# School Bus Route Optimization: A Case Study

Operations Management Course Project Requested by Sepehr Marefat Institution
Shahid Beheshti University
Faculty of Management and Accounting
Mohamed Shabanpour
December 2016

Content

## Executive summary

In the North of Tehran, the capital of Iran, school buses always have trouble finding the optimal route to pick up students and deliver them to schools on time, for the population density of the area is extremely high, and it contains many routing options. This project aims to choose an optimized route for school buses of a benchmarking company that contracts to serve a high school in Northern Tehran. It is formulated as a multi-objective optimization problem developed by mixed-integer linear programming (MILP) using PuLP library in Python. The objectives considered include minimizing the total number of buses required, the entire students' walking time to bus stops so as to find the optimal stations, and the total distance travelled. Various constraints impose restrictions on the objectives, including bus capacity and school time window. Computational experiment is reported using data from the high school and benchmarking company. The suggested model is confirmed to be effective with a saving of 12.78% in the distance travelled, 19.26% in cumulative travel time of buses, and 20.06% in total walking time.

## Introduction

School bus routing problem (SBRP) categorizes as a vehicle routing problem (VRP), which is an important and active topic in computer science and operations research. In the literature, the objective of VRP is to find a minimum-cost collection of routes in a network that achieve a particular objective subject to a set of constraints. VRP is the generalized problem of traveling salesman (TSM). This problem considers a salesperson who starts from the city of his residence and intends to discover the shortest route that takes him through a given set of customers' cities exactly once and eventually returns him to his start point (Park & Kim, 2010).

In mathematical terms, TSM can be described by a complete graph $G = (N, A)$, where $N$ denotes nodes representing cities and $A$ means a set of arcs stands for the routes between cities. Each arc $(i, j)\ \epsilon\ A$ is assigned a value equal to $d_{ij}$, which is the distance between city $i$ and $j$, where $i, j\ \epsilon\ N$. The salesman can pass the nodes asymmetrically or symmetrically; in an asymmetric circular path, the distance between a pair of nodes such as $i$ and $j$ depends on the direction of motion. That is, for the two cities $i$ and $j$, there is an arc $(i, j)$ in which $d_{ij} \neq d_{ji}$. On the other hand, in symmetric TSM, for all arcs in set $A$, $d_{ij} = d_{ji}$, which indicates that direction is not determinative.

TSM aims to find the shortest Hamiltonian circuit, which is a closed path that passes through all n nodes of the graph exactly once. Therefore, an optimal solution for TSM problem is an ordered set of the nodes, such as $\pi = \{1,2, \dots, n\}$ so that the length $f(\pi)$ is minimized. The $f(\pi)$ function is calculated as follows (Sadaghiani, 2010).

$$f(\pi) = \sum_{i=1}^{n-1} d_{\pi(i)\pi(i+1)} + d_{\pi(n)\pi(1)}$$

This project aims to plan an efficient schedule for a fleet of school buses (salesman) where each bus picks up pupils (customers) from various bus stops (cities) and delivers them to the schools while satisfying multiple constraints. In this study, a benchmarking company contracts with a high school in Tehran to transport its students using school buses.

The daily procedure of buses is as follows: In the morning, school buses leave their parking places from the benchmarking company, where the routes start. It should be noted that the company is adjacent to the school, so the distance between them is negligible, and it is assumed that buses start their travels from the school. Subsequently, they pick up the students at separate bus stops and deliver them to the school. In the afternoon, the procedure is reversed.

Based on decision makers' requests, three objective functions are taken into account to discover the optimal solution. To address operational costs, two objective functions were considered: minimizing the number of buses and the total traveling distance by each bus. Moreover, an objective function was estimated to maximizing customer satisfaction by minimizing the students' walking time to bus stops. Mixed-integer linear programming (MILP) was applied by PuLP library in Python to solve this problem.

In the following sections, the data is represented, the mathematical formulation of the problem is discussed, and the optimal routes are proposed.

# Methodology

## Data Summary

To explore an implementation for capacitated school bus routing problem with a time window, the Sepehr Marefat high school is chosen as the real-world case. The school has 182 students, 109 of which registered their names to use school buses. These students' homes are dispersed over 14 distinct neighborhoods from three different districts, which indicates the school bus stops zones. Each neighborhood contains three bus stops result in 42 total stations. All the available school buses have a capacity of 30. All classes start at 8 A.M., and students must be present at school at 7:30 in the morning. The students' walking time from their homes to the optimal bus stops are calculated as follows:

$$t_{ir} = D_{ir}/66.67$$

Where, $t_{in}$ is the average spent time of all students who walk from their homes to bus stop i and live in neighborhood r, $D_{ir}$ is the Average distance from students' homes in neighborhood r to bus stop i, and 66.67 meters per minute is the rate of students' walking.

Moreover, the latitudes and longitudes of houses, the school, bus stops locations, and the map of the studied area were collected via openstreetmap.org website. Subsequently, the distance matrix was computed using Haversine formula in Python, which is explained in this fashion:

Suppose the latitude and longitude of point 1 and 2 on earth is known. The distance between these points, d, is calculated as follows:

$$a = sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos\varphi_1 * \cos\varphi_2 * sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$c = 2 * atan2(\sqrt{a}, \sqrt{1-a})$$

$$d = R * c$$

Where $\varphi$is latitude, $\lambda$ is longitude, and R is the earth's radius. It should be noted that, in this formula, angles are in radians.

Furthermore, the travel commencement time for each bus is determined from a set of suggested start times. Based on existing planning, the earliest bus begins its tour at 6 a.m, and the latest starts at 6:30 a.m. Therefore, in the proposed model, each bus must begin traveling between 6 and 6:30 in the morning. The departure time starts from 6 and is available every five minutes, so there are a total of seven options. Additionally, each bus spends a specific amount of time at each stops to pick up students. This consumed time is called stop duration in the dataset and is calculated regarding the population of each neighborhood.

# Mathematical Formulation

## Notation

| **Set** | **Definition** |
|---|---|
| $K = \{1,2,3,\dots,k,\dots,q\}$ | Set of available buses, where q is the optimal number of buses. |
| $B = \{1,2,3,\dots,i,j,\dots,42\}$ | Set of all bus stops (nodes). |
| $N = \{1,2,3,\dots,r,\dots,14\}$ | Set of all neighborhoods. |
| $P = \{1,2,3,\dots,109\}$ | Set of all students to be served. |
| **Parameters** | |
| $C_k$ | The capacity of bus k. |
| $p_r$ | Students' population in neighborhood r. |
| $p_i, p_s$ | Students' population at bus stops i and s. |
| $u_{ik}$ | Accumulated number of students at bus stop i served by bus k |
| $d_{irs}$ | Average distance between student s's home and stop i in neighborhood r. |
| $c_{ijk}$ | Cost (distance travelled) of route $(i,j)$ served by bus k, $\forall\, i \neq j$. |
| $ts_{irk}$ | Due time for all students to get off bus k in neighborhood r at stop i. |
| $Sd_{ik}$ | Stop duration of bus k at bus stop i. |
| $T_{ik}$ | Accumulated travel time and stop duration of bus k at stop i. |
| $t_{ijk}$ | Spent time of bus k to travel (i,j) route. |
| $L_k$ | The latest time for bus k to reach the school. |
| $S_k$ | Start time of bus k. |
| $ttl_k$ | Difference between $L_k$ and $S_k$, which indicates the travel time limit of bus k. |

| Decision Variables | |
|---|---|
| $x_{ijk}$ | Binary value, which denotes if bus k take $(i, j)$ route. |
| $b_i$ | Binary value, which denotes if bus stop i is selected. |
| $w_{ir}$ | Binary value, which denotes if stop i is in neighborhood r. |
| $p_{rs}$ | Binary value, which denotes if student s is in neighborhood r. |

It should be highlighted that all time parameters are reported in minute, distances in meters, and rate in meters per minute.

## Objective Functions

As mentioned earlier, three objective functions are examined simultaneously to evaluate the school buses route and schedule. The first objective is to minimize the total number of required buses. It is an objective related to the operational cost and is indispensable for both the benchmarking company and the school. To determine the minimum number of buses k to serve all points and collected all kids, subsequent equation is utilized:

$$q \; = \; min(k) \; s.t. \sum_{n=1}^{k} n \geq P$$

Minimizing the total walking time of students from their homes to bus stops is the second objective related to customer satisfaction. To estimate this objective, first, a student's distance to all bus stops in his neighborhood was calculated. Subsequently, using the rate formula, the spent walking time was computed. It should be pointed out that the average speed of students is considered 66.67 meters per minute. The objective is introduced as follows:

$$Minimize \quad \sum_{i \in B} \sum_{r \in N} \sum_{s \in P} d_{irs} w_{ir} p_{rs}$$

The third objective function minimizes the total distance travelled by school buses. Consequently, the traveling time is also minimized, for speed measures the distance travelled over the change in time. Here, the bus's average speed is constant and equals 833.33 meters per minute. This objective addresses both operational cost and customer satisfaction and ensures higher quality service. It is computed by the following function:

$$Minimize \sum_{i,j \in B} \sum_{k=1}^{q} c_{ijk} x_{ijk}$$

## Constraints

Various constraints have been considered for SBRP in this project, which are given and explained in this section.

$$\sum_{j\epsilon B}\sum_{k\epsilon K} x_{ijk} \geq 3, \quad \forall\, i = 1 \tag{1}$$

$$\sum_{j\epsilon B} x_{ijk} \leq 1, \quad \forall\, i \,\epsilon\, B \;\wedge\; k \,\epsilon\, K \tag{2}$$

$$\sum_{j\epsilon B\cup\{0\}} x_{ijk} = \sum_{j\epsilon B\cup\{0\}} x_{jik}, \quad \forall\, i \,\epsilon\, B \;\wedge\; k \,\epsilon\, K \tag{3}$$

$$\sum_{k\in K}\sum_{j\epsilon B\cup\{0\}} x_{ijk} = 1, \quad \forall\, i \,\epsilon\, B \tag{4}$$

$$\sum_{j\epsilon B} x_{0jk} \leq 1, \quad \forall\, k \,\epsilon\, K \tag{5}$$

$$\sum_{j\epsilon B} x_{0jk} = 1, \quad \forall\, k \,\epsilon\, K \tag{6}$$

$$x_{iik} = 0, \quad \forall\, i \,\epsilon\, B \;\wedge k \,\epsilon\, K \tag{7}$$

$$\sum_{i\epsilon B} b_i w_{ir} = 1, \quad \forall\, r \,\epsilon\, N \tag{8}$$

$$\sum_{k\epsilon K} u_{ik} \geq p_i, \quad \forall\, i \,\epsilon\, B \;\wedge i \;\neq 0 \tag{9}$$

$$u_{ik} \leq C_k, \quad \forall\, i \,\epsilon\, B \;\wedge k \,\epsilon\, K \tag{10}$$

$$if\ \left(p_i + p_j\right) >\ C_k \;\Rightarrow \sum_{j\epsilon B}\sum_{k\epsilon K} x_{ijk} = 0, \quad \forall\, i \,\epsilon\, B \;\wedge k \,\epsilon\, K \tag{11}$$

$$if\ \left(p_i + p_j\right) >\ C_k \;\Rightarrow \sum_{i\epsilon B}\sum_{k\epsilon K} x_{ijk} = 0, \quad \forall\, j \,\epsilon\, B \;\wedge k \,\epsilon\, K \tag{12}$$

$$u_{jk} - u_{ik} + \left(C_k - \left(p_i + p_j\right)\right) * x_{jik} + C_k * x_{ijk} \leq C_k - p_i, \quad \forall\, i,j \,\epsilon\, B \;\wedge i,j \neq 1 \;\wedge k \,\epsilon\, K \tag{13}$$

$$u_{jk} + \left(C_k - p_j\right) * x_{ijk} \leq C_k, \quad \forall\, i = 1 \;\wedge\; j \,\epsilon\, B \wedge k \,\epsilon\, K \tag{14}$$

$$\sum_{k\epsilon K} u_{jk} - \sum_{i\epsilon B}\sum_{k\epsilon K} x_{ijk} * p_i \geq p_j, \quad \forall\, i \;\neq 1 \wedge j \,\epsilon\, B \tag{15}$$

$$if\ \left(t_{j0k} + T_{jk}\right) > ttl_k \implies \sum_{i\epsilon B}\sum_{k\epsilon K} x_{ijk} = 0, \quad \forall\, j \,\epsilon\, B \;\wedge k \,\epsilon\, K \tag{16}$$

$$HT_k \in \{0,1,2\} \quad (17)$$

$$-59 < MT_k < 59 \quad (18)$$

$$0 \leq T_{ik} \leq ttl_k, \quad \forall\, i \in B \ \wedge k \in K \quad (19)$$

$$0 < t_{ijk} + Sd_{ik} + Sd_{jk} \leq ttl_k, \quad \forall\, i,j \in B \ \wedge i \neq j \ \wedge \ k \in K \quad (20)$$

$$t_{iik} = 0, \quad \forall\, i \in B \wedge \ k \in K \quad (21)$$

$$S_k + T_{ik} - Sd_{ik} \leq ts_{irk} \leq S_k + T_{ik}, \quad \forall\, i \in B \ \wedge r \in N \wedge \ s \in P \ \wedge \ k \in K \quad (22)$$

Constraint (1) ensures that the Hamiltonian cycle must occur, (2) and (4) implies that every bus stops leave only once and is visited by exactly one bus, (3) guarantees that the same bus arrives and departs from each node it serves, and (5) specifies that each bus used, at most, once. Moreover, constraint (6) indicates the travel begins at the benchmarking company, (7) ensures that no buses can drive the same origin and destination, (8) stipulates that each neighborhood is visited precisely once, and (9) that all students at a bus stop must be delivered only by a bus.

Constraints (10), (11), (12), and (13) are related to the capacity restriction. Constraint (10) ensures that the load on bus k when departing from node $i$ is always lower than the bus capacity, (11) and (12) points that a bus can proceed to neighborhoods where their population does not exceed the capacity, and (13) implies that the capacity of bus $k$ needs to be between $u_{ik}$ and $u_{jk}$, if it takes the $(i,j)$ route. Also, (14) states that if 1 is the first bus stop, then the accumulated students on the bus at stop $j$ is equal to or less than the students' population at stop $j$; in contrast, (15) signifies that if 1 is not the first bus stop, then the collected pupils on the bus at stop $j$ is equivalent or greater than the students' population at stop $j$.

Other constraints are associated with time. Prior to explaining the restrictions, it should be specified that values for each time parameter are calculated as follow:

$$Time \ = \ Rate \ * \ traveled \ distance$$

Where, Rate equals 833.33 meters per minute for buses and 66.67 meters per minute for students, and the distance travelled is the length between stops $i$ and $j$.

Constraint (16) assures that a bus can travel to a new neighborhood if it can advance to the school in due time.

Constraints (17) and (18) are connected to calculating the travel time limit of bus $k$ $(ttl_k)$. The difference between $L_k$ and $S_k$ are computed in this fashion: according to the school regulations, $L_k$ equals to 7:30 a.m. From the set of start times, $S_k$ is chosen for each bus. Then, the difference between hours of $L_k$ and $S_k$ are calculated, which is termed as $HT_k$.

Subsequently, the difference between the minute part of $L_k$ and $S_k$ are measured, and are termed as $MT_k$. In the end, $ttl_k$ is computed as follows:

$$ttl_k = HT_k * 60 + MT_k$$

Moreover, (19) implies that the accumulated time must not be negative and not exceed the travel time limit. The time spent at stops and through traveling must be positive and not surpass the travel time limit is addressed by (20). In addition, (21) points that travel time between the same stops must be zero, and (22) describes students' walking time range (Angelelli & Mansini, 2004) (Ripplinger, 2005) (Fügenschuh, 2009).

## Analysis

It is complicated to find a solution that optimizes all objectives in multi-objective problems, and sometimes no such answer exists at all. The designed mathematical model cannot be used directly to solve the problem because there is no known polynomial-time algorithm for solving such a multi-objective problem. Besides, the previously reported approaches cannot be practiced to this NP-hard problem due to the differences in the formulations. Thus, a new algorithm must be developed (Bektas & Elmastas, 2007).

In this study, a three-stage algorithm was employed to solve multiple objectives. Considering the mathematical formulation, the MILP approach is applied using PuLP library, an open-source package that allows mathematical programs to be described in Python. LpVariable, LpConstraint, and other objects were applied to create the model enabling a PuLP solver to operate properly. In the implementation process, the default solver of PuLP, prob.solve(), was used. The procedure of solving the problem is explained in the following paragraphs.

Initially, considering the total number of students and the capacity of the buses, the first objective function is assessed, and the minimum number of buses is concluded 4. Then, the customer satisfaction objective is taken into account to minimize the walking time of students to a bus stop in a given neighborhood. To choose the optimal bus stops in each community, the following algorithm is devised:

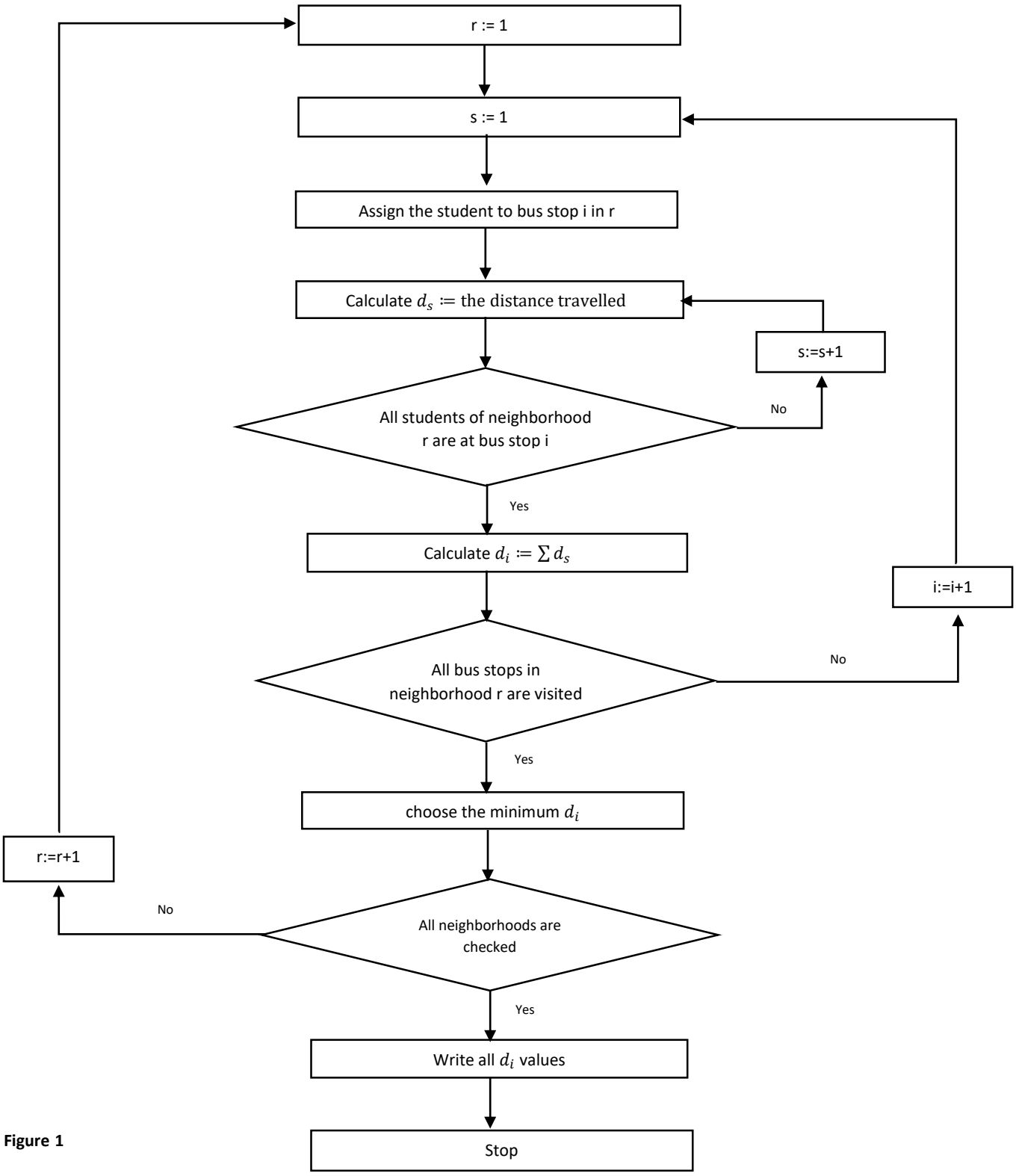



**Figure 1**

Subsequent to implementing this algorithm to the data, the optimal bus stops were found. Out of 42 bus stops, 14 stations, one for each neighborhood, were determined and assigned to numbers 1 to 14. In consequence, the final objective function, minimizing the

distance travelled and spent time of buses and students, was analyzed subject to the defined constraints. A simple algorithm of the third step is presented in Figure 2.

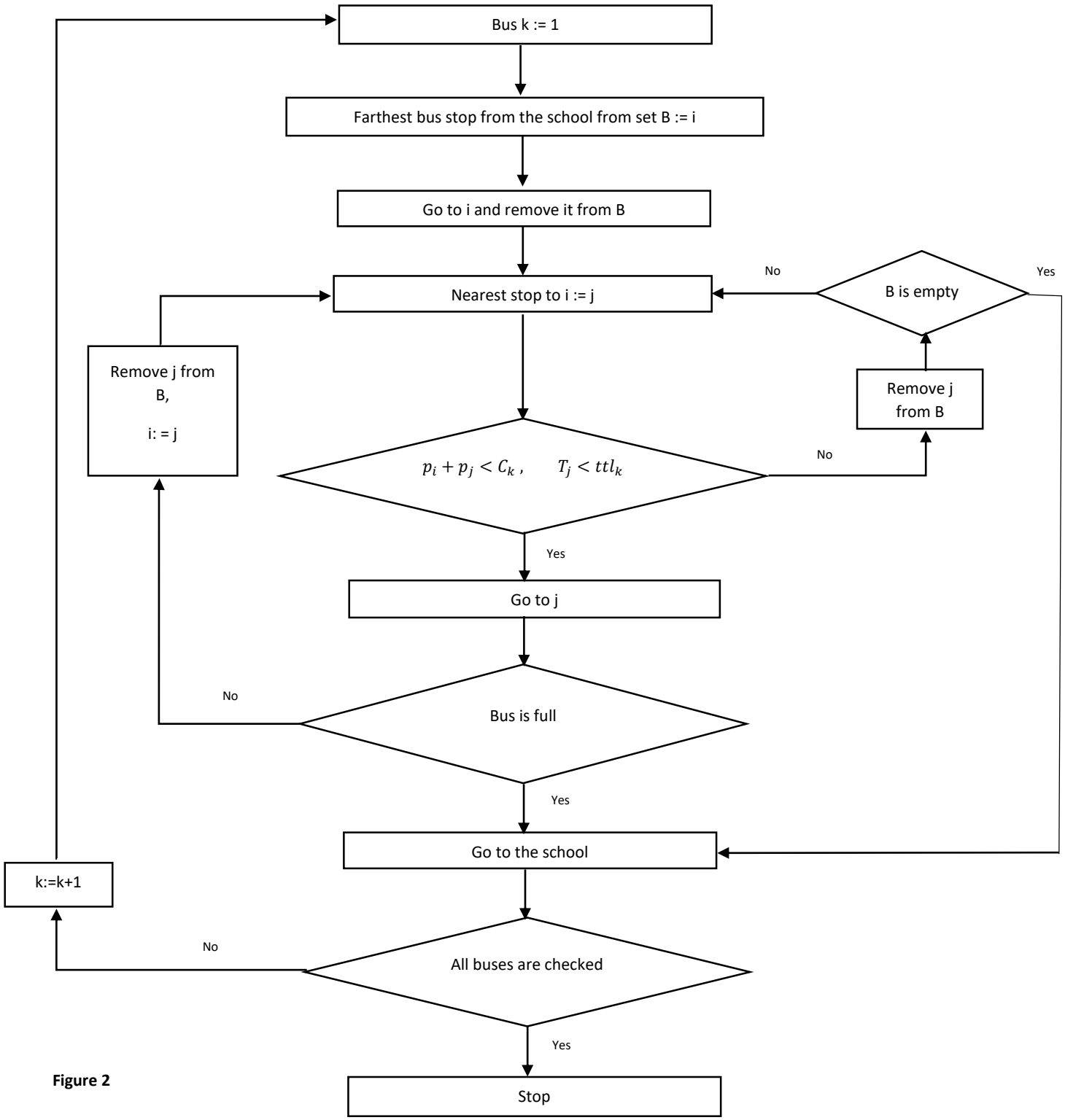



**Figure 2**

The model is applied to the benchmarking company to provide the most desirable service for Sepehr Marefat high school. Additionally, the results were compared to the existing planning for school buses, which was planned manually.

## Results

In the first step of implementing the model, four buses are set to serve pupils based on the school bus timetable. In the second step, 14 bus stops were selected, where school buses pick up students. The data of these stations are given in Table 1.

**Table 1**

| Neighbourhood | Neighbourhood ID | Stop ID | District | Number of Students | Latitude | Longitude | Total Walking Time | Stop Duration |
|---|---|---|---|---|---|---|---|---|
| NIAVARAN | OZ | 1 | 1 | 6 | 35.812600 | 51.479000 | 43.36 | 5 |
| AJUDANIYE | AJ | 2 | 1 | 9 | 35.801100 | 51.486355 | 51.84 | 5 |
| DEZASHIB | NI | 3 | 1 | 8 | 35.808335 | 51.453453 | 39.05 | 3 |
| CHIZAR | DE | 4 | 1 | 5 | 35.781100 | 51.441185 | 31.16 | 3 |
| TAJRISH | FA | 5 | 1 | 11 | 35.801731 | 51.433128 | 75.28 | 5 |
| FARMANIEH | TA | 6 | 1 | 13 | 35.797213 | 51.455697 | 69.35 | 5 |
| ELAHIYEH | QE | 7 | 1 | 10 | 35.775420 | 51.424446 | 58.61 | 3 |
| QEYTARIEH | CH | 8 | 1 | 7 | 35.792192 | 51.425480 | 65.90 | 5 |
| JORDAN | EL | 9 | 1 | 4 | 35.763152 | 51.417321 | 34.68 | 3 |
| EKHTIARIEH | JO | 10 | 3 | 4 | 35.772889 | 51.459549 | 28.48 | 3 |
| DARROUS | DA | 11 | 3 | 5 | 35.761201 | 51.445185 | 42.05 | 3 |
| PASDARAN | ME | 12 | 4 | 9 | 35.771375 | 51.486151 | 54.12 | 5 |
| MEHRAN | EK | 13 | 3 | 6 | 35.748472 | 51.454829 | 38.53 | 5 |
| OZGOL | PA | 14 | 4 | 12 | 35.790256 | 51.513537 | 59.38 | 3 |

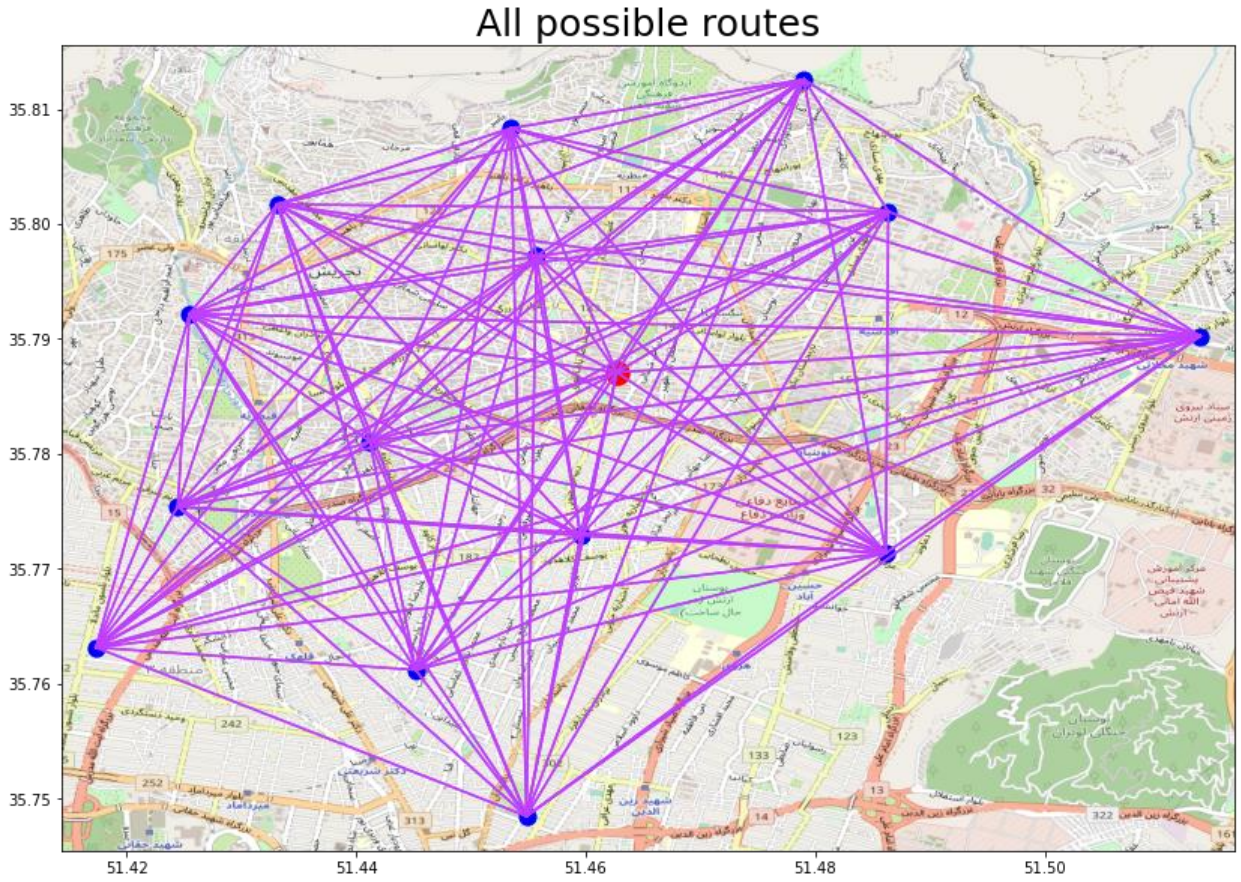
Moreover, the distance matrix of these bus stops was measured using Haversine formula, which is shown in the following table.

**Table 2**

| NAME | SCHOOL | NIAVARAN | AJUDANIYE | DEZASHIB | CHIZAR | TAJRISH | FARMANIYEH | ELAHIYEH | QEYTARIEH | JORDAN | EKHTIARIEH | DARROUS | PASDARAN | MEHRAN | OZGOL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SCHOOL | 0.000000 | 3.194888 | 2.641858 | 2.504954 | 2.052841 | 3.125230 | 1.290183 | 3.688249 | 3.405648 | 4.885241 | 1.606033 | 3.285910 | 2.745409 | 4.354943 | 4.600284 |
| NIAVARAN | 3.194888 | 0.000000 | 1.440989 | 2.352716 | 4.890299 | 4.310957 | 2.710760 | 6.428691 | 5.335149 | 7.824465 | 4.752873 | 6.480343 | 4.630595 | 7.458999 | 3.985466 |
| AJUDANIYE | 2.641858 | 1.440989 | 0.000000 | 3.075188 | 4.643003 | 4.802279 | 2.799387 | 6.273810 | 5.580673 | 7.524598 | 3.961801 | 5.787611 | 3.306306 | 6.508530 | 2.732895 |
| DEZASHIB | 2.504954 | 2.352716 | 3.075188 | 0.000000 | 3.225205 | 1.975193 | 1.253537 | 4.500381 | 3.097198 | 5.990532 | 3.980810 | 5.295559 | 5.060041 | 6.659728 | 5.781437 |
| CHIZAR | 2.052841 | 4.890299 | 4.643003 | 3.225205 | 0.000000 | 2.407232 | 2.219579 | 1.637325 | 1.878894 | 2.936604 | 1.892178 | 2.242649 | 4.199436 | 3.832439 | 6.607297 |
| TAJRISH | 3.125230 | 4.310957 | 4.802279 | 1.975193 | 2.407232 | 0.000000 | 2.097197 | 3.029672 | 1.265681 | 4.521977 | 3.996931 | 4.637670 | 5.855760 | 6.239362 | 7.365865 |
| FARMANIYEH | 1.290183 | 2.710760 | 2.799387 | 1.253537 | 2.219579 | 2.097197 | 0.000000 | 3.718481 | 2.782808 | 5.132765 | 2.727780 | 4.116435 | 3.976249 | 5.422050 | 5.275466 |
| ELAHIYEH | 3.688249 | 6.428691 | 6.273810 | 4.500381 | 1.637325 | 3.029672 | 3.718481 | 0.000000 | 1.867908 | 1.508427 | 3.180296 | 2.450425 | 5.586614 | 4.062531 | 8.206623 |
| QEYTARIEH | 3.405648 | 5.335149 | 5.580673 | 3.097198 | 1.878894 | 1.265681 | 2.782808 | 1.867908 | 0.000000 | 3.312904 | 3.749742 | 3.878775 | 5.944132 | 5.537497 | 7.947794 |
| JORDAN | 4.885241 | 7.824465 | 7.524598 | 5.990532 | 2.936604 | 4.521977 | 5.132765 | 1.508427 | 3.312904 | 0.000000 | 3.962066 | 2.524348 | 6.278989 | 3.758863 | 9.191141 |
| EKHTIARIEH | 1.606033 | 4.752873 | 3.961801 | 3.980810 | 1.892178 | 3.996931 | 2.727780 | 3.180296 | 3.749742 | 3.962066 | 0.000000 | 1.836002 | 2.406562 | 2.749162 | 5.240611 |
| DARROUS | 3.285910 | 6.480343 | 5.787611 | 5.295559 | 2.242649 | 4.637670 | 4.116435 | 2.450425 | 3.878775 | 2.524348 | 1.836002 | 0.000000 | 3.866560 | 1.662006 | 6.963578 |
| PASDARAN | 2.745409 | 4.630595 | 3.306306 | 5.060041 | 4.199436 | 5.855760 | 3.976249 | 5.586614 | 5.944132 | 6.278989 | 2.406562 | 3.866560 | 0.000000 | 3.805595 | 3.243091 |
| MEHRAN | 4.354943 | 7.458999 | 6.508530 | 6.659728 | 3.832439 | 6.239362 | 5.422050 | 4.062531 | 5.537497 | 3.758863 | 2.749162 | 1.662006 | 3.805595 | 0.000000 | 7.047964 |
| OZGOL | 4.600284 | 3.985466 | 2.732895 | 5.781437 | 6.607297 | 7.365865 | 5.275466 | 8.206623 | 7.947794 | 9.191141 | 5.240611 | 6.963578 | 3.243091 | 7.047964 | 0.000000 |

Furthermore, all available routes between these stops are shown in the following graph, where the red dot indicates the school.

**Figure 3**



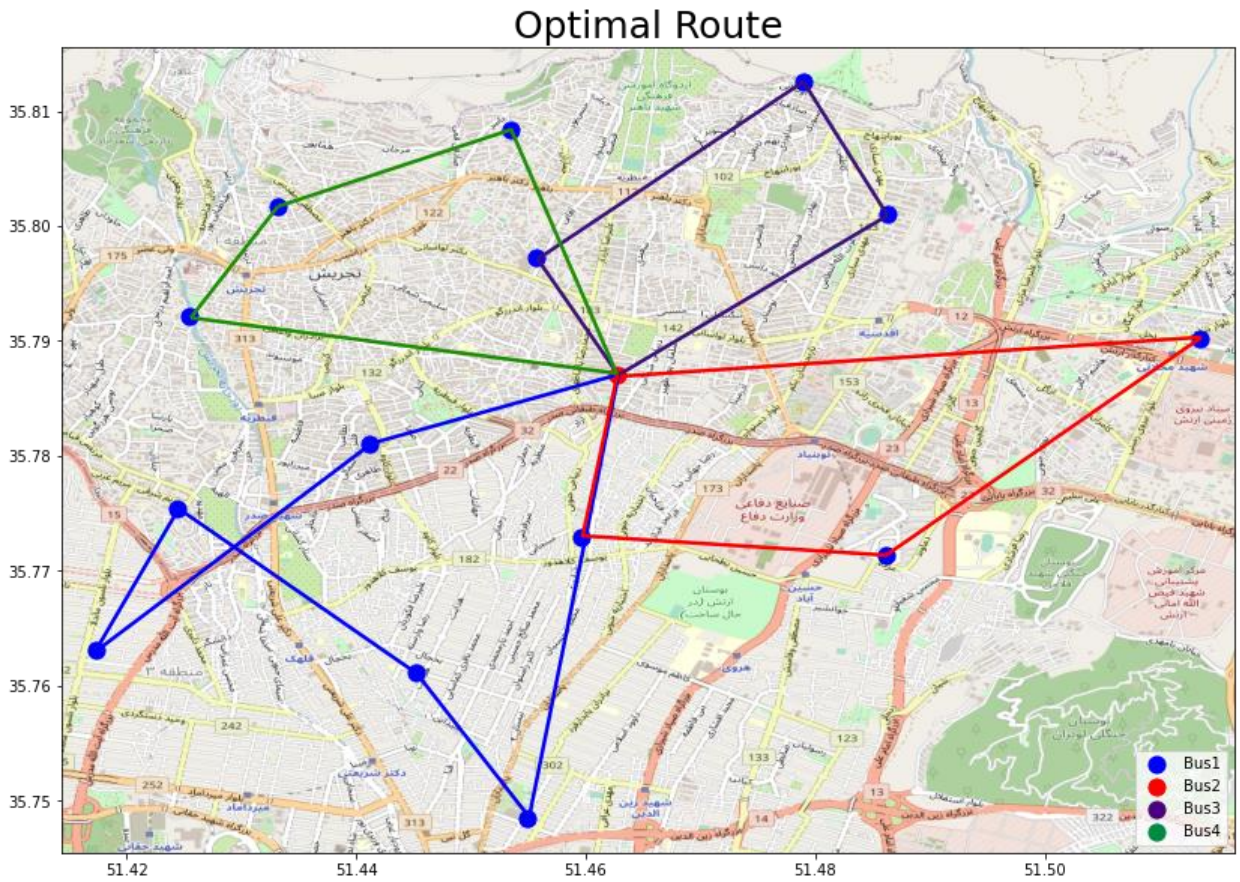

In the final step, the optimal route for the four buses subject to capacity and time window constraints were explored from all available courses. The optimal routes are arranged as follows:

**Table 3**

| | Delivered students | Optimal route |
|---|---|---|
| Bus 1 | 30 | SC → ME → DA → EL → JO → CH→ SC |
| Bus 2 | 28 | SC → OZ → PA → EK → SC |
| Bus 3 | 21 | SC → AJ → NI → FA → SC |
| Bus 4 | 30 | SC → QE → TA → DE → SC |

The optimal solution for each bus is shown in Figure 4.

**Figure 4**




In the end, the results of the existing planning and the optimal model based on the proposed mathematical formulation were compared to evaluate its performance. The findings are shown in the following tables.

**Table 4**

| Route | Number of Buses | Minimum Distance (MD) | Total Distance Travelled | Percentage above (MD) | Total Travel Time | Total Walking Time | Time Window |
|---|---|---|---|---|---|---|---|
| Optimal | 4 | 82900 | 102400 | 23.52% | 229.34 | 691.79 | Satisfied |
| Existing | 4 | 82900 | 117400 | 41.62% | 284.06 | 865.39 | Satisfied with random delays |

**Table 5**

| Bus | Distance Travelled of ER | Distance Travelled of OR | Shortened Distance | % of Shortened Distance | Travel Time of ER | Travel Time of OR | Shortened Time | % Of Shortened Time |
|---|---|---|---|---|---|---|---|---|
| Bus 1 | 46300 | 40700 | 5600 | 12.10% | 91.62 | 83.90 | 7.72 | 8.43% |
| Bus 2 | 35300 | 30400 | 4900 | 13.89% | 71.40 | 59.52 | 11.88 | 16.64% |
| Bus 3 | 18500 | 14000 | 4500 | 24.32% | 69.24 | 43.84 | 25.40 | 36.68% |
| Bus 4 | 17300 | 14200 | 3100 | 17.92% | 51.80 | 42.08 | 9.72 | 18.76% |

## Conclusion

This study presents a practical solution for a school bus benchmarking company that wants to provide services for Sepehr Marefat High School in Tehran. For this NP-hard problem, MILP model was created using PuLP library in Python. A set of data was first generated to assess the formulation. Next, for the computational experiment, the real-world data was collected from the school. Also, the detailed data of the existing scheduling was acquired from the benchmarking company to compare with the proposed model results.

Furthermore, the average traffic time for vehicles in Districts 1, 3, and 4 of Tehran is considered and calculated based on Average Daily Traffic (ADT) in the previous year's Fall and adds to the travel time of each bus.

The findings reveal that four buses are required to pick up all of the student subject to the capacity constraint. Also, the optimal solution of the recommended model reduces operational costs and enhances customer satisfaction. More precisely, total distance travelled declined by 12.78%, travel time of buses lowered by 19.26%, and total walking time decreased by 20.06%, which are significant advancements. Moreover, the total distance travelled is 23.52% above the sum of all minimum routes directly from stops to the school, which is an acceptable value compared to that of the existing plan, which equals 41.62%. Besides, in table 5, it is observed that the model decreases operational costs for each bus.

# References

Angelelli, E. & Mansini, R., 2004. The Vehicle Routing Problem with Time Windows and Simultaneous Pick-up and Delivery. *Quantitative Approaches to Distribution Logistics and Supply Chain Management,,* pp. 249-267.

Bektas & Elmastas, 2007. Solving school bus routing problems through integer programming. *Journal of the Operational Research Society,* 58(12), p. 1599–1604.

Fügenschuh, A., 2009. Solving a school bus scheduling problem with integer programming. *European Journal of Operational Research ,* 193(3), p. 867–884.

Park, J. & Kim, B. I., 2010. The school bus routing problem: A review. *European Journal of Operational Research,* 202(2), p. 311–319.

Ripplinger, D., 2005. Rural school vehicle routing problem. *Transportation Research,* 1922(1), p. 105–110.

Sadaghiani, J., 2010. Solving Traveling Salesman Problem Using Ant Colony Optimization. *Industrial Management Study,* 8(18), pp. 105-122.

# Python Code

## Libraries

```
import pandas as pd
import numpy as np
from pulp import *
from math import ceil, sin, cos, sqrt, atan2, radians
import matplotlib.pyplot as plt
```

### Generate a data set

```
def data_generator(mainfile, walking_distance, cost_matrix, Bus_travel_time, Bus_travel_time_limit, parameter,  buscapacity=30):

    dataframe = pd.ExcelFile(excel)
    sheet_1, sheet_2 = dataframe.parse(Cost_matrix), dataframe.parse(parameter)

    data = {}

    data['Stops_Locations'] = list(sheet_1.columns[1:].dropna())
    data['Homes_Locations'] = list(sheet_1.columns[2:].dropna())
    data['Student_num'] = sheet_2['Student_num'].dropna()
    data['Bus'] = sheet_2['Bus'].dropna()
    data['Capacity'] = list(sheet_2['Capacity'].dropna().astype('int32'))
    data['start_time'] = sheet_2['Start time'].dropna()
    data['earliest'] = sheet_2['Earliest time'].dropna()
    data['latest'] = sheet_2['Latest time'].dropna()
    data['stop_duration'] = sheet_2['Stop_duration'].dropna()
    data['Total_wlaking_time'] = sheet_2['Total Wlaking time'].dropna().astype('float32')
    data['Student_rate'] = sheet_2['Student rate'].dropna().astype('float32')
    data['Bus_rate'] = sheet_2['Bus rate'].dropna().astype('float32')
    data['Bus_travel_time_limit'] = sheet_2['Bus travel time limit'].dropna().astype('float32')

    walking_distance = {}
    for Stops_i in range(data['Stops_Locations'].__len__()):
        for Homes_j, cost in enumerate(sheet_1.loc[Stops_i,data['Homes_Locations']]):
            walking_distance.update({'{}_{}'.format(str(Stops_i), str(Homes_j)): int(cost)})
    data['walkingdistance'] =  walking_distance

    cost_matrix = {}
    for Neighborhood_i in range(data['Stops_Locations'].__len__()):
        for Neighborhood_j, cost in enumerate(sheet_1.loc[Neighborhood_i,data['Stops_Locations']]):
            for idx, capacity in enumerate(data['Capacity']):
                if Neighborhood_i != Neighborhood_j:
                    cost_matrix.update({'{}_{}_{}'.format(str(Neighborhood_i), str(Neighborhood_j), str(idx)): ceil(cost*(capacity/buscapacity))})
    data['cost_matrix']=cost_matrix

    Bus_travel_time = {}
    for Stops_i in range(data['Stops_Locations'].__len__()):
        for Stops_j, cost in enumerate(sheet_1.loc[Stops_i,data['Homes_Locations']]):
            for Bus_k, rate in enumerate(data['Capacity'].__len__()):
                if Stops_i != Stops_j:
                    Bus_travel_time == (cost/rate)
    data['Bus_travel_time']= Bus_travel_time

    Bus_travel_time_limit = {}
    for Bus_k in range(data['Capacity'].__len__()):
        for earliest in enumerate(data['earliest']):
```

```
        for latest in enumerate(data['latest']):
            Bus_travel_time_limit == (latest-earliest)
    data['Bus_travel_time_limit']= Bus_travel_time_limit

    return data



mainfile = 'SBRP.xlsx'
cost_matrix = 'cost_matrix_toy'
parameters = 'parameters_toy'
data  = data_generator(mainfile, Cost_matrix, parameters)

Bus_num = data['Capacity'].__len__()
cost_matrix = data['cost_matrix']
```

## Variables

```
x = LpVariable.dicts('x', [i for i in cost_matrix.keys()], lowBound=0, upBound=1, cat='Binary')
b = LpVariable.dicts('b', [i for i in Bus_num.keys()] lowBound=0, upBound=1, cat='Binary')
w = LpVariable.dicts('w', [i for i in Stops_Locations.keys()] lowBound=0, upBound=1, cat='Binary')
p = LpVariable.dicts('p', [i for i in Student_num.keys()] lowBound=0, upBound=1, cat='Binary')
u = LpVariable.dicts('u', [f"{i}_{j}" for i in range(data['Stops_Locations'].__len__()) for j in range(Bus_num)], lowBound=0,
upBound=max(data['Capacity']), cat='Integer')
T = LpVariable.dicts('T', [f"{i}" for i in range(data['Stops_Locations'].__len__())], lowBound=0, cat='integer')
t = LpVariable.dicts('t',[i for i in range Bus_travel_time.keys()],lowBound=0, upBound=max(data['Bus_travel_time'], cat='Continuous')
ttl = LpVariable.dicts('ttl',[i for i in Bus_travel_time_limit.keys()], lowBound=0, upBound=179, cat='Continuous')
```

# objectives

prob = LpProblem('School_Bus_Routing_Problem', LpMinimize)

### First objective function

```
min_Bus = ceil( data['Student_num'].sum() / (30))
print('{} buses is required.'.format(min_Bus))
```

### Second objective function

```
prob += lpSum( w[key] * p[key] * walking_distance[key] for key in walking_distance.keys() )
```

### Third objective function

```
prob += lpSum( x[key] * cost_matrix[key] for key in cost_matrix.keys() )
```

## Constraints

```
#1
prob += lpSum( x[i] for i in cost_matrix.keys() if (i.split('_')[0]=='0') ) >= 3

#2
for k in range(Bus_num):
    prob += lpSum( x[i] for i in cost_matrix.keys() if (i.split('_')[0]=='0') and (i.split('_')[2]==str(k)) ) <= 1
```

```
#3
for k in range(Bus_num):
   for i in range(len(data['Stops_Locations'])):
      prob += lpSum( x[f'{i}_{j}_{k}']-x[f'{j}_{i}_{k}'] for j in range(len(data['Stops_Locations'])) if i!=j ) == 0

#4
for k in range(Bus_num):
   for i in range(len(data['Stops_Locations'])):
      prob += lpSum( x[f'{i}_{j}_{k}'] for j in range(len(data['Stops_Locations'])) == 1
#5,6

for k in range(Bus_num):
   for j in range(len(data['Stops_Locations'])):
      prob += lpSum( x[f'{'0'}_{j}_{k}']) == 1
#7
for k in range(Bus_num):
   for i in range(len(data['Stops_Locations'])):
      prob += lpSum( x[f'{i}_{i}_{k}']) == 0

#8

for r in range(len(data['Neighborhood'])):
   for i in range(len(data['Stops_Locations'])):
      prob += lpSum( b[f'{i}'] * w[f'{i}_{r}']) ==1



for j in range(1,len(data['Stops_Locations'])):

#9,10

   prob += lpSum( u[f'{j}_{i}'] for i in range(Bus_num) ) >= data['Student_num'][j]
      for i in range(Bus_num):
         prob += u[f'{j}_{i}'] <= data['Capacity'][i]
#11
   prob += lpSum( x[i] for i in cost_matrix.keys() if (i.split('_')[0]==f'{j}') and
               (data['Student_num'][int(j)]+data['Student_num'][int(i.split('_')[1])]<=data['Capacity'][int(i.split('_')[2])]) ) == 1

#12
   prob += lpSum( x[i] for i in cost_matrix.keys() if (i.split('_')[1]==f'{j}') and
               (data['Student_num'][int(j)]+data['Student_num'][int(i.split('_')[0])]<=data['Capacity'][int(i.split('_')[2])]) ) == 1

#13
   for i in range(1, len(data['Stops_Locations'])):
      if j!=i :
         for k in range(Bus_num):
            Buscap = data['Capacity'][k]
            Student_num_j = data['Student_num'][j]
            Student_num_i = data['Student_num'][i]
            prob += u[f'{j}_{k}'] >= (u[f'{i}_{k}'] + Student_num_j - Buscap) + Buscap*(x[f'{j}_{i}_{k}']+x[f'{i}_{j}_{k}']) -
((Student_num_j+Student_num_i)*x[f'{j}_{i}_{k}'])

#14
   for k in range(Bus_num):
      Buscap = data['Capacity'][k]
      Student_num = data['Student_num'][j]
      current_node = x[f'0_{j}_{k}']
      prob += u[f'{j}_{k}'] <= ( Buscap - ( Buscap - Student_num) * current_node )
#15
   Student_num = data['Student_num'][j]
```

```
    temp = lpSum(u[f'{j}_{i}'] for i in range(Bus_num)) - lpSum( (x[f'{Neighborhood}_{j}_{i}'] * data['Student_num'][Neighborhood]) for
Neighborhood in range(1,len(data['Stops_Locations'])) if Neighborhood != j for i in range(Bus_num) )
    prob += temp >= Student_num

#16

    for k in range(Bus_num):
        prob += lpSum( x[f'{i}_{j}_{k}'] for i in in range(len(data['Stops_Locations'])) and
                (t[f'{j}_{0}_{k}'] + T[f'{j}_{k}']) > ttl[f'{k}']

#19
    for k in range(Bus_num):
        prob += lpSum( T[f'{j}_{k}']) >= 0 and
                ( T[f'{j}_{k}']) <= ttl[f'{k}']
#20
    for k in range(Bus_num):
        prob += lpSum( t[f'{i}_{j}_{k}'] for i in in range(len(data['Stops_Locations'])) + data['stop_duration'][i] + data['stop_duration'][j]) <= ttl[f'{k}']

#21
    for k in range(Bus_num):
        t[f'{j}_{j}_{k}'] == 0
```

## Specific time window constraint

```
T_Max = 360

for j in range(1,len(data['Stops_Locations'])):


    for I in range(len(data['Stops_Locations'])):
        if j != I:
            x_agg = lpSum(x[f'{I}_{j}_{i}'] for i in range(Bus_num))
            temp = t[f'{I}'] + ( (data['stop_duration'][I]+data['Bus_rate']*dmdf[f'{I}_{j}']) * x_agg ) - ( data['latest'][I] * (1-x_agg) )
            prob += t[f'{j}'] >= temp

    prob += t[f'{j}'] >= data['earliest'][j]
    prob += t[f'{j}'] <= data['latest'][j]

    prob += t[f'{j}'] + data['stop_duration'][j] + data['Bus_rate']*dmdf[f'{j}_0']*lpSum( x[f'{j}_0_{i}'] for i in range(Bus_num) ) <= T_Max
```

## Solving the problem

```
%time prob.solve()
print("Operation_status = {}\n".format(LpStatus[prob.status]))
print("Optimal_value = {}\n".format(value(prob.objective)))

def get_next(route):
    travel = [route[0]]
    _range_ = route.__len__()-1
    for i in range(_range_):
        start = travel[-1].split("_")[1]
        temp = [ arc for arc in route if arc.split("_")[0] == start ]
        travel.append(temp[0])
    return travel

result = [ k for k, v in x.items() if value(v)!=None and value(v)>0 ]

route_list = []
for i in range(Bus_num):
```

```
    route_list.append([ f"{r.split('_')[0]}_{r.split('_')[1]}" for r in result if r.split('_')[2] == str(i) ])

travel, v_list = [], []
for idx, r in enumerate(route_list):
    if len(r) != 0:
        travel.append(get_next(r))
        v_list.append(idx)
```

## Final Result

```
print("Operation_status = {}\n".format(LpStatus[prob.status]))
print("Optimal_value = {}\n".format(value(prob.objective)))
for idx, r in zip(v_list, travel):
    print( f"● Bus({idx}) Route (Capacity={data['Capacity'][idx]}) : "+' -> '.join( [ data['Location'][int(i.split('_')[0])]+"
({})".format(str(data['Student_num'][int(i.split('_')[0])])) for i in r]
                                                    + [ data['Location'][0]+" ({})".format(str(data['Student_num'][0]))])
                                                    + " →→→ Number of students on the bus = {}
students.".format(sum(data['Student_num'][int(i.split('_')[0])] for i in r)))
```

## Plotting the result

```
ll = pd.read_csv('LATLON.csv')

n = int(input('Enter number of Stops = '))  # number of stops
N = [i for i in range(1,n+1)]  # Create set of nodes
q = {i:np.random.randint(1,10) for i in N}  # Create demands at Stops loactions
V = [0] + N  # Add the Wirehouse into the set of nodes
A = [(i,j) for i in V for j in V if i!=j]

BBox = ((ll.LON.min()-0.003, ll.LON.max()+0.003,
        ll.LAT.min()-0.003, ll.LAT.max()+0.003))

teh_map = plt.imread('tehmap.png')
```

## Plot the map

```
fig, ax = plt.subplots(figsize = (14,12))
ax.scatter(ll.LON[1:], ll.LAT[1:], zorder=1, alpha= 1, c='b', s=74)
ax.scatter(ll.LON[0], ll.LAT[0], c='r', s=74)
ax.set_title('Plotting Spatial Data')

ax.imshow(teh_map, zorder=0, extent = BBox, aspect= 1)
```

## All possible route

```
f, ax1 = plt.subplots(1,figsize = (14,12))


ax1.scatter(ll.LON[1:], ll.LAT[1:], c = 'b', s=124)
ax1.scatter(ll.LON[0], ll.LAT[0], c='r', s=240)
ax1.set_title('All possible routes', fontsize = 25)
ax1.imshow(teh_map, zorder=0, extent = BBox, aspect= 1)

for i,j in A:
```

```
    ax1.plot([ll.LON[i],ll.LON[j]], [ll.LAT[i], ll.LAT[j]], c = '#BF3EFF')
```

## Optimal route

```
f, ax2 = plt.subplots(1, figsize = (14,12))


# Optimal Route
ax2.scatter(ll.LON[1:], ll.LAT[1:], c = 'b', s=124)
ax2.scatter(ll.LON[0], ll.LAT[0], c='r', s=124)
ax2.set_title('Optimal Route', fontsize = 25)
ax2.imshow(teh_map, zorder=0, extent = BBox, aspect= 1)
ax2.legend(['Bus1','Bus2','Bus3','Bus4'], loc='lower right')

for i,j,k in result:
    if k == 0
        ax2.plot([ll.LON[i],ll.LON[j]], [ll.LAT[i], ll.LAT[j]], c = 'b')
    elif k== 1
        ax2.plot([ll.LON[i],ll.LON[j]], [ll.LAT[i], ll.LAT[j]], c = 'r')
    elif k==2
        ax2.plot([ll.LON[i],ll.LON[j]], [ll.LAT[i], ll.LAT[j]], c = '#4B0082')
    else
        ax2.plot([ll.LON[i],ll.LON[j]], [ll.LAT[i], ll.LAT[j]], c = '#008B45')
```