EAI/Springer Innovations in Communication and Computing

Mohammad M. Banat
Sara Paiva *Editors*

Smart Technologies for Smart Cities

Springer

Mohammad M. Banat • Sara Paiva
Editors

Smart Technologies for Smart Cities

Springer

Contents

Development of a Java Library to Solve the School Bus Routing Problem

Alberto Ochoa-Zezzatti, Ulises Carbajal, Oscar Castillo, José Mejía, Gilberto Rivera, and Saúl Gonzalez

1 Introduction

Each day, buses make two trips. In the morning, students are picked up at their homes, while in the afternoon, buses transport students from the school to their homes [3]. However, bus stops are not generated by area, where the number of students is concentrated, and thus, reduce the number of stops that would be made by improving the transfer time. There is an additional generalization to the problem, which is to allow transfers, that is, students who attend different schools can share a single bus and change it during their route to school [5]. However, the waiting time is linked to the number of transfers, the more transfers as the time increases. The authors of the study chose to solve the SBRP by means of an algorithm of optimization of colony of ants, because they were developed to find good solutions, in a time of reasonable calculation for routing problems [3], which was used to solve the routing phase, in order to find the best route. When analyzing the results, it was observed that some parameters have greater impact than others and that the distance of the routes can be reduced, but this would increase the time; the problem was solved by a two-phase scheme where the place where they would be assigned was assigned. Students were collected, and the second phase consisted of the optimization of the routes, through the algorithm of colony optimization of ants, and obtained an improvement of 15.2% in the cost and in the time of travel as well

A. Ochoa-Zezzatti (✉) · U. Carbajal · J. Mejía · G. Rivera · S. Gonzalez
Universidad Autónoma de Ciudad Juárez, Ciudad Juárez, Mexico
e-mail: alberto.ochoa@uacj.mx

O. Castillo
Instituto Tecnologico de Tijuana, Tijuana, Mexico

M. M. Banat, S. Paiva (eds.), *Smart Technologies for Smart Cities*,
EAI/Springer Innovations in Communication and Computing,
https://doi.org/10.1007/978-3-030-39986-3_9

as to take them to class in the morning and back to their homes. An estimate of the distance traveled is made; for all buses, the requirements of the school are that students should not travel more than 25 kilometers [4].

1.1 Related Research

A combination of transport arrangements with different capacities is proposed [2]. Considering several means of transport at the same time (cars, vans, buses), it is observed that the routes are not generated, but the stops. The SBRP has a total of five subproblems, including data preparation, bus stop selection, bus route generation, school bell time adjustment, and route scheduling, which can be seen in Fig. 1.

These subproblems are closely related to each other and must be resolved in an integrated manner [5]. In each proposed study, you can find different restrictions, such as the distances between bus stops, the time of arrival to classes, and the capacity of the buses, applying the restrictions, and in a case where it was necessary to transport 519 students to a primary school, the number of buses from 26 to 18 was minimized, and the bus utilization average improved from 60.49% to 87.37% [1]. For this, an algorithm was developed that can solve part of the SBRP, which when applied can show the efficiency compared with manual methods and thus is able to reduce the distances traveled. A model is analyzed to optimize routes, by grouping people traveling with the same destination. Using the algorithm DBSCAN, this algorithm takes advantage of the density of the points to determine the groups; this indicates that a circle is defined around this, obtaining its neighbors. For the

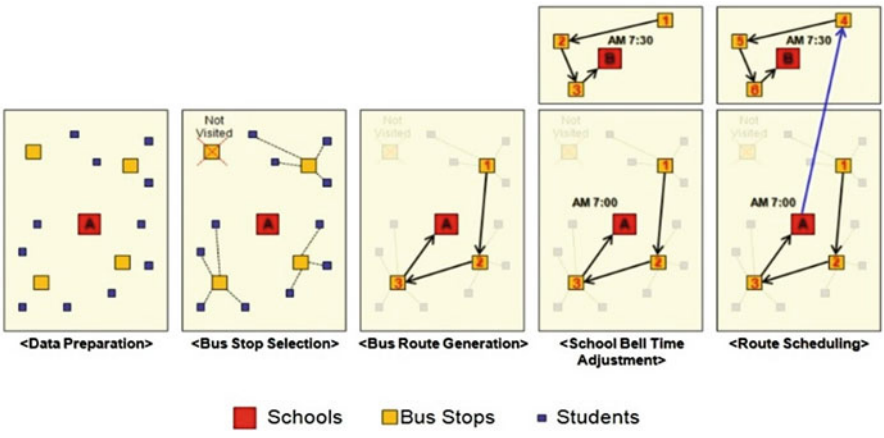

Fig. 1 Subproblems Involved school bus routing problem

existence of a cluster, it is necessary to have a minimum number of points within this cluster, that is, it is necessary to have a minimum number of neighbors of the point. Therefore, the algorithm iterates through the data set, verifies dense regions, and associates the neighboring points with the same clusters, taking into account the condition of the minimum number of points per group [2]. The algorithm considers the density of the points; sometimes the minimum number of clusters is not counted, so these points are taken as noise. Tests were conducted with a population of 437 people, where the DBSCAN algorithm generated 33 clusters for a total of 394 people representing 90.2% of the total; the rest of the population was taken as noise; this is due to the low density in the area which they are from. The results obtained are satisfactory for the authors, since they can verify that the time of transfer of the people can be reduced, and, thus, reduce the traffic of the city, to reduce the number of vehicles on the roads. Several studies have been analyzed with respect to the School Bus Routing Problem, where each of them has been implemented in different environments, with different restrictions, such as distance and time, or having as objective only the creation of bus stops. Not all show the improvements and optimizations that could be obtained by applying the algorithm, and some of them are not developed as a program, and a few are not easy to implement in other environments, since they are designed for a specific environment. It is proposed to generate the routes in an optimized way, and only buses will be used, since it is focused on a school district. It can be observed that similar works have already been implemented satisfactorily; with this basis, it is considered possible to implement a similar tool for Juarez City.

1.2 Problem Statement

The SBRP has been studied constantly since its first publication by the authors Rita M. Newton and Warren H. Thomas in 1969. Since most of the studies have been carried out thanks to the appearance of the day-to-day problems with their own restrictions, we do not have an overview of the SBRP [13]. The SBRP can be solved in five steps as already mentioned, which are data preparation, selection of stops, route development, time adjustment, and route planning. Although these subproblems are not independent, but are highly interrelated, they are treated separately due to the complexity and size of the problem. Although the SBRP itself is a unique and independent problem, a single subproblem or a combination of its subproblems can be classified as a variant of existing optimization problems. By itself, the subproblem of generation of bus routes is very similar to the vehicle routing problem (VRP) since it seeks to generate efficient routes; it also has part of the traveling salesman problem (TSP) since in this they have to travel efficiently and through all cities; in SBRP, you have to go through all the bus stops.

2 Framework of Reference

A library programmed in Java was realized, with a basic scheme of a database that stores the coordinates of the school, the students, and the bus stops. The library was able to take the coordinates of the database, calculate the distance between stops, and find an optimal route between them. By making these processes work together, the library can be used to solve the SBRP regardless of the region in which the problem is located, that is, whether it is in the urban area or in a rural region. For the operation, you must have the data of the coordinates, as well as the school and those of the bus stops loaded in the database; these data are necessary for the proper functioning of the library. The library is designed to be used by calling two functions within it, getDatos and neighborC; with these functions, the information was extracted from the database, and the optimal route was calculated.

2.1 Smart Cities

It is a term that began to be used in the 1990s to refer to cities where, thanks to the integration of information and communication technologies (ICT), energy and transport have progressed [6]. They begin by modernizing their critical infrastructure to provide sustainable development, help improve the efficiency of available resources, and improve the quality of life of citizens [7]. Among the advantages presented by smart cities are improving the efficiency of public administrations, interconnecting the different systems that make up the cities, facilitating the management of a greater number of public services, reducing the traffic of the transportation systems, and improving the speed of response in emergencies [6]. Although there are multiple areas that benefit from the use of ICT within smart cities, this work is closely related to the area of public transport, given that Cd. Juarez currently does not have a technological implementation for the use of public transportation that serves citizens in their mobilization within it.

2.2 Graph Theory

The first step to solve the SBRP is the preparation of the data, which refers to having the roads, the geographical location of the school, and the students. As a sample of them, 90 points were taken to simulate the students. With the help of Google Maps, it was possible to create a map with the geographical location of the school and the students; also the coordinates of each point could be obtained. In Fig. 2, you can see the locations of the students in color and the school.

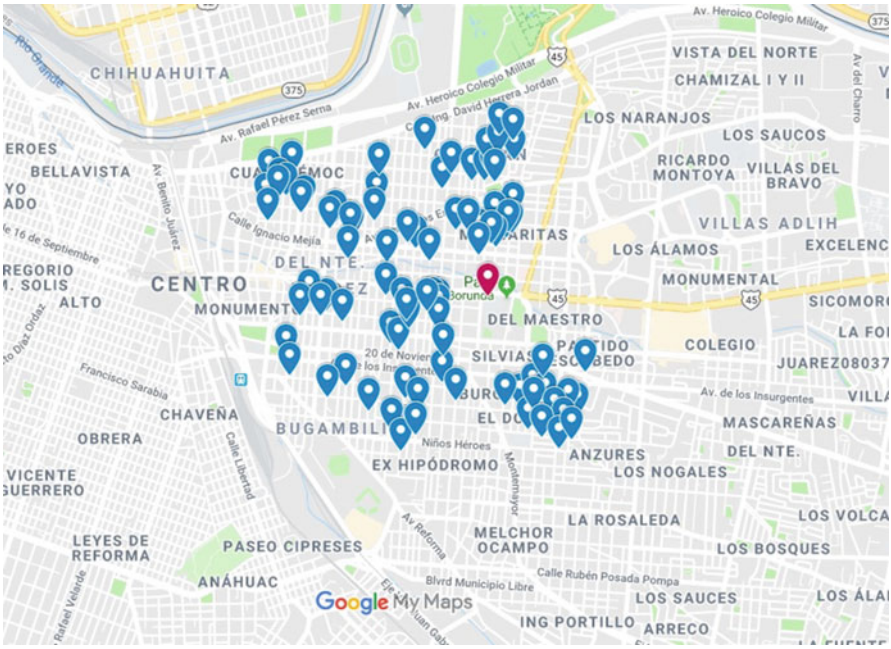

Fig. 2 Location of the points on the map

2.3 Stop Selection

By having the geographical location of the school and the students, we proceeded to select the bus stops that would be visited for the promotion and descent of the students. These do not have a specific distance between them; they were located in the points with student population, because it was proposed that students should not travel more than 500 meters of walking distance. In Fig. 3, the marks of the points are identified which were divided by color; the blue marks represent the location of the students, the green marks represent the locations of the bus stops, and the red mark is for the location of the high school.

Once the bus stops were created, groups were created, and these groups are made up of the bus stop and the students; to these groups, the students were assigned with the condition that they cannot cover a distance greater than 500 meters of walking distance. In the following Fig. 4, it is possible to see a part of the created groups.

Once the groups were created, the students were assigned to their corresponding stop; in this part, it was considered in addition to the distance that has to be traveled by the student and the radius of the zone that the group covers. For a student to be assigned to a group, he has to comply with the condition of not moving more than 500 meters to reach the stop; if the radius of two bus stops is at intersection and the student is between that intersection, it is assigned to the group you are closest to. In Fig. 10, the groups created, and the students assigned to them are shown (Fig. 5).

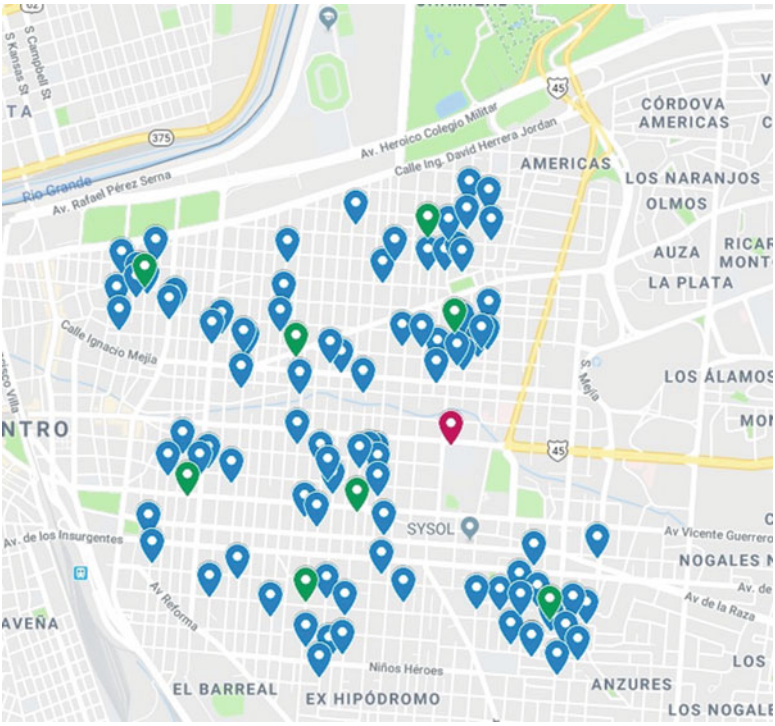

Fig. 3 Location of bus stops

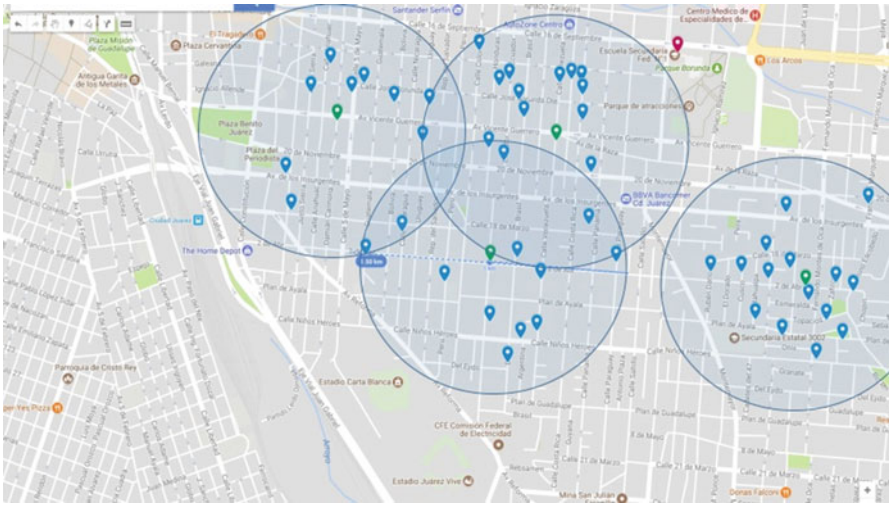

Fig. 4 Creation of the groups to be analyzed

Fig. 5 Students assigned to groups

2.4 Dijkstra's Algorithm

This algorithm arises in order to find the shortest distance from a node of origin to any other specified node and was created in 1959 by Edsger Wybe Dijkstra. Each node is assigned a weight; this weight is a value that represents how short the distance of that node is at the beginning. This algorithm is considered to belong to the algorithm $O(n^2)$ where n is the number of nodes belonging to the network, given that increasing the number of nodes usually increases the solution time [10]. It consists in broad strokes, having initialized the distances between nodes and grouping the nodes or vertices into three types, the nodes marked, adjacent, and distant. The initial node is a marked node, the adjacent nodes are nodes connected to a marked node, and the distant nodes are nodes that have not yet been taken into account. It must be taken into account that each adjacent node saves which node was accessed. The procedure is as follows: the initial node is a node marked as visited and with zero cost, the adjacent nodes are marked to the current node, and its cost is calculated from the current node; from this point, the algorithm will cycle until all the nodes are marked nodes; from the marked nodes, the lowest cost is taken and becomes the current node, and the nodes connected to the current node that were adjacent to another node become adjacent nodes of the current node if its cost from the current node is smaller than they already have [11].

2.5 Technological Framework

Since the data was stored, call the distance function, which receives the two lists of arrangements, and return a double-type matrix; in this function, the calculations of the distances between points were made, and the matrix that returns counts with the distances between said points. To calculate the distance between the points, two nested cycles and the following equation were used:

$$d = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2} \tag{1}$$

where the "*X*" is taken as the latitudes and the "*Y*" as the lengths; this is inside nested, to calculate the distance of a starting point toward all other stops; in this way, all the stops will be connected between them. When the distance is calculated with the same point, the diagonal of zero is generated, taking into account that the distance is the same in the inverse points, that is, the distance in the point (1,2) is the same distance in point (2.1), and it can be confirmed that the distance between the points is calculated correctly; in the following Table 1, the distances between points are shown.

2.5.1 vecinoC Function in Java

This function is where the calculation is performed to obtain the optimal route, and the function receives two double matrices which are the distance matrices; the reason why it receives twice the same matrix is due to the procedure that is performed to obtain the optimal route; when doing the search in the matrix, the values are being modified with 100.00. In this function, the near-neighbor algorithm is implemented, in which nested cycles are used; the first cycle is to have control of the initial point of each route and to look for different optimal paths from other points and to find a better route. The second and third cycle is where the search for the minimum value between each point of the matrix is made (Fig. 6).

Table 1 Distance matrix between points

1	0	1	2	3	4	5	6	7	8
0	0	0.004754	0.008854	0.016618	0.008528	0.013256	0.005438	0.009781	0.008884
1	0.004754	0	0.004237	0.015505	0.007914	0.01495	0.00897	0.013555	0.013046
2	0.008854	0.004237	0	0.014227	0.008267	0.016203	0.01211	0.016556	0.017278
3	0.016618	0.015505	0.014227	0	0.008102	0.009119	0.014198	0.015547	0.024567
4	0.008528	0.007914	0.008267	0.008102	0	0.007995	0.007189	0.010359	0.016796
5	0.013256	0.01495	0.016203	0.009119	0.007995	0	0.008442	0.00738	0.018732
6	0.005438	0.00897	0.01211	0.014198	0.007189	0.008442	0	0.004585	0.010616
7	0.009781	0.013555	0.016556	0.015547	0.010359	0.00738	0.004585	0	0.012128
8	0.008884	0.013046	0.017278	0.024567	0.016796	0.018732	0.010616	0.012128	0

Fig. 6 First cycles

```
for(int x=0; x<myList.length; x++)
{
   start=x;
   columna=x;
   arreglo[renglon][columna]=x;
    renglon++;
   ruta.clear();
   ruta.add(start);


   for(int i = 0; i <myList.length-1; i++)
       {

           for(int j = 0; j < myList.length; j++ )
           {

               if(min>myList[start][j])
               {
                       min=myList[start][j];
                       nodo=j;

               }

           }
```

Within the third cycle, there is a condition where the minimum value of the distance matrix was analyzed when finding it and the value of the node is taken to form the route. In the following image, the code of the cycles is shown. When leaving the third cycle, a sum was made with the distance between the nodes and another for where the matrix will be modified with the maximum value, to mark the points where the minimum value was found and discard them when analyzing the matrix again. The route was stored in an ArrayList; this is updated when leaving the cycle that modifies the matrix and when leaving the cycle for another condition in which the distances are compared, and the minor is conserved.

3 Development

In this section, the most outstanding aspects are presented during the development of the project; they are the diagram of the proposed solution, the methodology used, and the development process.

Table 2 Distance matrix with maximum diagonal

	0	1	2	3	4	5	6	7	8
0	100	0.004754	0.008854	0.016618	0.008528	0.013256	0.005438	0.009781	0.008884
1	0.004754	100	0.004237	0.015505	0.007914	0.01495	0.00897	0.013555	0.013046
2	0.008854	0.004237	100	0.014227	0.008267	0.016203	0.01211	0.016556	0.017278
3	0.016618	0.015505	0.014227	100	0.008102	0.009119	0.014198	0.015547	0.024567
4	0.008528	0.007914	0.008267	0.008102	100	0.007995	0.007189	0.010359	0.016796
5	0.013256	0.01495	0.016203	0.009119	0.007995	100	0.008442	0.00738	0.018732
6	0.005438	0.00897	0.01211	0.014198	0.007189	0.008442	100	0.004585	0.010616
7	0.009781	0.013555	0.016556	0.015547	0.010359	0.00738	0.004585	100	0.012128
8	0.008884	0.013046	0.017278	0.024567	0.016796	0.018732	0.010616	0.012128	100

3.1 Proposed Solution

3.1.1 Implementation of the Near-Neighbor Algorithm

As mentioned, this algorithm was implemented in the neighborC function; when this algorithm was performed in Java, the optimal route for the SBRP solution was obtained, and the implementation was carried out using the distance matrix looking for the minimum between the points. The algorithm is completely adaptable for the number of stops that are necessary; this is because the routes are stored in a matrix that takes the size of its lines and columns of the size of the ArrayList.

When calculating the distance of the equal points of the matrix, for example, m [0] [0] will be placed with a double greater than all the distances shown in Table 2; this modification was made in the distance function in order to discard these points as minimum since if they remain as zero they would be taken as small values. The following table shows how it was formed.

In this way, it is ensured that the minimum real values are taken, having the minimum of each point; two situations necessary for a good result are processed, and both situations happen when having the minimum value; one is when finding said value, and that value is substituted in the matrix by the greater value; this would mark the point as visited, and it would be discarded that that point be contemplated again; the next situation is to save the point of the visited matrix in an ArrayList which would be forming the optimal route.

3.2 Construction of the Prototype

The database has five tables, in which the information contained is necessary to resolve the SBRP. The tables are the following:

- Depósito_autobuses: This table has only two fields, since it is designed to have better control of the buses that count and its capacity; the fields are id_autobus;

this field is the field to identify buses; and the field ability is to store the amount of students who can ride the bus.

- Parada_autobus: This table is designed to store the location of the bus stops; this is the most important table in the design since from here the data is taken to perform distance calculations in the Java library: in the field id_stop, this field is to identify the bus stop; in the field id_autobus, the bus that would pass through that stop is associated; in the field students_paid, the number of studies that count by stop is stored; in the field lat_bs, this field saves the latitude of the bus stop; and in the field lon_bs, the length of the bus stop is stored.
- Detail_parado: This table is designed to have the relationship between student and stop, to have a better detail of it, and to have a control that the student corresponds to the bus stop; the field id_stop is the identifier of the stop table of bus, and just like the field id_stop, the field student_id is the identifier of the student table.
- School: This table has the basic data of the school, including the location; in the fields, that account is school_id, which is the one with which the school is identified, and the field name, latitude, and longitude coordinates are stored in the lat_esc and lon_esc.
- Student: In this table, the basic data of the students will be stored, such as name, paternal surname, maternal surname in their corresponding fields, and the coordinates for the location in the lat_est and lon_est fields (Fig. 7).

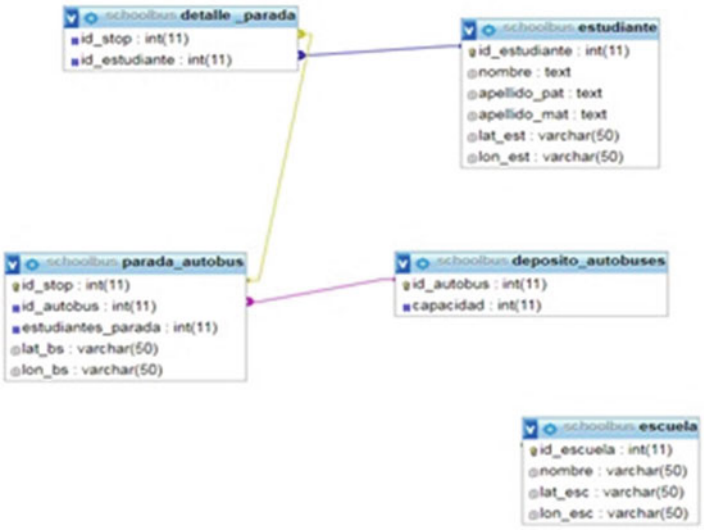

Fig. 7 Database relationship

```
public class conectar {
    private static Connection conn;
    private static final String driver = "com.mysql.jdbc.Driver"; //driver mysql-connector
    private static final String user="root";                      //Usuario de la bd
    private static final String password="";                      // psswd de la bd
    private static final String url="jdbc:mysql://localhost:3306/schoolbus";  //direccion de la bd
```

Fig. 8 Class connect

3.2.1 Data Connection and Management

For a better implementation of the code, a connection was made between the Java library and the database; with Java, this was possible by integrating a library called mysql-connector-java; in the project, version 5.1.46 was used. In this way, the connection can be handled as a class where the data of the database, such as the user, password, and location of the database, are passed as variables (Fig. 8).

Just as the class performs a function to call it from any other function within the program, taking advantage of that, it was sent to call from another class where the information obtained is processed, which are the coordinates of the bus stops. Being connected to the Java database, you can execute SQL query and extract the data from the stop_autobus table, since the information in the table could be processed in an array with the distances between the points.

3.3 Validation

A first route was obtained which starts at stop one, followed by two, four, six, seven, five, three, zero, and eight ending at the initial stop (1, 2, 4, 6, 7, 5, 3, 0, 8) which I throw a total distance of 7.9 kilometers. Each number on the route represents a bus stop, where zero was assigned to the school.

In Fig. 15, it shows how the route would be when following the route (Fig. 9).

4 Results

By adding the external cycle to the algorithm and performing the tests with the library, a new route was obtained for the graph, which would start at stop number four, following stops six, seven, five, three, two, one, zero, and eight, ending with the initial stop (4, 6, 7, 5, 3, 2, 1, 0, 8); the total distance of this route is 7.7 kilometers. It is possible to have a better route, since, when starting from a different point, the minimum distances to another bus stop change, or they are not discarded because it is a stop already visited. In Fig. 10, the route is shown following the route that was obtained.

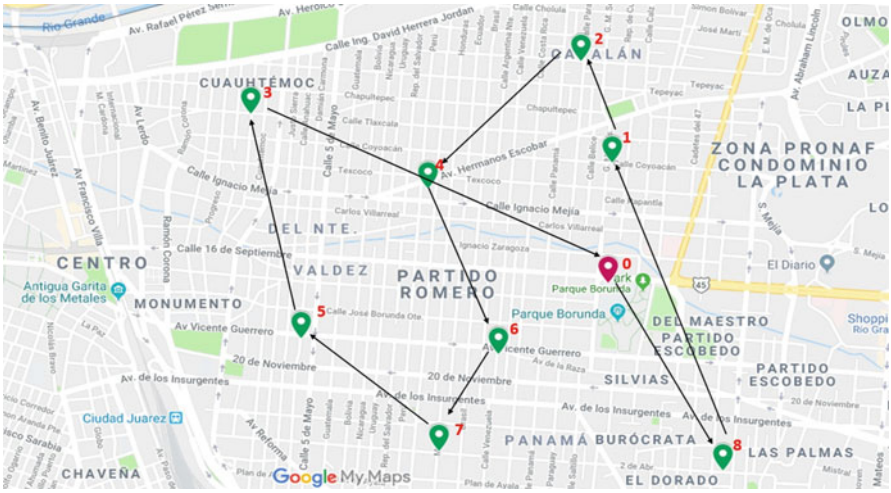

Fig. 9 First route obtained

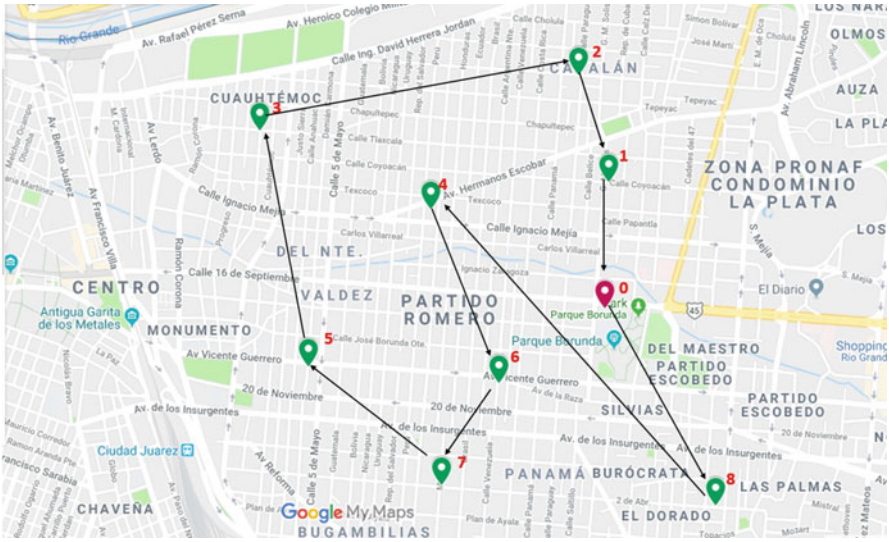

Fig. 10 Optimal route

The route is formed in such a way that the next stop number is the combination with the previous stop number; in this way, it is possible to know which is the point of departure and departure, in order to perform the sum of the distance and corroborate which is the optimal route. An important aspect is to solve with less time the resolution of a new neighborhood with student locations and determine the correct locations of each stop of this route, as you can see in Fig. 11.

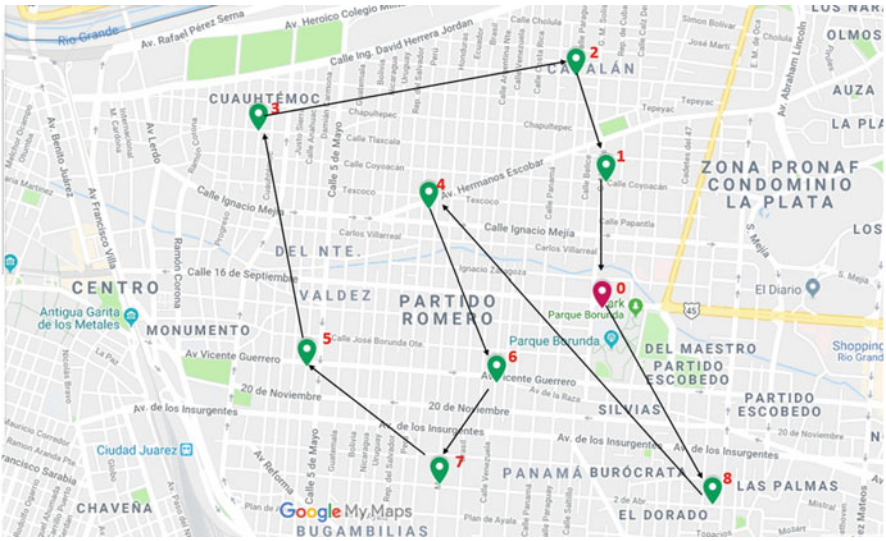

Fig. 11 Description of each group of students and their associated stops (**a**) and then the execution of waiting times (**b**)

5 Conclusions and Future Research

5.1 Regarding the Objective

In this section, we will analyze the results obtained by the library, where the route obtained will be shown, the distance between bus stops which is linear. It will show part of the code that was developed for the operation of the library and the realization of a test to confirm that the library is adaptable for any number of bus stops; it will also show the way in which the routes are saved and the total distances of them to verify that the route thrown by the library is optimal.

5.2 Testing Our Java Library Proposed

When using the library, this resulted in a first route found in the graph, which was taken as the optimum at that time; the code shown in the following figure is with which I throw this first route. In Table 3, the route with the distances of each point is shown; the minimum distance is from stop one, to stop two, with 42 meters, with the longest distance at stop three, to stop zero with 160 meters, as you can see in Fig. 12.

After some modifications to the code were made, which are shown in Fig. 13, a new route with a shorter distance was obtained; in Table 4, the improvement in the

Table 3 First route

Stop bus	Distance
[1][2]	0.00423728651379396
[2][4]	0.00826666232526575
[4][6]	0.00718901397967449
[6][7]	0.00458543400345120
[7][5]	0.00738010840028139
[5][3]	0.00911904868942050
[3][0]	0.01661762513116540
[0][8]	0.00888440712710274
[1, 8]	0.01304609811399880
Dist. Total	0.07932568428415420

Fig. 12 Code before modifying

```
min=myList[0][0];

for(int i = 0; i <myList.length-1; i++)
    {

        for(int j = 0; j < myList.length; j++ )
        {

            if(min>myList[start][j])
            {
                    min=myList[start][j];
                    nodo=j;

            }

        }

        dist= dist+min;
        min= 100.0000;

             for(int k=0; k < ruta.size();k++)
            {

                myList[nodo][(int)ruta.get(k)]=100.0000;
                myList[(int)ruta.get(k)][nodo]=100.0000;
```

route is shown, and unlike Table 3, it is possible to appreciate the decrease in total distance, and in this route, the minimum distance is from stop two, to stop one with 42 meters, with the greater distance at stop eight, to stop four with 167 meters. The minimum distances in the two tables are the same; the difference is the direction in which the points are traversed for each route.

The routes that are obtained by the library are stored in an array, in which they are saved by columns where the first number of the route is the number of the column to which it belongs. The following Table 5 shows the routes thrown by the library.

To corroborate the route that the library shows as the optimum, the distances of the other routes were added, the distances are shown in the following table where

Fig. 13 Modified code

```
for(int x=0; x<myList.length; x++)
{
   start=x;
   columna=x;
   arreglo[renglon][columna]=x;
    renglon++;
   ruta.clear();
   ruta.add(start);


   for(int i = 0; i <myList.length-1; i++)
       {

           for(int j = 0; j < myList.length; j++ )
           {

               if(min>myList[start][j])
               {
                       min=myList[start][j];
                       nodo=j;

               }

           }

           dist= dist+min;
           min= 100.0000;
```

Table 4 Optimal route

0	1	2	3	4	5	6	7	8
7.8	7.9	8.4	9.1	7.7	8.6	8.8	8.1	7.8

Table 5 Route arrangement

0	0	1	2	3	4	5	6	7	8
1	1	2	1	4	6	7	7	6	0
2	2	4	0	6	7	6	5	0	1
3	4	6	6	7	5	0	4	1	2
4	6	7	7	5	3	1	1	2	4
5	7	5	5	0	2	2	2	4	6
6	5	3	4	1	1	4	0	5	7
7	3	0	3	2	0	3	8	3	5
8	8	8	8	8	8	8	3	8	3

the distances are represented in kilometers, and the column corresponds to each one of the routes. In the Table 6, you can see that route four is effectively the shortest distance with 7.7 kilometers, and the worst route is the three with 9.1 kilometers.

Table 6 Total, distances by route

	0	1	2	3	4	5	6	7	8	9	10
0	100	0.004754	0.008854	0.016618	0.008528	0.013256	0.005438	0.009781	0.008884	0.00502	0.004781247
1	0.004754	100	0.004237	0.015505	0.007914	0.01495	0.00897	0.013555	0.013046	0.003581	0.000592864
2	0.008854	0.004237	100	0.014227	0.008267	0.016203	0.01211	0.016556	0.017278	0.00702	0.004484459
3	0.016618	0.015505	0.014227	100	0.008102	0.009119	0.014198	0.015547	0.024567	0.019018	0.016096238
4	0.008528	0.007914	0.008267	0.008102	100	0.007995	0.007189	0.010359	0.016796	0.011221	0.008497105
5	0.013256	0.01495	0.016203	0.009119	0.007995	100	0.008442	0.00738	0.018732	0.017582	0.015462907
6	0.005438	0.00897	0.01211	0.014198	0.007189	0.008442	100	0.004585	0.010616	0.010381	0.009276332
7	0.009781	0.013555	0.016556	0.015547	0.010359	0.00738	0.004585	100	0.012128	0.014798	0.01385946
8	0.008884	0.013046	0.017278	0.024567	0.016796	0.018732	0.010616	0.012128	100	0.011041	0.01281321
9	0.00502	0.003581	0.00702	0.019018	0.011221	0.017582	0.010381	0.014798	0.011041	100	0.003017095
10	0.004781	0.000593	0.004484	0.016096	0.008497	0.015463	0.009276	0.013859	0.012813	0.003017	100

Table 7 Distance with new bus stops

Stop bus	Distance
[4][6]	0.00718901397967449
[6][7]	0.00458543400345120
[5][7]	0.00738010840028139
[3][5]	0.00911904868942050
[2][3]	0.01422713077187880
[1][2]	0.00423728651379396
[0][1]	0.00475429595208644
[0][1]	0.00888440712710274
[4][8]	0.01679555968106130
Dist. Total	0.07717228511875080

5.3 Test with More Bus Stops

A test was carried out with more bus stops to verify that the library is fully adaptable; for the test, two stops were added randomly. Table 7 shows the distances between points and the maximum diagonal obtained with the new number of stops. The route that the library launched with these new routes is 9, 10, 1, 2, 4, 6, 7, 5, 3, 0, and 8. This new route has a total distance of 8.0 kilometers, with the minimum distance in the travel from stop ten to one with a distance of 500 meters. The following Table 8 shows the route and the distance between each point with the total distance.

When obtaining these results, it can be said that the test was satisfactory and that the library is adaptable for different numbers of stops. Besides that, the user can use the library in a simple way; when added to the project, he is using and importing the class coordinates from the library; a coordinate-type object is created in which it will serve to call the functions that are getDatos and vecinoC; the first function that has to be called is getData and creates two double-type matrices for it, because this function is the one that returns the distance matrix that will be sent to the neighborC function in which an ArrayList is used. This is because the function receives the two matrices, and the ArrayList has the optimal route that is returned by the function. In the following image, the code is shown (Fig. 14).

Table 8 Route with new distances

Bus stops	Distance
[9][10]	0.003017
[1][10]	0.000593
[1][2]	0.004237
[2][4]	0.008267
[4][6]	0.007189
[6, 7]	0.004585
[5, 7]	0.00738
[3, 5]	0.009119
[0, 3]	0.016618
[0, 8]	0.008884
[8, 9]	0.011041
Dist. Total	0.080931

Fig. 14 Call of functions

```
Coordenadas gfo = new Coordenadas();
double [][] x;
double [][] x1;
ArrayList ruta=new ArrayList();
x=gfo.getDatos();
x1=gfo.getDatos();

 ruta=gfo.vecinoC(x,x1);
```

6 Conclusions

In this research, we analyze the objective of this project, which is to develop a library programmed in Java and a minimum scheme in a relational database, which is publicly accessible for the development of an application, designed to solve the school bus routing problem; with the library, you will have a tool for the implementation and solution of the SBRP.

6.1 With Respect to the Main Objective of Our Investigation

When the library was created in the beginning, the Dijkstra algorithm was contemplated. By having the library finished with said algorithm, the expected results were not obtained, since this algorithm searches for the shortest path from an initial point to an endpoint, without going through all the points; for the user, it would be a bit more complicated to perform the optimization, since he would have to add all the adjacent points he needs with their respective distances. In this case of SBRP

having nine bus stops, the route that Dijkstra throws consists of four stops with two different final destinations, one with the final destination five, resulting in stops one, two, three, and five; the other route is with the final destination six, which gives stops one, nine, seven, and six, and both routes start at stop one. When making the library with the algorithm of near neighbor, a route was obtained in which all the points are crossed, and it is easier to use by the user; since the library makes all the necessary calculations, it would only have to fill the tables in the database for the proper functioning of the library. When performing the corresponding tests and verifying the result that the library showed as the optimum, it was possible to conclude that the objective of the project was met, since there is a library that can solve the SBRP and throws the optimal path as result.

6.2 Recommendation in the Future Application of Our Research

As recommendations for this project, you can implement some tool or algorithm, to take the flow of traffic from the streets and thus stop being linear and to have a route closer to reality; another of the things that could be improved is the call of the algorithm of near neighbor, not to have to send the two matrices of distances and not to occupy that space in memory.

6.3 Future Research

Being a project that improves a route, not only can it be used for schools, but also it can be used for parcel companies, delivery companies, or transport companies. Using a mobile device associated with a more detailed framework could even determine specific days with more transported objects associated with each student, as can be seen in Fig. 15.

An aspect of utmost importance will be to adapt in time, the specifications of each individual in childhood that allows it to be competitive, helping it with mobility in a smart city, as can be seen in Fig. 16.

The route is formed in such a way that the next stop number is the combination with the previous stop number; in this way, it is possible to know which is the point of departure and departure, in order to perform the sum of the distance and corroborate which is the optimal route. An important aspect is to solve with less time the resolution of a new neighborhood with student locations and determine the correct locations of each stop of this route, as you can see in Fig. 17.

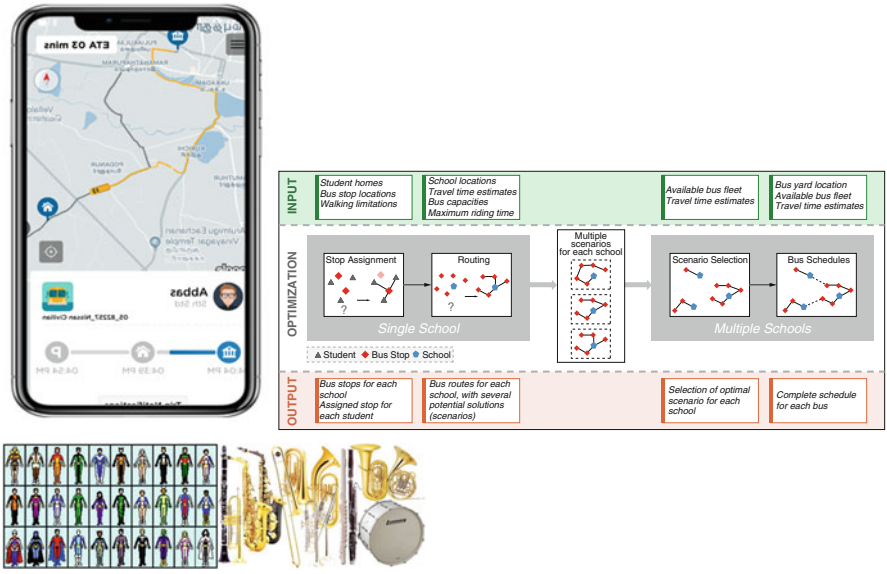

Fig. 15 Intelligent design of a mobile application based on Ubicuro Computing that would allow each student to send information associated with the transport of musical objects of a band and online specifications of each particular situation and of the dimensions and care of their respective musical instruments

Fig. 16 The OECD considers that by 2027, seven of the 12 components of a smart city will be associated with improving school performance, including student mobility

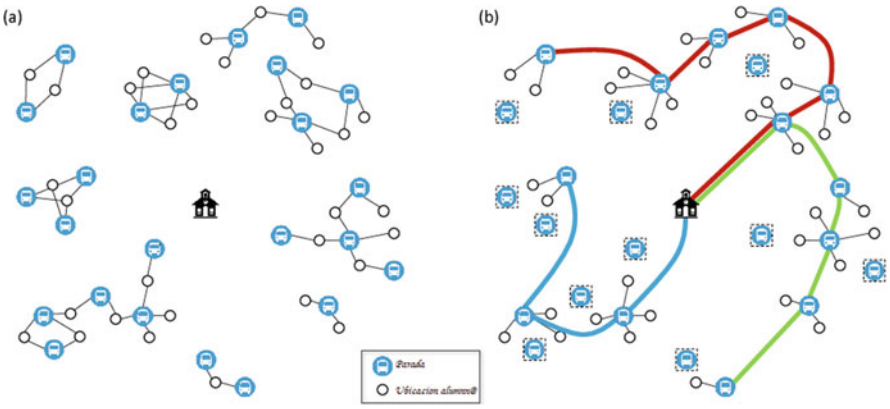

Fig. 17 Description of each group of students and their associated stops (**a**) and then the execution of waiting times (**b**)

References

1. Kang, M., Kim, S., Felan, J., Choi, H., & Cho, M. (2015). Development of a genetic algorithm for the school bus routing problem. *International Journal of Software Engineering and Its Applications, 9*(5), 107–126. Available: https://doi.org/10.14257/ijseia.2015.9.5.11.
2. Andrade, T., de Arruda Pereira, M., & Wanner, E. F. *Development of an application using a clustering algorithm for definition of collective transportation routes and times*,» DECOM – Centro Federal de Educação Tecnológica de Minas Gerais – Campus II – Brazil.
3. Jiménez, J. F., Arias-Rojas, J. S., & Montoya Torres, J. R. (2012). *Solving of school bus routing problem by ant colony optimization*, *Revista EIA*. [En línea]. Disponible: https://goo.gl/t2sWUT. Accedido: 17-Sep-2017.
4. Bekas, S. E. T. (2007). *Solving school bus routing problem through integer programming*. *The Journal of the Operational Research Society, 58*(*12*), 1599–1604. Palgrave Macmillan Journals.
5. Bögl, M., Doerner, K. F., & Parragh, S. N. (2015). *The school bus routing and scheduling problem with transfers*. [En línea]. Disponible: https://goo.gl/rQ9tcJ. Accedido: 23-Sep-2017.
6. Yingit, T., & Unsal, O. (2016). *Using the ant colony algorithm for real-time*. *The International Arab Journal of Information Technology, 13*(5). [En línea]. Disponible: https://goo.gl/z2xfvM. Accedido: 23-Sep-2017.
7. De Souza Lima, F. M. (2015). *A mixed load rural school bus routing problem whit heterogeneous fleet: A study for the brazilian problem*, tesis, Universidade Federal de Minas Gerais. [En línea]. Disponible en: https://goo.gl/DGdiJj. Accedido: 27-Sep-2017.
8. de Guevara, J. M. L. *Fundamentos de programación en Java*, Editorial EME. [En línea]. Disponible: https://goo.gl/tWUr2D. Accedido: 26-oct-2017.
9. Groussard, T.. (2012). *JAVA 7: Los fundamentos del lenguaje Java*, Ediciones ENI. [En línea]. Disponible: https://goo.gl/ak7yfv. Accedido: 26-Oct-2017.
10. Quintana, G., Marqués, M., Aliaga, J. I., & Aramburu, M. J. (2008). *Aprende SQL*, Publicacions de la Universitat Jaume I. [En línea]. Disponible: https://www.e-buc.com/portades/9788480217729_L33_23.pdf. Accedido: 26-Oct-2017.

11. Wilton, P., & Colby, J. (2005). *Beginning SQL* (p. 12). Indianapolis: Wiley.
12. Mehlhorn, K., Orlin, J., & Tarjan, R. (1987). *Faster algorithms for the shortest path problem*. Technical report CS-TR-154-88, Department of Computer Science, Princeton University. [En línea]. Disponible: ftp://ftp.cs.princeton.edu/reports/1988/154.pdf. Accedido: 26-Oct-2017.
13. Park, J., & Kim, B. I. (2010). The school bus routing problem: A review. *European Journal of Operational Research, 202*(2), 311–319. [En línea]. Disponible: https://bit.ly/2PrrQae. Accedido: 29-Oct-2017.
14. Gutin, G., Yeo, A., & Zverovich, A. (2002). Traveling salesman should not be greedy: Domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics, 117*(1–3), 81–86. [En línea]. Disponible: https://bit.ly/2C1iRno. Accedido: 02-Nov-2018.
15. Cunquero, R. M. (2003). *Algoritmos heurísticos en optimización combinatoria*. Universidad de Valencia, Facultad de Ciencias Matemáticas. [En línea]. Disponible: https://bit.ly/2TxPHTT. Accedido: 14-Dic-2018.
16. Torrubia, G., & Terrazas, V. (2012). *Algoritmo de Dijkstra. Un tutorial interactivo*. VII Jornadas de Enseñanza Universitaria de la Informática (JENUI 2001). [En línea]. Disponible: https://bit.ly/2PqhDrI. Accedido: 14-Dic-2018.
17. Sniedovich, M. (2006). Dijkstra's algorithm revisited: the dynamic programming connexion. *Control and Cybernetics, 35*((3), 599–620. [En línea]. Disponible: https://bit.ly/2JbVIEb. Accedido: 14-Abr-2019.