USING MATRICES AND HUNGARIAN METHOD TO SOLVE THE TRAVELING SALESMAN PROBLEM

Honors Thesis

Presented in Partial Fulfillment of the Requirements
For the Degree of Bachelor of Science in Mathematics

In the College of Arts and Sciences
at Salem State University

By

Briana Couto

Dr. Kathi Crowe
Faculty Advisor
Department of Mathematics

***

Commonwealth Honors Program
Salem State University
2018

Using Matrices and Hungarian Method to Solve the Traveling Salesman Problem

Briana Couto

December 12, 2017

Abstract

In this paper, we introduce the Traveling Salesman Problem (TSP) and solve for the most efficient route of the problem using the steps of the Hungarian method. Specifically, this paper discusses the properties of a TSP matrix, provides the steps for the Hungarian method, and presents examples that apply these concepts to a Traveling Salesman Problem. We do not consider any constraints on the order in which the localities are visited, nor do we take into account possible traffic at differing times. We use examples to show how the Hungarian method is used and why it is an efficient way to solve the Traveling Salesman Problem.

1 Introduction

The Traveling Salesman Problem has been some what of an anomaly for mathematicians since the early 1900s. The Traveling Salesman Problem (TSP) considers multiple localities to be visited and solves for the shortest route among these stops. We will consider the constraints in which no locality is visited twice before all other localities are visited, and that the salesman returns to the city in which they started.

The idea of mathematically solving for the most efficient route between multiple stops appears to have first been developed in the 1930s by Karl Menger, who stated that the problem would always be solvable with a finite number of trials. He believed that there were possible rules that would simplify the problem, allowing for less trials, but that he did not know what they would entail. Some progress was made in 1954 with linear-programming methods to solving a 49-city TSP, which is presented in the "granddaddy" of TSP papers, *Solution of a large scale traveling-salesman problem.* [**?**] Over time, more and more algorithms were developed, each becoming more efficient than the last, but admittedly still not as efficient as desired [**?**]. In this paper, we discuss the efficiency of a linear approach to solving the Traveling Salesman problem using matrices, specifically with the use of the Hungarian method.

The ideas presented in this article are based most predominantly on the research in *Classes of Matrices for the Traveling Salesman Problem* by Richard H. Warren. In Section 2, we review properties of matrices and, more specifically, how these properties correlate to the work we do with a TSP matrix. In Section 3 we explain and lay out the steps of the

Hungarian method and provide a simple example solving a TSP with this method. The information and example used in Section 3 are essential to understanding the more complex application in Section 4.

2 Properties of a TSP Matrix

The properties of a TSP matrix are similar to that of any matrix, but are essential to understanding the material in Section 3 and Section 4.

Recall from Section 1 that a TSP matrix A is $n \times n$, meaning there are an equal number of rows and columns. Logically, since we have a certain amount of cities that are acknowledged as both "from", our rows, and "to" our columns, this concept is natural.

The *entries* of a matrix are listed in rows and columns. Each entry can be identified by the row and column they are in. In the case of our TSP matrix A, any entry in A can be identified with the notation a_{ij}, with i being the row of the entry and j being the column of the entry. For a TSP matrix, this means that entry a_{ij} is the distance from city i to city j. Thus, the template matrix for a TSP matrix A is the same as any simple matrix, but has a more distinct, applicable meaning.

$$\begin{bmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & ... & a_{nn} \end{bmatrix}$$

In [**?**] we learn that a TSP on n cities is characterized by an $n \times n$ matrix $A = [a_{ij}]$, where a_{ij} is a real number and represents the distance from city i to city j. Since the distance from city i to city i is 0 (because there is no real distance from a location to itself) the main diagonal elements of A are always 0, and therefore are insignificant to the TSP. For the TSP A, the length of a tour t is the sum of

$$a_{1t(1)} + a_{2t(2)} + ... + a_{nt(n)}. \tag{1}$$

We can find the *optimal tour* for the TSP A such that the sum of (1) is the minimum tour t of all possible tours. As we are trying to solve the TSP for the shortest route among multiple stops, our goal is to solve for the optimal tour. As stated in [**?**] the TSP A has the same set of optimal tours as its triangular block form. Thus, the study of the TSP can actually be simplified to the study of triangular block form.

3 Hungarian Method

The Hungarian Method was developed by Kuhn and is a method of simplifying the rows and columns of a matrix A to reach optimal assignment. This essentially means that we can simplify the entries in our matrix so that we can more easily figure out what our shortest route will be for our TSP. Although the optimally assigned matrix B is not exactly equal

to our original TSP matrix A, this inequality is insignificant. What matters is that the zero entries in B correlate to the positions of the entries in A, which provides us with the shortest route, and therefore solves our TSP in a much more simple process. [**?**]

In [**?**] we learn the steps of the Hungarian method and how to transform our matrix using these steps.

Step 1: Find the smallest entry in each row and subtract it from every entry in its row.

Step 2: Find the smallest entry in each column and subtract it from every entry in its column.

Step 3: Draw lines through the columns and rows of the matrix, connecting all the 0's using as few lines as possible.

Step 4: Testing for optimality, the number of lines drawn should equal n. the number of rows in the matrix, in which case we are finished. If the numbers of lines drawn is less than n, we move to step 5.

Step 5: Find the smallest entry not covered by one of our lines. Subtract this number from each uncovered row, and add it to each covered column. From there, we go back to Step 3 and follow the rest of the steps as necessary.

Note that the operations in Step 1 and Step 2 minimize the value of the entries in correlation to the lowest entry in the row/column.

Now we will work through a simple example to apply the Hungarian method to a TSP.

Example 3.1

A salesman needs to stop through Peabody, Salem, and Boston in one route. He wants to find the most efficient route among these stops. Below is a chart of the distances between each of the stops.

From/To	Boston	Peabody	Salem
Boston	0	30	40
Peabody	30	0	10
Salem	40	10	0

Using the information described in the table, we can create a TSP matrix A. Recall from Section 1 that the main diagonal of the TSP matrix is insignificant to the problem. We will therefore disregard the entries of the main diagonal with "-" in the appropriate entries. This gives the TSP matrix for this problem.

$$\begin{bmatrix} - & 30 & 40 \\ 30 & - & 10 \\ 40 & 10 & - \end{bmatrix}$$

Now we will use the Hungarian method to simplify our matrix.

Step 1:
The smallest entries are 30 in row 1, 10 in row 2, and 10 in row 3. We subtract each number from all of the entries in their individual rows.

$$\begin{bmatrix} - & 0 & 10 \\ 20 & - & 0 \\ 30 & 0 & - \end{bmatrix}$$

Step 2:
The smallest entries are 20 in column 1, 0 in column 2, and 0 in column 3. We subtract each number from all of the entries in their individual columns.

$$\begin{bmatrix} - & 0 & 10 \\ 0 & - & 0 \\ 10 & 0 & - \end{bmatrix}$$

Step 3:
We draw vertical and horizontal lines through matrix A to connect all of the 0's in our matrix with as few lines as possible.

$$\begin{bmatrix} - & 0 & 10 \\ 0 & - & 0 \\ 10 & 0 & - \end{bmatrix}$$

Step 4:
We find that we have less lines than columns in our matrix ($2 < 3$) so we must move on to Step 5.

Step 5:
The smallest uncovered entry in the matrix is 10. We subtract 10 from each of the entries in the uncovered rows,

$$\begin{bmatrix} - & -10 & 0 \\ 0 & - & 0 \\ 0 & -10 & - \end{bmatrix}$$

and add 10 to each of the entries in the covered columns,

$$\begin{bmatrix} - & 0 & 0 \\ 0 & - & 0 \\ 0 & 0 & - \end{bmatrix}$$

and move back to Step 3.

Step 3:
We draw vertical and horizontal lines through matrix A to connect all of the 0's in our matrix with as few lines as possible.

$$\begin{bmatrix} - & 0 & 0 \\ 0 & - & 0 \\ 0 & 0 & - \end{bmatrix}$$

Step 4:
We find that we have an equal amount of lines and columns, thus we have completed the Hungarian method.

These steps simplify our matrix to easily comparable numbers and routes, but there is more to be done to solve the TSP for this example. We have simplified our matrix down, but we now need to figure out which route is the quickest. In this case, since our matrix A has all 0's for its entries, all routes are equal. For conclusion's sake, we will pick a route to be our shortest, most efficient route.

If we recall, the TSP matrix A is set up so that each entry is the distance from one city to the other. For example, the entry in a_{21} is the distance from Peabody to Boston. When solving the TSP matrix for the quickest route, we start in the first column and pick the smallest entry. In this case, both entries are 0, so either entry is efficient. We choose entry a_{21}, the distance from Peabody to Boston. This means we have chosen to start in Peabody and travel first to Boston.

$$\begin{bmatrix} - & 0 & 0 \\ \boxed{0} & - & 0 \\ \not{0} & 0 & - \end{bmatrix}$$

Now we move to row 1, the "from Boston" row. Recall that we want to visit as many localities as possible without having to touch any of the twice. Thus, since we started in Peabody, and the only other location left is Salem, we will travel from Boston to Salem.

$$\begin{bmatrix} - & \cancel{0} & \boxed{0} \\ \boxed{0} & - & 0 \\ \cancel{0} & 0 & - \end{bmatrix}$$

We have now visited all of the localities, so we must return back to our starting city, Peabody.

$$\begin{bmatrix} - & \cancel{0} & \boxed{0} \\ \boxed{0} & - & 0 \\ \cancel{0} & \boxed{0} & - \end{bmatrix}$$

Comparing our chosen, boxed routes from Peabody to Boston to Salem and back to Peabody, to our original TSP matrix A, we can find the complete, shortest route among our stops.

$$\begin{bmatrix} - & 30 & \boxed{40} \\ \boxed{30} & - & 10 \\ 40 & \boxed{10} & - \end{bmatrix}$$

Thus, our shortest route is 80 minutes, from Peabody to Boston to Salem and back to Peabody.

4 Complex Examples

Example 4.1

We are given the following table of distances in time between multiple stops.

From/To	Beverly	Salem	Weymouth	Braintree	Smithfield	Danvers	Woburn	Springfield
Beverly	0	13	45	50	95	15	32	140
Salem	15	0	40	44	85	22	43	122
Weymouth	42	35	0	7	42	55	40	145
Braintree	48	39	10	0	47	58	45	155
Smithfield	105	90	35	42	0	98	100	202
Danvers	12	24	52	55	102	0	35	148
Woburn	35	42	32	42	102	33	0	112
Springfield	135	125	142	150	212	150	110	0

Using the information described in the table, we can create a TSP matrix A. Again, recall from Section 1 that the main diagonal of the TSP matrix is insignificant to the problem. This gives the TSP matrix for this problem.

$$\begin{bmatrix} - & 13 & 45 & 50 & 95 & 15 & 32 & 140 \\ 15 & - & 40 & 44 & 85 & 22 & 43 & 122 \\ 42 & 35 & - & 7 & 42 & 55 & 40 & 145 \\ 48 & 39 & 10 & - & 47 & 58 & 45 & 155 \\ 105 & 90 & 35 & 42 & - & 98 & 100 & 202 \\ 12 & 24 & 52 & 55 & 102 & - & 35 & 148 \\ 35 & 42 & 32 & 42 & 102 & 33 & - & 112 \\ 135 & 125 & 142 & 150 & 212 & 150 & 110 & - \end{bmatrix}$$

We will use the Hungarian method from Section 3 to simplify this TSP matrix A.

Step 1: Subtract the smallest entry in each row from all of the entries in its row.

$$\begin{bmatrix} - & 0 & 32 & 37 & 82 & 2 & 19 & 127 \\ 0 & - & 25 & 29 & 70 & 7 & 28 & 107 \\ 35 & 28 & - & 0 & 35 & 48 & 33 & 138 \\ 38 & 29 & 0 & - & 37 & 48 & 35 & 145 \\ 70 & 55 & 0 & 7 & - & 63 & 65 & 167 \\ 0 & 12 & 40 & 43 & 90 & - & 23 & 136 \\ 3 & 10 & 0 & 10 & 70 & 1 & - & 80 \\ 25 & 15 & 32 & 40 & 102 & 40 & 0 & - \end{bmatrix}$$

Step 2: Subtract the smallest entry in each column from all of the entries in its column.

$$\begin{bmatrix} - & 0 & 32 & 37 & 47 & 0 & 19 & 47 \\ 0 & - & 25 & 29 & 35 & 5 & 28 & 27 \\ 35 & 28 & - & 0 & 0 & 46 & 33 & 58 \\ 38 & 29 & 0 & - & 2 & 46 & 35 & 95 \\ 70 & 55 & 0 & 7 & - & 61 & 65 & 87 \\ 0 & 12 & 40 & 43 & 55 & - & 23 & 56 \\ 3 & 10 & 0 & 10 & 35 & -1 & - & 0 \\ 25 & 15 & 32 & 40 & 67 & 38 & 0 & - \end{bmatrix}$$

Step 3: Draw vertical and horizontal lines through matrix A to connect all of the 0's in our matrix with as few lines as possible.

$$\begin{bmatrix} - & 0 & 32 & 37 & 47 & 1 & 19 & 47 \\ 0 & - & 25 & 29 & 35 & 6 & 28 & 27 \\ 35 & 28 & - & 0 & 0 & 47 & 33 & 58 \\ 38 & 29 & 0 & - & 2 & 47 & 35 & 95 \\ 70 & 55 & 0 & 7 & - & 62 & 65 & 87 \\ 0 & 12 & 40 & 43 & 55 & - & 23 & 56 \\ 3 & 10 & 0 & 10 & 35 & 0 & - & 0 \\ 25 & 15 & 32 & 40 & 67 & 39 & 0 & - \end{bmatrix}$$

Step 4: We have less lines than columns ($6 < 8$) so we must move to Step 5.

Step 5: The smallest uncovered entry is 2, so we will subtract it from the uncovered rows, and add it to all the covered columns.

$$\begin{bmatrix} - & 0 & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 25 & 27 & 33 & 4 & 28 & 25 \\ 37 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 38 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 70 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 10 & 40 & 41 & 53 & - & 23 & 54 \\ 5 & 10 & 2 & 10 & 35 & 0 & - & 0 \\ 25 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

We then return to Step 3.

Step 3: Draw vertical and horizontal lines through matrix A to connect all of the 0's in our matrix with as few lines as possible.

$$\begin{bmatrix} - & 0 & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 25 & 27 & 33 & 4 & 28 & 25 \\ 37 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 38 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 70 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 10 & 40 & 41 & 53 & - & 23 & 54 \\ 5 & 10 & 2 & 10 & 35 & 0 & - & 0 \\ 25 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

Step 4: Again, we have less lines than columns ($6 < 8$) so we must move to Step 5.

Step 5: The smallest uncovered entry is 4, so we will subtract it from the uncovered rows, and add it to all the covered columns.

$$\begin{bmatrix} - & 0 & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 21 & 23 & 29 & 0 & 24 & 21 \\ 41 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 10 & 36 & 37 & 49 & - & 19 & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & 0 \\ 29 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

Once more, we return to Step 3.

Step 3: Draw vertical and horizontal lines through matrix A to connect all of the 0's in our matrix with as few lines as possible.

$$\begin{bmatrix} - & 0 & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 21 & 23 & 29 & 0 & 24 & 21 \\ 41 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 6 & 36 & 37 & 49 & - & 19 & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & 0 \\ 29 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

Step 4: We finally have 8 lines (equal to our 8 columns) thus we have the Hungarian form of A.

Now that we have simplified our TSP matrix A, we want to solve for the shortest route. We want to start with the smallest entry in the matrix. In this case, since there are many 0's to choose from, we will select the 0 in entry a_{12}. Note that when we pick a route, we

automatically disregard its opposite entry. (i.e. Entry a_{12} was chosen, so entry a_{21} has been crossed out. This is because we do not want to go back to the city we came from.)

$$\begin{bmatrix} - & \boxed{0} & 34 & 37 & 47 & 1 & 21 & 47 \\ \cancel{0} & - & 21 & 23 & 29 & 0 & 24 & 21 \\ 41 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 6 & 36 & 37 & 49 & - & 19 & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & 0 \\ 29 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

In selecting this entry, we have decided to start from Beverly and travel first to Salem. Now we must find the shortest route from Salem. This is entry a_{26}, 0, to Danvers.

$$\begin{bmatrix} - & \boxed{0} & \cancel{34} & \cancel{37} & \cancel{47} & \cancel{1} & \cancel{21} & \cancel{47} \\ \cancel{0} & - & 21 & 23 & 29 & \boxed{0} & 24 & 21 \\ 41 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & \cancel{6} & 36 & 37 & 49 & - & 19 & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & 0 \\ 29 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

Now we must find the shortest route from Danvers. Note that we have already visited Beverly and Salem, so we will dsiregard those routes. This entry is a_{67}, 19, to Woburn.

$$\begin{bmatrix} - & \boxed{0} & \cancel{34} & \cancel{37} & \cancel{47} & \cancel{1} & \cancel{21} & \cancel{47} \\ \cancel{0} & - & \cancel{21} & \cancel{23} & \cancel{29} & \boxed{0} & \cancel{24} & \cancel{21} \\ 41 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ \cancel{0} & \cancel{6} & 36 & 37 & 49 & - & \boxed{19} & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & 0 \\ 29 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

Now we must find the shortest route from Woburn. Again, we disregard the routes to the cities we have already come from. This entry is a_{78}, 0, to Springfield.

$$\begin{bmatrix} - & \boxed{0} & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 21 & 23 & 29 & \boxed{0} & 24 & 21 \\ 41 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 6 & 36 & 37 & 49 & - & \boxed{19} & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & \boxed{0} \\ 29 & 13 & 32 & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

The shortest route from Springfield is to Weymouth. This is entry a_{83}, 32.

$$\begin{bmatrix} - & \boxed{0} & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 21 & 23 & 29 & \boxed{0} & 24 & 21 \\ 41 & 28 & - & 0 & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 6 & 36 & 37 & 49 & - & \boxed{19} & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & \boxed{0} \\ 29 & 13 & \boxed{32} & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

From Weymouth, the closest city is Braintree, entry a_{34}, 0.

$$\begin{bmatrix} - & \boxed{0} & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 21 & 23 & 29 & \boxed{0} & 24 & 21 \\ 41 & 28 & - & \boxed{0} & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & 0 & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 6 & 36 & 37 & 49 & - & \boxed{19} & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & \boxed{0} \\ 29 & 13 & \boxed{32} & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

After getting to Braintree, we only have one stop left- Smithfield. This is entry a_{45}, 0.

$$\begin{bmatrix} - & \boxed{0} & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 21 & 23 & 29 & \boxed{0} & 24 & 21 \\ 41 & 28 & - & \boxed{0} & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & \boxed{0} & 45 & 35 & 93 \\ 74 & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 6 & 36 & 37 & 49 & - & \boxed{19} & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & \boxed{0} \\ 29 & 13 & \boxed{32} & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

Since we have now visited all 8 cities, we want to return to where we started, Beverly.

$$\begin{bmatrix} - & \boxed{0} & 34 & 37 & 47 & 1 & 21 & 47 \\ 0 & - & 21 & 23 & 29 & \boxed{0} & 24 & 21 \\ 41 & 28 & - & \boxed{0} & 0 & 47 & 35 & 58 \\ 42 & 27 & 0 & - & \boxed{0} & 45 & 35 & 93 \\ \boxed{74} & 53 & 0 & 5 & - & 60 & 65 & 85 \\ 0 & 6 & 36 & 37 & 49 & - & \boxed{19} & 48 \\ 9 & 10 & 2 & 10 & 35 & 0 & - & \boxed{0} \\ 29 & 13 & \boxed{32} & 38 & 65 & 37 & 0 & - \end{bmatrix}$$

Now that we have established our most efficient route, we will determine the actual time of our route by finding these entries in our original TSP matrix A.

$$\begin{bmatrix} - & \boxed{13} & 45 & 50 & 95 & 15 & 32 & 140 \\ 15 & - & 40 & 44 & 85 & \boxed{22} & 43 & 122 \\ 42 & 35 & - & \boxed{7} & 42 & 55 & 40 & 145 \\ 48 & 39 & 10 & - & \boxed{47} & 58 & 45 & 155 \\ \boxed{105} & 90 & 35 & 42 & - & 98 & 100 & 202 \\ 12 & 24 & 52 & 55 & 102 & - & \boxed{35} & 148 \\ 35 & 42 & 32 & 42 & 102 & 33 & - & \boxed{112} \\ 135 & 125 & \boxed{142} & 150 & 212 & 150 & 110 & - \end{bmatrix}$$

Thus, this route will take 483 minutes, or 8 hours and 3 minutes.

CONCLUSION

There are many different algorithms that could be equally as efficient as this linear approach to solving the Traveling Salesman Problem, and there are many factors that could

change the efficiency of this method. We did not consider the factor of traffic throughout different times of the day, which would drastically change the results of a TSP as a whole. We have also considered that there is a possible, more efficient algorithm or method to solving the Traveling Salesman Problem. Much like the mathematicians before us, we recognize that there may be a better, undiscovered algorithm to solving a given TSP. Research could be done to compare the efficiency of this and other methods to solving the TSP to find the most efficient algorithm. However, this approach to using the Hungarian method is straight forward and applies linear algebra concepts to a real life problem.

References

[1] The math forum.

[2] Tsp history.

[3] Robert Edward Beck and Bernard Kolman. *Elementary Linear Programming with Applications*. Elsevier Inc., 2nd edition, 1995.

[4] R. Fulkerson G. Dantzig and S. Johnson. Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410, 1954.

[5] Richard H. Warren. Classes of matrices for the traveling salesman problem. *Linear Algebra and its Applications*, 139:53–62, 1990.