A Multi-objective Mathematical Optimization Model and a Mobile Application for Finding Best Pedestrian Routes Considering SARS-CoV-2 Contagions

Juan E. Cantor(✉), Carlos Lozano-Garzón, and Germán A. Montoya

Systems and Computing Engineering Department, Universidad de Los Andes, Bogotá, Colombia
{je.cantor,calozanog,ga.montoya44}@uniandes.edu.co

Abstract. Given the spread and pandemic generated by the SARS-CoV-2 coronavirus and the requirements of the Universidad de Los Andes in terms of guaranteeing the health and safety of the students considering possible contagions at the university campus, we propose and design a mobile application to obtain the best route between two places in the campus university for a student. The path obtained by our proposal, reduces the distance traveled by the student as well as a possible contagion of the coronavirus during his journey through the university campus. In this sense, two types of costs were modeled: one cost represents the distance cost to travel the campus by a student, and the second one represents the contagion susceptibility that a student has when he is passing through the campus. In summary, it was developed and validated a solution algorithm that minimizes these two types of costs. The results of our algorithm are compared against a Multi-Objective mathematical optimization solution and interesting findings were found. Finally, a mobile application was created and designed in order to obtain optimal routes to travel between two points in the university campus.

Keywords: SARS-CoV-2 · Pedestrian traffic · Multi-objective optimization · Heuristic · Mobile application

1 Introduction

The pandemic spread by the SARS-CoV-2 virus, commonly called COVID-19, led multiple countries worldwide to take and enforce social restrictions in order to limit the spread of the virus. The most common contingency measures, taken by several governments, is to limit the movements of its citizens, as well as the social interactions that occur between them. This measure is known as "*social distancing*" and has the purpose of minimizing the exchange of segregations

H. Florez and M. F. Pollo-Cattaneo (Eds.): ICAI 2021, CCIS 1455, pp. 191–206, 2021.
https://doi.org/10.1007/978-3-030-89654-6_14

between one person to another, which is the main event that causes the exponential growth of the contagion of the coronavirus [7].

However, in many scenarios, there are a large number of people spending daily hours walking into multiple pedestrian congestions, where the established social distancing is not accomplished. In 2019, Universidad de Los Andes reported around 14 thousand undergraduate students, 3.000 master's students, and about 364 doctoral students. [22] in a campus area that covers around 11.4 ha [21]. In this sense, this institution has a large volume of pedestrians circulating daily through the different facilities of the campus (multiple corridors and interconnection building routes). This scenario was the common behavior before the implementation of virtual classes by the institution due to the pandemic. Therefore, due to the need to fulfill the social distancing within the campus considering the possible agglomerations, we propose a mathematical model for minimizing the person exposure in places with a risk of contagion (places with high traffic of pedestrians at a specific time). In addition, the mathematical model is proposed to be accessed through a mobile application with the aim of a pedestrian user can determine a route to traverse the campus minimizing the risk of contagion at a specified time.

The remainder of the paper is organized as follows: Sect. 2 presents related works, and the identification and relevance of the problem to be treated. Section 3 shows the descriptions and justifications of the multi-objective mathematical optimization model design process developed for the implementation of the mobile application. Section 4 describes the understanding of the problem modeled in the phases presented in Sect. 3. It also describes the elaboration and validation of the algorithm solution. Section 5 presents the design and development of the mobile application that corresponds to the front-end solution of the proposal. Section 6 analyzes and discusses the findings obtained by our proposal.

2 Related Work

The same COVID-19 susceptibility probability equation for an individual in pedestrian congestions was used through these related works. Likewise, these studies presented simulations to represent the population behavior under different types of events in a casual day. In other words, these studies emphasized how displacement or mobility events are related to the spread of the SARS-CoV-2 virus.

In [6], researchers performed simulations of human mobility models to study the dynamics of coronavirus infection in order to get a dynamic model of the virus spread. To capture the nature of the epidemiological dynamics, the study made use of the *SIR* (Susceptible, Infected and Recovered) model. Through contagion rates planned in the model, the behavior of different groups populations was simulated. If a susceptible person and an infected person meet, there is a probability that the susceptible person will become infected. Time after infection, the person typically recovers. Therefore, the study introduced a contact graph constructed from everyday situations such as agglomerations in public

trans-port, work, home, and others (scenarios in which the *SIR* model could be applied and studied). Finally, after executing the simulations of the contact graph with the epidemiological dynamics previously stated, the study set out strategies as conclusions, such as the suspension of public transport, total quarantine, and others social decisions, in order to prevent and reduce the spread of the virus.

For instance, one of the researched works created an epidemiological model based on agents (called *PanCitySim*) in which the movement of many individuals were modeled in a city. For this purpose, the model used a graph to represent the contact of many activities performed by the population in the whole city. In this sense, the spatio-temporal properties of an epidemic were studied according to the *SIR* model in order to represent the impact of human interactions to the transmission of the virus [5].

Therefore, it was suitable to study human mobility models in cities in order to understand the spreading behavior of COVID-19. Based on these related works, in our proposal, a solution could apply to the mobility patterns associated with pedestrians and their environment within the Universidad de Los Andes, as well as mathematical models and probabilities associated with the spreading of the virus.

3 Multi-objective Mathematical Optimization Model

In this research we propose a mathematical model for minimizing the distance traveled by a student and the risk of contagion when he is walking through campus. Therefore, the model included a transportation cost matrix represented as the distances between each point or node that composes the route of the map to travel. Likewise, it was also established a matrix of interaction costs that represents the pedestrian traffic through the points or nodes in the map. In addition, in the model we defined a minimum coverage radius for each node, which is the maximum distance that must be for each pair of nodes to be a connection.

In order to define the initial nodes network, the *GeoJSON* web tool was used [12], where nodes were placed consecutively on each path of the university campus. Once the coordinates of the different nodes have been exported in a file in *JSON* format, they were loaded as parameters to the mathematical model. To understand the mathematical model, sets, variables, and parameters are presented in Table 1.

Thus, in order to verify the solution of the initial model, the objective function was defined:

Minimize

$$\sum_{i \epsilon N} \sum_{j \epsilon N} X_{ij} \cdot (Ct_{ij} + Ci_{ij}) \tag{3.0.1}$$

Subject to:

$$\sum_{j \epsilon N} X_{ij} = 1 \qquad \forall i \, \epsilon \, N | i = S \tag{3.0.2}$$

$$\sum_{j \epsilon N} X_{ij} = 0 \qquad \forall i \, \epsilon \, N | i = S \tag{3.0.3}$$

$$\sum_{i \epsilon N} X_{ij} = 1 \qquad \forall j \, \epsilon \, N | i = D \tag{3.0.4}$$

$$\sum_{i \epsilon N} X_{ij} = 0 \qquad \forall j \, \epsilon \, N | i = D \tag{3.0.5}$$

$$\sum_{j \epsilon N} X_{ij} - \sum_{j \epsilon N} X_{ji} = 0 \qquad \forall i \, \epsilon \, N | i \neq D \wedge i \neq S \tag{3.0.6}$$

$$X_{ij} \cdot Ct_{ij} \leq RC \qquad \forall i, j \, \epsilon \, N \tag{3.0.7}$$

Table 1. Sets, parameters and variables of the initial mathematical model

Category	Symbol	Description
Sets	N	Number of nodes in the network
Parameters	Ci_{ij}	Interaction cost between node i and node j, which is the number of people that walk through the link (i,j)
	Ct_{ij}	Transport cost between node i and node j, which is the euclidean distance between these nodes
	S	Source node number, in which a student started its walk
	D	Destination node number, in which a student finished its walk
	RC	Node's Ratio Coverage, which is the maximum distance that two pairs of nodes must have to exist a connection link between them
Variables	X_{ij}	Binary decision variable that shows whether the student chooses the link between node i and node j as a part of its path from the source node to the destination node. 0 if otherwise

The objective function of the mathematical model is subjected to the constraints represented in Eq. 3.0.2 to Eq. 3.0.6. In Eq. 3.0.2 it is indicated that the source node should only have one outflow, that is, the student can only take one path from its point of origin. Equation 3.0.3 ensures that the student cannot be returned to its source node. Equation 3.0.4 ensures that the student reaches its destination node. Equation 3.0.5 establishes that once the student reaches the destination node, he cannot travel to other paths or nodes. For intermediate nodes between the source node and the destination node, Eq. 3.0.6 establishes that if a student reaches an intermediate node, they must leave that node. Finally, to guarantee that the model chose links within the ratio coverage of each node, Eq. 3.0.7 was proposed.

In terms of the mathematical model implementation, we executed the model on the Pyomo optimization framework on an Intel ®Core i5-6200U, 2.3 GHz, 7.7 GiB system.

3.1 Interaction Cost Function

The susceptibility probability of contagion proposed by [6] was used in our proposal, which can be observed in Eq. 3.0.7.

$$P_{n,t} = 1 - e^{-\theta \cdot \sum_{m \epsilon M} q_{m,t} \cdot i_{nm,t} \cdot t_{nm,t}} \tag{3.1.1}$$

Where M is the set of people with whom a user was in close contact, with the following variables and parameters: $P_{n,t}$ is the Probability of person n to receive an infectious dose. However, as stated in [8], this probability should not be understood as "probability of contracting the virus", since this depends on the immune system of each person. θ is the Calibration factor for a particular disease. $q_{nm,t}$ corresponds to the Microbial load between two people, which is the amount of virus that the person m transmits in time t. $i_{nm,t}$ is the Contact intensity between the person n and m, which corresponds to the distance. Finally, $t_{nm,t}$: Time that person n interacts with person m for a time t.

Now, for the simplicity of the proposed model, we assumed that the distance between each pedestrian and each user corresponds to 1 m and that their interaction time was 0.5 s. However, due to the fact that as of the date this project was developed, no calibration parameters and microbial load of COVID-19 were known. Thus, parameters estimated by the previously mentioned studies were used.

First, the value $q_{m,t}$ was assigned as 1 based on estimates made by [6]. Similarly, the calibration parameter θ was taken as $\frac{1}{20}$ based on studies conducted by [8] in closed spaces such as aircraft interiors during commercial flights.

We considered that the choice of this calibration value should be made on the assumption that all routes in the campus are in a closed space (which is the assumption of a pessimistic scenario). With this decision, it was possible to establish an equal load of micro-organisms in the environment for all the connections between nodes, and therefore, this allowed the model to calculate the best path based on the segregations caused by other students and not on the dispersion caused by other factors.

3.2 Pedestrian Traffic Simulation on the University Campus

In order to establish parameters related to the interaction cost of the model in a post COVID-19 scenario, we processed the dataset of the anonymized schedules of undergraduate students in the first semester of 2019, as well as the semester current academic schedule. The data was pre-processed and cleaned through the data analysis software *KNIME* [17]. Using this tool, the courses list was matched to the courses registered by the students. The resulting data set had the following attributes with 65, 536 records: student ID, course ID, department or faculty of the course, name of the course, current academic cycle of the course, building where the course was taught, time and minutes when the course started, time and minutes when the course ended and days on which the course was taught.

Once the final set of data to be used as parameters in the model was got, we loaded the students pedestrian traffic data into a non-relational database [10] using executable scripts [19].

Then, each record of the data set was uploaded as the origin and destination buildings that each student has in a change of classes in an academic cycle, day and specific time.

For a student's first class of the day, we decided in our simulation that a student would arrive at the university campus through the transport nodes identified in *Uniandina Mobility Survey (Encuesta de Movilidad Uniandina)* [23], where each means of transport has an associated percentage, which represents the percentage that uses said means of transport. Then, a random variable was defined that oscillated within the ranges of percentages and that from the value that said variable took, it was assigned within the percentage of the closest mean of transport. Likewise, we defined the nodes in the network equivalent to where the transportation stops are within the university campus.

For non-consecutive classes belonging to the same day, we decided to define random nodes where students would spend their free hours. The choice of these nodes was made arbitrarily.

Once the data had been uploaded to the database, a hierarchical non-relational database was got based on the academic year, day, and class change schedule.

To model the route pattern that each student was going to use when moving around the campus, we decided that for the simulation the students would look for the shortest way to go from their source building to their destination building. For this task, we assumed that each student would move through each node using the Dijkstra Algorithm for minimal cost path (taking the distance between each pair of nodes as cost) [1].

Consequently, the selection of each path by each student assigned the interaction costs of the mathematical model and the solution heuristic. At Table 2 is the terminology used for Algorithm 1, which was used to map the pedestrian traffic parameters to the interaction cost variable:

Table 2. Related terminology used for the Algorithm 1

Symbol	Description	Assumptions
N	Set of network nodes	
E	Set of students that displace in a specific academic cycle, day and schedule	
$S(i)$	Student source node i	$\forall i \epsilon E \quad \forall S(i) \epsilon N$
$D(i)$	Student destiny node i	$\forall i \epsilon E \quad \forall D(i) \epsilon N$
$Sp(i)$	Optimal set of links that a student i has to take to displace from $S(i)$ to $D(i)$	$\forall i \epsilon E \quad \forall S(i), D(i) \epsilon N$
D_{ij}	Optimal links calculated by Dijkstra Algorithm from node i to node j displacement	$\forall i, j \epsilon N$

Algorithm 1: Algorithm for the simulation of pedestrian traffic and interaction costs assignment to the network nodes' links

```
C_ij = 0        ∀i, j ϵ N;
Sp(e) = {}      ∀e ϵ E;
for e ϵ E do
    Sp(e) = Dijkstra(S(e), D(e));
    for i, j ϵ N do
        if i, j ϵ Sp(e) then
            C_ij ← C_ij + 1;
        end
    end
end
```

3.3 Pareto Front Development

Consecutively, we implement a Pareto Front in order to observe the behavior of the transport cost and the interaction cost functions when they are minimized and one function has more importance or weight over the other.

Therefore, the Pareto Front was calculated through the method ε *-constraint* [2]. For this computation, the previously defined network of nodes was used and the parameters of the transport function were assigned based on the traffic loaded in the database.

Finally, we defined a weights vector in which each vector' element multiply each function on each iteration, in order to give greater weight to one function compared to the other. The mathematical expression was defined:

$$W = [10, 20, \ldots, 50]$$

where for each element of W would represent the upper limit that f_2 should have when wanting to minimize f_2 (this is reflected in the additional restriction described below). Therefore, each element W was iterated to obtain values close to the limit of f_2. It should be noted that the mathematical model of the method also includes the restrictions initially set in the Sect. 3.

The Pareto Front run was performed on an Intel ®Core i5-6200U, 2.3 GHz, 7.7 GiB.

for each $w \epsilon W$:

$$f_1 = \sum_{i \epsilon N} \sum_{j \epsilon N} X_{ij} \cdot Cij$$

$$f_2 = \sum_{i \epsilon N} \sum_{j \epsilon N} X_{ij} \cdot Cij$$

Minimize

$$f_1$$

Subject to:

$$f_2 \leq w \tag{3.3.1}$$

See Eq. 3.0.2–Eq. 3.0.7

4 Heuristic

As this problem can be represented as a minimum cost problem, where for each link there are two types of cost - transport cost and interaction cost respectively - we used known heuristic to solve the minimum cost problem by combining in a same scale the two types of cost for each link. Therefore, the Dijkstra Algorithm [1] was used as a solution algorithm, where the user, through the front-end of the project, could choose the criterion to minimize given their preferences and input parameters.

Regarding the unification of costs, the costs of each link were normalized, and then multiplied by a weight represented by a scalar between 1 and 0. Finally, the product of the normalized costs and their scalar was added, resulting in a single cost matrix. It should be noted that the scalar weight is the representation of the path criterion selected by the user (*i.e.* 0 as least importance, 1 as greatest importance). However, we found that given the modeling of pedestrian traffic described in the Subsect. 3.2 and the obtained Pareto Front results in different scenarios, the problem of minimization of the transport cost function and the interaction cost function only had solutions in the limits of each function, which is a solution was not found that minimized the two objectives "as equal", but two optimal solutions were presented that minimize only one function.

4.1 Bi-Objective Problem Understanding

To understand the reason for the existence of only two optimal solutions for the minimization problem of f_1 and f_2 (as conventions, f_1 is referred as the transportation cost function and f_2 is referred as the interaction cost function.), it must be remembered how the distribution of traffic was defined in the Subsect. 3.2, which was done under the assumption that every student who moved around the campus would choose the shortest route in the network of nodes.

Multiple scenarios were obtained where all the costs associated with f_2 are distributed by the links that minimize f_1. Therefore, when it is desired to minimize only f_1, links are obtained when it is desired to maximize f_2. Similarly, when only f_2 is minimized, the chosen links have no costs for this function. Those links, since they are not optimal for f_1, maximize the value of said function.

To describe the understanding of the problem in more detail, see the terminology described below:

Be

$$D : \text{Set of optimal links selected by Dijkstra's Algorithm}$$
$$C_1 : \text{Set of optimal links to minimize} f_1$$
$$C_2 : \text{Set of links where there are costs associated with} f_2$$

Where from the student pedestrian traffic displacement model, the following sets relationship occurs:

$$C_1 \epsilon D$$
$$C_2 \epsilon D$$

4.2 Single Cost Approach to Solution Heuristic

For the solution heuristic we proceeded to use Dijkstra using as the only cost the weighting of the normalization of the costs of f_1 and f_2. For this, the scale characteristic normalization method was applied as follows:

$$f_1^{'}(i,j) = \frac{f_1(i,j) - min(f_1)}{max(f_1) - min(f_1)} \tag{4.2.1}$$

$$f_2^{'}(i,j) = \frac{f_2(i,j) - min(f_2)}{max(f_2) - min(f_2)} \tag{4.2.2}$$

Where $f_k^{'}(i,j)$ is the normalization of the scaling characteristic of f_k in the link of node i and node j. $max(f_k)$ corresponds to the maximum value in the data set of the function k. For the case of both functions, the maximum value was the integer 99999, since that was the default value in the cost matrix where there was no link between a pair of nodes. Finally, $max(f_k)$ is the minimum value in the data set of the function k. For f_1 it corresponded to the link between nodes of shorter length, and for f_2 it was the minimum number of students who traveled on a link given a cycle, day and time.

Finally, the costs of the normalized functions were weighted through the weights that were later defined from the input parameters entered by the user, and they were assigned to a new function. Likewise, from the new function, we defined the heuristic solution that would have as parameters the inputs received by the front-end of the mobile application:

$$F(i,j) = w \cdot f_1^{'}(i,j) + (1-w) \cdot f_2^{'}(i,j) \tag{4.2.3}$$

$$H(s,d) = Dijkstra(F,s,d) \tag{4.2.4}$$

Where F is the function that stores the weight of the two initial normalized functions. w corresponds to the weight that the user implicitly enters as a parameter in the front-end to give a weight to each function. $Dijkstra$ is the function

that returns the optimal links given the unified cost function F, a source node s and a destination node d given a cycle, day and time. Finally, H is the function that represents the problem's heuristic solution. Once the heuristic solution was modeled, the results of the algorithm were compared with the solutions provided by the Pareto Front in different scenarios.

4.3 Results Comparisson and Validation with the Pareto Front

For the validation of the results produced by the heuristic developed with the Pareto Front points, we defined different scenarios to vary the pedestrian traffic given a cycle, day and time with the same origin and destination node in each scenario. This was made with the purpose of comparing how is the variation of the optimal route of two points from the assignment of different costs of f_2 (since the costs of f_1, being the length of each link between nodes remained fixed in the different executions. However, it should be noted that the accumulated cost of f_1 changes from the different routes selected).

The specifications of the different scenarios are observed in Table 3.

For the generation of the Pareto Front, the methods described in Subsect. 3.3 was used.

Regarding the values of w for F, different values were iterated in the domain of w ($[0, ..., 1]$) to observe the points got by the heuristic in each iteration.

Table 3. Specifications of the scenarios chosen for the comparison of the solutions thrown by the solution heuristic and the Pareto Front.

Scenario number	Academic cycle	Day	Schedule	Source building	Destiny building
1	1	Wednesday	10:50–11:00	Julio Mario Santodomingo	Mario Laserna
2	1	Monday	13:30–14:00	Julio Mario Santodomingo	Mario Laserna
3	1	Thursday	13:30–14:00	Julio Mario Santodomingo	Mario Laserna

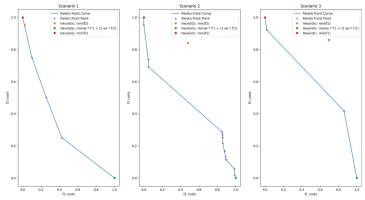
It is observed in Fig. 1a that different points were got from the Pareto front, as well as the heuristic solution from the same source and destiny buildings, but with different pedestrian traffic and assigned weights for each $w \epsilon F$.

f As main evidence, we observed that since it is a problem with only two optimal solutions available the Pareto Front, points were generated particularly at the limits of each function. Likewise, it was obtained that when $w \neq 0$ or $w \neq 1$ the heuristic yielded the same solution in each iteration. Therefore, the solutions thrown by the heuristics behave as if they were only minimize f_2, this is due to a single cost function where the costs of f_2 associated with the optimal f_1 paths made the heuristic look for paths that were not optimal for f_1 in order to minimize weight associated with f_2. For such reason it can be seen that the algorithm solution does not generate optimal results to minimize f_1.

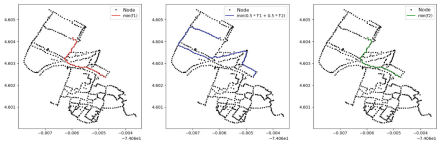
Thus, in the iteration when $w = 1$-which is when the heuristics were only considered minimizing f_1- a solution was got at one end of the Pareto Front, where the associated cost of f_1 is minimized and the cost associated with f_2

is maximized. However, on iteration when $w = 0$, which is the case where the heuristic only minimized f_2. We found that although that the solutions obtained minimize the accumulated cost of f_2 with respect to the iteration scenario when $w = 1$, These solutions are solutions dominated by the points that make up the Front, and additionally, it is a non-optimal solution when minimizing f_2 compared to the solutions obtained in the iterations when $w \notin \{0, 1\}$

For additional information of the described scenarios, see Fig. 1b, in where the optimal path selected by the heuristic is shown for the different values iterated by w in its domain. Note the similarity of routes chosen when the heuristic only minimizes f_1 or f_2, and as in a different scenario (Scenario 2) increases the cumulative cost of f_1 to minimize the cumulative cost of f_2 From the observations derived from the results, and from the naturalness of the modeled problem described in Subsect. 3.2, we decided to only expose two possibilities of paths in the mobile application, where each option corresponds to the minimization of each function. Therefore, to obtain these results, the domain of w was configured to only take the values of 1 and 0.5 (which is a decimal that assigns equal weight to both normalized functions).



(a) Pareto Fronts and Heuristic solutions for scenarios 1, 2 and 3



(b) Mapped routes

Fig. 1. Scenarios and routes comparisons for different values of w between 0 and 1

4.4 Results Comparisons and Validation Based on the Mathematical Model

From the configurations established for the solution heuristic in the Subsect. 3.2 - where f_1 and f_2 were equally normalized and weighted to minimize f_2 - The execution times used by both the mathematical model and the heuristics to minimize f_1 and f_2 were compared, as well as the values taken by the functions in each scenario.

The mathematical model was configured in the optimization framework *Pyomo* [4] and the heuristic was written and executed in a *script* written in the programming language *Python*.

The specifications of the chosen and executed scenarios can be observed in the Table 4.

Table 4. Scenarios specifications.

Scenario number	Academic cycle	Day	Schedule	Source building	Destiny building
1	1	Wednesday	10:50–11:00	Julio Mario Santodomingo	Mario Laserna
2	1	Thursday	13:50–14:00	Julio Mario Santodomingo	Mario Laserna
3	1	Monday	13:50–14:00	Julio Mario Santodomingo	Mario Laserna

First, the execution times of each scenario were compared, where it was observed that the execution times got by the heuristic are approximately 98.45% less than the times obtained by the mathematical model. The previous result facilitates the response time of the calculation of an optimal route when the heuristic is integrated as a service exposed to the mobile application.

Both the heuristic and the mathematical model were executed on an Intel ®Core i5-6200U, 2.3 GHz, 7.7 GiB. Additionally, in the executions made by *Pyomo* the mathematical model was solved using the *GLPK solver*.

The details of the execution times for each scenario can be observed in the Table 5. Subsequently, the resulting values of f_1 and f_2 were compared for each tool in the minimization of each function of each scenario. As it is observed in Table 5, the heuristics got relatively lower values for f_2 compared to the mathematical model in each case of minimization of f_1 (It should be noted that in these cases the two tools got the same result for f_1), while in the cases of minimization of f_2 each tool had the same results for this function.

Table 5. Comparison of the resulting f_1, f_2 values in the execution of each scenario with their corresponding execution time.

Scenario number	Function to minimize	Heurístic value for f_1 (units)	Mathematical model for f_1 (units)	Heurístic value for f_2 (units)	Mathematical Model value for f_2 (units)	Heuristic Execution Time (sec)	Mathematical model execution time (sec)
1	f_1	$3.08 \cdot 10^{-3}$	$3.08 \cdot 10^{-3}$	495	495	2.761	182.504
	f_2	$6.58 \cdot 10^{-3}$	$6.99 \cdot 10^{-3}$	5	5	2.519	164.167
2	f_1	$3.08 \cdot 10^{-3}$	$3.08 \cdot 10^{-3}$	540	540	2.522	153.907
	f_2	$6.68 \cdot 10^{-3}$	$7.48 \cdot 10^{-3}$	19	19	2.423	171.088
3	f_1	$3.08 \cdot 10^{-3}$	$3.08 \cdot 10^{-3}$	682	682	2.544	161.058
	f_2	$6.64 \cdot 10^{-3}$	$7.11 \cdot 10^{-3}$	35	35	2.695	165.645

5 Mobile Application Design

We implemented *REST API software* architecture style [20] using the web *framework Flask* [18] written in *Python*. Likewise, the steps and instructions for the heuristic solution (see Subsect. 4.2) were developed in the programming language *Python*. The input parameters that the *REST API* received for the solution heuristic to calculate the optimal path were: Text string containing the initials of the building in which the student is located (s). Text string containing the acronym of the building the student wants to go to (d). Text string that

shows the academic cycle in which it is desired to calculate pedestrian traffic. By default, this value corresponds to "*1*" (*cycle*). Text string that indicates the particular day set to calculate pedestrian traffic (*day*). Text string that indicates the schedule for changing classes of pedestrian traffic. Note that class change hours are expressed in the 24-h system (*schedule*). Integer that shows which function it is wanted minimize (1) or if it is wanted to minimize the transport cost function (0), if it is wanted to minimize the interaction cost function (w). For more details on what these functions represent, see Sect. 3.

It should be noted that these methods are received by the server through a *POST* request. To test the connection between the *Flask* server and the client, the server was run on a local machine Intel ®Core i5-6200U, 2.3 GHz, 7.7 GiB machine and perform client API requests through the *Postman* [9] software.

For the development of the mobile application, we designed so that it could be executed in the operating system *Android 9* [13]. Likewise, through the programming language *Kotlin* [16] and the graphical interfaces for the design of the *UI* components provided by the *Android Studio* development environment, we implemented the *front end* logic and graphical components to show the probability of susceptibility and the accumulated distance when traveling a route.

For the generation of the university campus map, a *Google Maps* software development kit was used available for *Android* [15].

In Fig. 2 it can be observed the graphical interface of the application with the susceptibility probability and the accumulated distance in the lower bar as well as the optimal route drawn. It should be noted that the icons and figures of the application were taken from the webpage *Flaticon.com* [11].

Regarding the selection of parameters to be sent to the server, a graphical Android component (*AlertDialog*) was used to create a form that would allow the user to select the source building, the destiny building, the opening hours, change of classes available for that day and the route criteria to be minimized. In order to get significant records of the students' trajectory for the generation of pedestrian traffic, we assumed that the default academic cycle parameter was 1. Likewise, the day was automatically assigned according to the current day on which the application was used. If the user uses the application on Sunday - the day on which the university does not hold academic classes - the application assigns Monday as the default parameter. In the Fig. 3 and Fig. 4 it can be observed the parameter selection form.

Once the parameters were selected, when pressing the *Get Path* button of the form, the application - using the *HTTP Android Volley* [14] library - send the request *POST* to the *IP* address of the local machine where the server was deployed.

Once the server response is received, the application updated its interface, showing in the lower bar the susceptibility probability that the user has when traveling the route, as well as the accumulated distance that they have to walk. An example of how the route is drawn in the application can be observed in the Fig. 2.

In Fig. 5 it is observed the final architecture diagram.

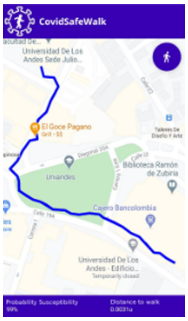


Fig. 2. Screenshot of the mobile application.

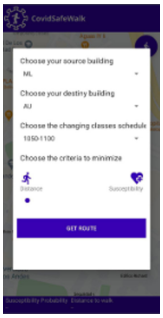


Fig. 3. Parameter selection form with the distance minimization criterion selected.

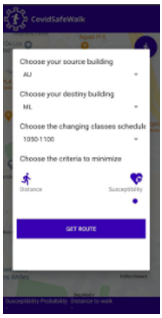


Fig. 4. Parameter selection form with the susceptibility minimization criteria selected.

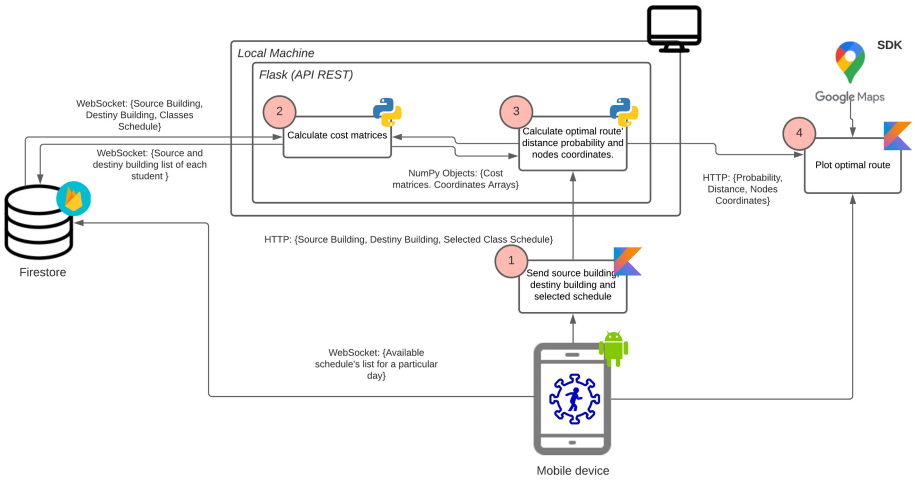


Fig. 5. Diagram of the software architecture built. In the upper right corner of each activity/device it can be observed the programming language, software or platform used. In the upper left corner is the step number of the sequence of steps followed by the process execution. The intermediate texts between the arrows connecting each activity/device show the communication protocols used.

5.1 Validations

In order to observe the behavior and functionalities of the final product through its graphical interface, the scenarios presented in the Table 6 were considered to observe the variation between the probability of susceptibility and the accumulated distance when minimizing each criteria, given the different origin-destination buildings and pedestrian traffic on a certain day and class change schedule.

It should be noted that the scenarios presented took Thursday and "1" as the default day and academic cycle respectively, since the tests were made on that specific day of the week.

Table 6. Specifications of the tested scenarios on the mobile application, describing the obtained values for each criteria based on the selected criteria to minimize

Scenario number	Schedule	Source building	Destiny building	Selected criteria	S. probability value (percentage)	A. distance value (units)
1	10:50–11:00	Mario Laserna	Aulas	Distance	99%	0.0016
				Susceptibility	85%	0.0025
2	13:30–14:00	Mario Laserna	City U	Distance	99%	0.0023
				Susceptibility	66%	0.0037
3	10:50–11:00	J. M. Santodomingo	Mario Laserna	Distance	100%	0.0031
				Susceptibility	99%	0.0068

6 Conclusions

Based on the developed work, a *software* tool has been built in order to optimize the most important criteria to be considered by a pedestrian - a Universidad de Los Andes student- in terms of health effectiveness and safety. Therefore, the developed mobile application allows a pedestrian - related to Universidad de Los Andes - to consult in a easy and quickly way a mobile application that solves the COVID-19 risk contagion problem from a mobility pattern and displacements perspective.

Likewise, based on the pedestrian traffic simulation and the proposed cost functions, it has been possible to generate and understand a pedestrian agglomerations' approximation, given a set of student classes that had time attributes. Moreover, based on the data hierarchy and aggregated pedestrian mobile users data - given the case the developed mobile application is used by multiple users - strategies can be established for data analysis from the different hierarchy data levels (day, academic cycle, etc.). This has been done with the purpose of understanding the mobility patterns that university students generate, and therefore, allows a high-level decision making process that avoids pedestrian agglomerations in a certain period of time.

Additionally, as a future work feature, the validation process must be complemented from the medical domain point of view. In other words, the inclusion and approval of health entities that comprehend the COVID-19 contagion processes could help to calibrate and adjust our work in order to propose a joint solution validated from the computational and medical point of view.

References

1. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
2. Haimes, Y., Lasdon, L., Wismer, D.: On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE Trans. Syst. Man Cybern. **SMC-1**(3), 296–297 (1971). https://doi.org/10.1109/TSMC.1971.4308298
3. Harko, T., Lobo, F.S.N., Mak, M.K.: Exact analytical solutions of the Susceptible-Infected-Recovered (SIR) epidemic model and of the SIR model with equal death and birth rates (2014). https://arxiv.org/abs/1403.2160
4. Hart, W., Watson, J.P., Woodruff, D.: Pyomo (2019). https://pyomo.readthedocs.io/en/stable/
5. Kumar, N., Oke, J.B., Nahmias-Biran, B.-h.: Activity-based contact network scaling and epidemic propagation in metropolitan areas (2020). https://arxiv.org/abs/2006.06039
6. Muller, S.A., Balmer, M., Neumann, A., Nagel, K.: Mobility traces and spreading of COVID-19 (2020). https://www.medrxiv.org/content/10.1101/2020.03.27.20045302v1.full.pdf
7. Organización Mundial de la Salud: Q&A. How is COVID-19 transmitted (2020). https://www.who.int/news-room/q-a-detail/q-a-how-iscovid-19-transmitted
8. Schultz, M., Fuchte, J.: Evaluation of aircraft boarding scenarios considering reduced transmissions risks (2020). https://www.mdpi.com/2071-1050/12/13/5329
9. Asthana, A., Kane, A., Sobti, A.: Postman API client (2014). https://www.postman.com/product/api-client/
10. Firebase Inc.: Firebase (2012). https://www.firebase.google.com/
11. Freepik Company: (2020). https://www.flaticon.com/
12. Geographic JSON working group: GeoJSON (2016). https://geojson.org/
13. Google LLC: Android pie (2019). https://www.android.com/versions/pie-9-0/
14. Google LLC: (2020). https://developer.android.com/training/volley
15. Google LLC: Maps SDK for Android (2020). https://developers.google.com/maps/documentation/android-sdk/overview
16. JetBrains: Kotlin (2020). https://github.com/JetBrains/kotlin/releases/latest
17. KNIME: KNIME (2020). https://www.knime.com/
18. Grinberg, M.: Flask Web Development: Developing Web Applications with Python. O'Reilly Media Inc., Sebastopol (2018)
19. Python Software Foundation: Python 3.6 (2020). https://www.python.org/
20. World Wide Web Consortium: Web services architecture (2004)
21. Universidad de Los Andes: Campus en Cifras (2019). https://campusinfo.uniandes.edu.co/es/campusencifras
22. Universidad de Los Andes: Facts and Figures (2019). https://planeacion.uniandes.edu.co/en/statistics/factsand-figures
23. Universidad de Los Andes. Vicerrectoría de Servicios y Sostenibilidad: Reporte de Sostenibilidad 2019 (2019). https://sostenibilidad.uniandes.edu.co/images/Reporte2019/Reporte-de-sostenibilidad-2019W.pdf
24. Zadeh, L.: Optimality and non-scalar-valued performance criteria (1963). https://ieeexplore.ieee.org/abstract/document/1105511