 0).End(xlDown)).Rows.Count
31.     'now that we know the number of cities, redimension distance array
32.     ReDim distance(1 To Size, 1 To Size)
33.     'take the coordinates as a range
34.     Set coords = Range(Range("a2"), Range("a2").End(xlDown)).Resize(, 3)
35.
36.
37.     'control not any two nodes are the same node
38.     For i = 1 To Size
39.         For j = i + 1 To Size
40.             If Range("b1").Offset(i).Value = Range("b1").Offset(j).Value Then
41.                 If Range("c1").Offset(i).Value = Range("c1").Offset(j).Value Then
42.                     MsgBox "You have entered a node more than once"
43.                     Exit Sub
44.                 End If
45.             End If
46.         Next j
47.     Next i
48.
49.     'put in the first arm of the matrix
50.     Range("H3") = "City"
51.     Range("H3").Font.Bold = True
52.     Range("H1") = "Distance Matrix"
53.     Range("H1").Font.Bold = True
54.     With Range("H3")
55.     For i = 1 To Size
56.         .Offset(i, 0) = i
57.         .Offset(i, 0).Font.Bold = True
```

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

```
    Next
    'second arm of the matrix
    For j = 1 To Size
        .Offset(0, j) = j
        .Offset(0, j).Font.Bold = True
    Next
    'fill it in with distances
    For i = 1 To Size
        For j = 1 To Size
            'the default value is 0
            If i = j Then
                Range("H3").Offset(i, j) = 0
            'otherwise look for euclidean distance
            Else
            'search for the coordinates for each value
                x1 = WorksheetFunction.VLookup(i, coords, 2, False) 'x of i
                y1 = WorksheetFunction.VLookup(i, coords, 3, False) 'y of i
                x2 = WorksheetFunction.VLookup(j, coords, 2, False) 'x of j
                y2 = WorksheetFunction.VLookup(j, coords, 3, False) 'y of j
                temp_dist = Sqr(((x1 - x2) ^ 2) + ((y1 - y2) ^ 2))
                distance(i, j) = temp_dist              'reading the distance
                Range("H3").Offset(i, j) = temp_dist
            End If
        Next
    Next
    End With

    'Array where route will be stored. Starts and ends in City 1
    ReDim Route(1 To Size + 1)
    Route(1) = 1
    Route(Size + 1) = Route(1)

     'Boolean array indicating whether each city was already visited or not. Initialize all cities (except City 1) to
    False
    ReDim traveled(1 To Size)
    traveled(1) = True
    For i = 2 To Size
        traveled(i) = False
    Next

    'Total distance traveled is initially 0. Initial current city is City 1
    totalDistance = 0
    StartPoint = 1



    For MatrixYcounter = 2 To Size
        'initialize maxDistance to 0
        minDistance = 9999999
        For i = 1 To Size
            If i <> StartPoint And Not traveled(i) Then
                If distance(StartPoint, i) < minDistance Then
                    EndPoint = i
                    minDistance = Range("H3").Offset(StartPoint, i)
                End If
            End If
        Next i
        'store the next city to be visited in the route array
        Route(MatrixYcounter) = EndPoint
        traveled(EndPoint) = True
        'update total distance travelled
```

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

```
119.        totalDistance = totalDistance + minDistance
120.        'update current city
121.        StartPoint = EndPoint
122.    Next MatrixYcounter
123.
124.    'Update total distance traveled with the distance between the last city visited and the initial city, City 1.
125.    totalDistance = totalDistance + distance(StartPoint, 1)
126.
127.    'Print Results
128.    With Range("A2").Offset(Size + 5, 0)
129.        .Offset(0, 0).Value = "Nearest neighbor route"
130.        .Offset(1, 0).Value = "Stop #"
131.        .Offset(1, 1).Value = "City"
132.
133.        For MatrixYcounter = 1 To Size + 1
134.            .Offset(MatrixYcounter + 1, 0).Value = MatrixYcounter
135.            .Offset(MatrixYcounter + 1, 1).Value = Route(MatrixYcounter)
136.        Next MatrixYcounter
137.
138.        .Offset(Size + 4, 0).Value = "Total distance is " & totalDistance
139.
140.        For i = 1 To Size + 1
141.
142.        .Offset(Size + 5 + i, 0).Value = Application.WorksheetFunction.VLookup(Route(i), coords, 2, True)
143.        .Offset(Size + 5 + i, 1).Value = Application.WorksheetFunction.VLookup(Route(i), coords, 3, True)
144.
145.        Next i
146.
147.        .Offset(Size + 6).Select
148.        Range(Selection, Selection.End(xlToRight)).Select
149.        Range(Selection, Selection.End(xlDown)).Select
150.        ActiveSheet.Shapes.AddChart2(240, xlXYScatterLines).Select
151.
152.        ActiveChart.ChartTitle.Text = "Initial basic feasible route by Nearest Neighbour"
153.
154.    End With
```

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

After we found a basic initial feasible solution. We improved our result with the 2-opt algorithm. ‘opt algorithm was more challenging than we were expecting. Partly by we couldn’t remember the structure of the algorithm that we saw from the lecture but it was mostly about the lack of source on the internet. After several failed or semi-success attempts, we decided to meet and break the algorithm part by part to internalize what’s going on how can we convert it to a code section. We struggled until we realized the in between the focus points should be reversed. From that point on we could move towards a successful code bit by bit by breaking what we are doing, what is ought to be and what is happening by debugging. But in the end, we were successful and the algorithm goes like this:

1. Create a comparison array which will be the indicator of the difference
2. Construct a for loop with complexity n^2
3. Set New route
4. Find new routes distance
5. Compare it to the old route if smaller set old route with new route if not go to step 3

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

```
1.      ReDim Newroute(1 To Size + 1)
2.
3.      Iterationcounter = 0
4.
5.      Iterationflag = False
6.
7.
8.      For i = 1 To Size
9.          Newroute(i) = Route(i)
10.     Next i
11.
12. repatfromthestart:
13. Iterationflag = False
14.
15.     For i = 1 To Size
16.         For j = i + 1 To Size
17.             Newroute(i) = Route(j)
18.             Newroute(j) = Route(i)
19.             Newroute(Size + 1) = Newroute(1)
20.             For k = 1 To (j - i)
21.                 Newroute(i + k) = Route(j - k)
22.             Next k
23.
24.             For k = 1 To Size
25.                 Newroutedistance = Newroutedistance + distance(Newroute(k), Newroute(k + 1))
26.             Next k
27.
28.
29.
30.             If Newroutedistance < totalDistance Then
31.
32.
33.                 For k = 1 To Size + 1
34.                     Route(k) = Newroute(k)
35.                 Next k
36.
37.
38.
39.                 totalDistance = Newroutedistance
40.                 Iterationcounter = Iterationcounter + 1
41.                 Iterationflag = True
42.                 With Range("A2").Offset(Size + 5, 3 * Iterationcounter)
43.                     .Offset(0, 0).Value = "Nearest neighbor route"
44.                     .Offset(1, 0).Value = "Stop #"
45.                     .Offset(1, 1).Value = "City"
46.
47.                     For MatrixYcounter = 1 To Size + 1
48.                         .Offset(MatrixYcounter + 1, 0).Value = MatrixYcounter
49.                         .Offset(MatrixYcounter + 1, 1).Value = Route(MatrixYcounter)
50.                     Next MatrixYcounter
51.
52.
53.                 .Offset(Size + 4, 0).Value = "Total distance is " & totalDistance
```

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

```
54.                 .Offset(Size + 5, 0).Value = "Iterationcounter = " & Iterationcounter
55.
56.                 For I = 1 To Size + 1
57.
58.                     .Offset(Size + 5 + I, 0).Value = Application.WorksheetFunction.VLookup(Route(I), coords, 2, True)
59.                     .Offset(Size + 5 + I, 1).Value = Application.WorksheetFunction.VLookup(Route(I), coords, 3, True)
60.
61.                 Next I
62.
63.                 .Offset(Size + 6).Select
64.                 Range(Selection, Selection.End(xlToRight)).Select
65.                 Range(Selection, Selection.End(xlDown)).Select
66.                 ActiveSheet.Shapes.AddChart2(240, xlXYScatterLines).Select
67.
68.                 ActiveChart.ChartTitle.Text = "Iteration" & Iterationcounter
69.                 End With
70.
71.
72.             End If
73.
74.             Newroutedistance = 0
75.             If Iterationflag Then GoTo repatfromthestart
76.         Next j
77.
78.         For k = 1 To Size + 1
79.             Newroute(k) = Route(k)
80.         Next k
81.     Next i
```

041503044 – Doğa Poyraz TAHAN
041503043 – Süeda ÖZDEMİR
041503023 – Mehmet Berkay ÖZSÖZ

Upon the second heuristic, even though we struggled a lot we were not able to figure if the new route creates sub-circles out. But we create a code that can separate if a node has more than one entering and exiting. For that, first, we create an array with 2 helper array, which helper arrays indicate from which node to which note the edge is, and hold their distances. We extract from the array loopy ones and the reciprocal ones and obtained a better manageable smaller array. To prevent more than two outgoing and incoming we look at the helper function if it is used more than twice or not. Because we do not have the complete code we are not adding to this paper but it will be in our Excel file since we believe it was near complete.

For the Bonus part, we used a lot of record macro. We tried to construct a model and use it with the solver. Although we could construct a model with 8 nodes or less, we couldn't understand how the solver works with VBA. So, we stuck at the optimization part. But If optimize button is clicked we create a sheet that can be easily optimized with just opening the solver and pressing solve.

The user interface was the easiest and the most straightforward part since it doesn't require much skill from the developer.

Conclusion

Our project was mostly hard earned. But we enjoyed the experience since it pushed us to our limits and research beyond our capabilities. Our discussion was mostly about how the VBA is so huge and there are lots of ways to do the something. One can easily forget or not remember a way to do a thing much more easily. So, knowing the steps what you want to build and researching during coding is a vital part of the experience.