Efficient Transit Network Design, Frequency Adjustment, and Fleet Calculation Using Genetic Algorithms

Miguel Jiménez-Carrión[1*], Gustavo A. Flores-Fernandez[2] and Alejandro B. Jiménez-Panta[3]

[1*] Professor, Facultad de Ingeniería Industrial, Universidad Nacional de Piura, Castilla-Piura, Perú. mjimenezc@unp.edu.pe, Orcid: https://orcid.org/0000-0001-9632-5085

[2] Egresado de la Facultad de Ingeniería Industrial de la Universidad Nacional de Piura, Perú. fgustavoalexis@gmail.com, Orcid: https://orcid.org/0000-0002-4488-4325

[3] Egresado de la Facultad de Ciencias de la Ingeniería, de la Pontificia Universidad Católica del Perú, Perú. alejandrob.jimenezp@pucp.edu.pe, Orcid: https://orcid.org/0009-0002-4688-5935

Received: July 17, 2023; Accepted: September 24, 2023; Published: November 30, 2023

Abstract

The main objective of this study was to implement a computational prototype in two stages: the first stage primarily focused on generating efficient routes based on an evolutionary algorithm. In other words, the complex computational problem was solved in the first stage. The second stage then shifted its focus towards determining the fleet size and frequencies using an allocation algorithm. This approach was designed to address the complex combinatorial search problem within a public transportation network. In the first stage, the prototype utilizes the metaheuristic known as Genetic Algorithms (GA). Within the GA operators, an innovative method called "aggregated crossover" is employed, with an additional mutation procedure that maintains feasible descendants. In the second stage, an allocation algorithm is used, taking into account the routes generated in the first stage. The results demonstrate that in the first stage, the GA metaheuristic consistently delivers highly efficient routes in each run, confirming that the combinatorial complexity of the problem is effectively resolved in this initial phase. These results were validated on Mandl's Swiss Road network, showing superior solutions compared to those presented in previous studies. Notably, the execution time for this process is only 35 minutes.

Keywords: Genetic Algorithm, Public Transportation, Route Design, Fleet, Frequency, Computational Evolutionary Processes.

1 Introduction

Modeling the urban transportation problem from a systemic approach perspective involves determining each of the components involved, such as transportation routes, transportation vehicles, users utilizing transportation vehicles, urban transportation service demand, the regulating entity for transportation in the city, service quality standards, transportation policies, as well as the growth rate of the vehicle fleet, population growth rate in the city, and when we add traffic light policies, modes, and control directives for these systems to the mix, it becomes highly complex, not only to model but also to optimize. In this

Journal of Interner Services and Information Security (JISIS), volume: 13, number: 4 (November), pp. 26-49
DOI: *10.58346/JISIS.2023.I4.003*

*Corresponding author: Professor, Facultad de Ingeniería Industrial, Universidad Nacional de Piura, Castilla-Piura, Perú.

regard, authors like Fan and Mumford (2010) state that the Urban Traffic Routing Problem (UTRP) is NP-Hard and involves route design. Asadi-Bagloee and Ceder (2011) express that the transit network design problem is complex and cumbersome. Cipriani et al. (2012) suggest that transit network design is a complex non-convex problem. Similarly, Nicolić and Teodorović (2013) state that transit network design is one of the most critical problems faced by transit operators and city authorities worldwide, and they further argue that it belongs to the class of difficult combinatorial optimization problems, with a challenging optimal solution to discover. Kiliç and Gök (2014) also contend that the transit route design problem requires the use of complex combinatorial search techniques. Ahmed et al. (2019) affirm that the problem falls into the category of challenging combinatorial problems, with an optimal solution that is hard to find due to the complexity arising from the vast search space and the numerous constraints imposed on solution construction.

Mahdi-Amiripour et al. (2015) argue that optimal routes must meet passenger demand matrix and achieve a better compromise solution for users, operators, and the community. They propose a genetic algorithm as a tool to handle the complexity of the problem, consisting of four steps: (1) generating a set of potential routes, (2) designing the bus network, (3) verifying the routes for implementation, and (4) examining route extensions for improvement. The proposed method is validated through a reference bus network and a single-set problem in the Mandl's network. On the other hand, Arbex and da-Cunha (2015) maintain that the multi-objective problem of transit network design and frequency setting (TNDFSP) involves finding a set of routes and their associated frequencies for operation in an urban public transportation system. TNDFSP is a difficult combinatorial optimization problem, with a large search space and multiple constraints, leading to numerous infeasible solutions. In their work, they propose an Alternate Objective Genetic Algorithm (AOGA) to efficiently solve it, aiming to cyclically alternate between generations. The two proposed objectives are to minimize costs for both passengers and operators (Saenko, I., 2014).

When studying the transit network design problem from the perspective of mathematical programming, Cancela et al. (2015) defines the number and itinerary of bus routes and frequencies for the public transportation system, adding that the routes must cover a specific origin-destination demand, and they must be in harmony with users and operators. In this scenario, they review existing mathematical programming formulations and propose a mixed-integer linear programming (MILP) formulation. However, when validating their mathematical formulation on Mandl's Swiss Road network, they present two solutions, one with 20 routes and another with 12 routes, and compare them with the results of Asadi-Bagloee and Ceder (2011). This approach contrasts with a complete constructive multi-objective algorithm where they handle the multi-criteria nature of the problem to find Pareto-optimal solutions, proposed by Owais and Osman (2018). They also develop a new frequency configuration algorithm and conduct experimental studies on two real-sized networks to validate the performance and robustness of their algorithm, using Mandl's Swiss Road network.

In the same mainstream, Buba and Lee (2019) consider the Urban Transit Network Design Problem (UTNDP) as an NP-hard problem, characterized by a vast search space, multi-objective nature, and multiple constraints, making the evaluation of candidate route sets time-consuming and challenging. In their work, they propose a Hybrid Differential Evolution with Particle Swarm Optimization (DE-PSO) algorithm to solve the UTNDP, aiming to simultaneously optimize the route configuration and service frequency with specific objectives like minimizing costs for passengers and operators. The computational results of the proposed hybrid algorithm outperform the benchmark in most previous studies using the Mandl's Swiss Road network.

Yang and Jiang (2020) and Katsaragaskis et al. (2020) tackle the urban transportation problem with different metaheuristics, both aiming to balance costs for passengers and operators. The former implements an algorithm based on genetic algorithms to produce an approximate Pareto front with much higher solution quality and improved computational efficiency. Their experiments were conducted on the Mandl's network and four others. Meanwhile, the latter proposed to solve the urban transportation problem by implementing their algorithms based on Cat Swarm Optimization (CSO), achieving near-optimal solutions for the problem. In particular, the latter modifies the classic cat swarm optimization algorithm. Concerning the urban transportation problem, Vermeir et al. (2021) argue that the state-of-the-art lacks optimal solutions and propose a Branch-and-Bound algorithm. They introduce three concepts to determine optimal solutions: (1) a new domain-based group generation method, (2) the introduction of essential links, i.e., links that can be determined in advance and must be present in the optimal solution, and (3) a new network representation based on adding only additional edges. Experiments demonstrate their significant contribution to the algorithm's success, and the results were validated on Mandl's Swiss Road network.

Finally, Jiménez-Carrión et al. (2023) presented a computational prototype for generating urban transportation routes, inspired by evolutionary theory, to address the complex combinatorial search problem of a public transportation network. The prototype uses the Python programming language for coding and employs the metaheuristic known as genetic algorithms (GAs). The results show that the computational prototype is accurate and fast, delivering highly efficient solutions in each run for the urban transportation problem. The results were validated on Mandl's Swiss Road network, proving to be better solutions than those presented in previous studies. In conclusion, the evolutionary process of the computational prototype converges with highly satisfactory solutions due to innovative mechanisms of genetic operators like crossover, mutation, and quality function, giving equal weight to the parameters it comprises. However, it does not determine the fleet or the frequency of buses. It has been observed that in each run of the computational algorithm, the percentages between direct trips and trips with transfers tend to vary, impacting the quality of the individual, becoming unstable even when this change is relatively insignificant in the results. Regarding this issue, the authors' solution for the set of 7 routes highlights this problem, showing substantial improvement when considering transfer minimization criteria.

In this scenario, this research was proposed with the aim of improving the algorithm of the first phase proposed by Jiménez-Carrión et al., (2023), which had optimal results when applying innovative methods designed by the same author, such as the method called "reproducción agregada" (aggregated reproduction), which, together with an equally innovative mutation procedure called "poda" (pruning), always maintains feasible offspring. On the other hand, to develop the second phase to complement the solution, delivering the design of the routes, as well as determining the frequency on each route and the necessary fleet of buses. With this commitment and a clear understanding that in the urban transport route problem, despite conflicting interests, highly efficient routes can be generated that minimize the average passenger travel time, the total route time across all routes, while maximizing the percentage of passengers taking direct trips without transfers, satisfying the total demand. Additionally, to determine the necessary bus fleet and frequencies on each route, thus solving the problem of urban transport planning and management known as the Urban Transport Routes Problem (UTRP). The importance of this article is to demonstrate that the two well-defined phases of the UTRP can be performed separately, unlike most algorithms that combine the two phases into a single algorithm, as shown in the discussion section, resulting in lower-quality solutions than those defined by the proposed algorithm. Considering that in the second phase, fleet distribution and frequency seek a balance between demands on routes and repeated sections, unlike what is defined by other authors who use probability distributions like the

multinomial logit model, which in many cases biases demand allocation to different routes. In this case, we propose that the complexity of the problem is resolved in the first phase, and on the other hand, this contribution will serve as a tool for use by authorities responsible for the administration and control of urban transport, a problem that is exacerbated with the increase in the automotive fleet in cities. Finally, there are solutions that achieve a minimum total travel time for vehicles on all routes at the expense of a high average travel time per passenger. On the other hand, there are solutions that achieve a very low average travel time per passenger at the cost of a high travel time for vehicles on all routes. This trend continues to this day. For this reason, we propose that there must be an equitable compromise between these conflicting parameters that leads us to find highly satisfactory solutions. To this end, the following methodology was proposed.

2 Methodology

The steps to successfully complete the current research consisted of: 1) expanding the literature review to include articles that not only conclude with route design but also with fleet allocation and frequency determination; 2) using the mathematical model, problem definition, and chromosome design of the computational algorithm proposed by Jiménez-Carrión et al. (2023), making improvements in the algorithm code so that in each algorithm run after route design, it always delivers the correct quality of the route set for each solution, concluding with the verification of algorithm stability using a 2 x 3 factorial design, with two factors and three levels for each factor and four repetitions; 3) developing the code for the second phase with its corresponding functionality for fleet allocation and frequency determination for each route in the route set; 4) verifying the integrity and consistency of the second-phase algorithm; 5) conducting the analysis and discussion of results for the algorithm overall, both in route design and fleet allocation and frequency determination; 6) finally, drawing conclusions.

Definition of the Problem

As previously mentioned, the research focuses on the Mandl transportation network, shown in Figure 1, which is expressed as the undirected graph G = (N, A), where 'N' is the set of nodes in the network n0, n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, and 'A' is the set of edges a(0,1), a(1,2), a(1,3), a(1,4), a(2,5), a(3,4), a(3,5), a(3,11), a(5,7), a(5,14), a(6,9), a(6,14), a(7,9), a(7,14), a(8,14), a(9,10), a(9,12), a(9,13), a(10,11), a(10,12), a(12,13). Each node or vertex ni in N represents a stop or another type of demand node, and each edge a (i, j) in the set A represents a weighted edge between stops i and j. A weight can represent the distance or travel time between nodes or vertices or can be a function of these parameters, as proposed by Kiliç and Gök (2014). Each edge is associated with a positive integer weight. Travel demands in a transportation network are expressed as a matrix called the demand matrix, D, in which each element (i, j) represents the number of passengers traveling from node i to node j per day. Although transport demands are dynamic in nature, the elements of D are calculated for a specific period within a day. The demand matrix D for this network is shown in Table 1, which has 16 rows and 16 columns, with the first column and the first row representing the nodes of the network. It should be noted that some elements in the table are zero (0), indicating no passenger demand between i and j. However, there are elements with very high demands, such as element a(5,9) or element a(9,5), which have a demand of 880 passengers. Thus, the transit network route search problem is defined by the graph G and the matrix D, and the solution to the problem is a set of routes (a subgraph of the network), as it does not include all edges between nodes. Importantly, the solution must allow reaching all nodes in the network.

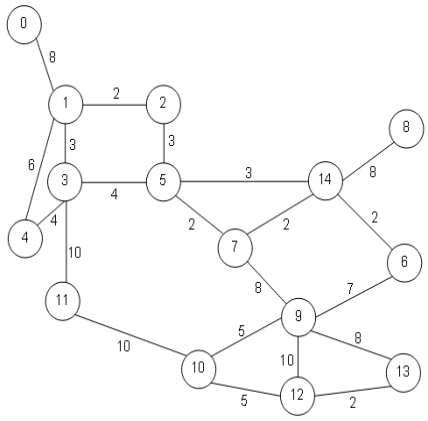


Figure 1: Mandl's Network (Adapted from Arbex and da-Cunha, 2015)

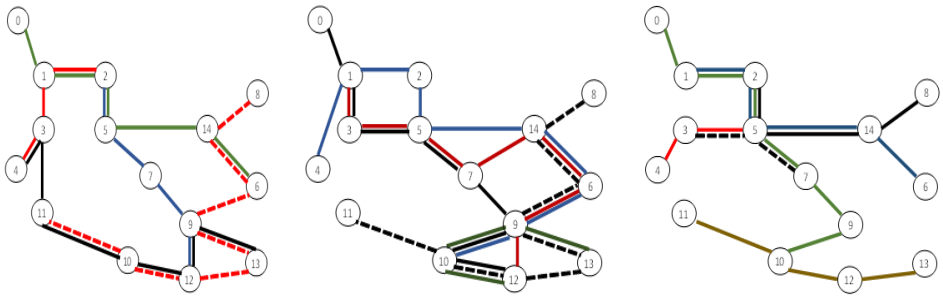
Table 1: Demand Matrix (D) between Nodes in Mandl's Network (Adapted from Arbex and da-Cunha, 2015)

D =

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	400	200	60	80	150	75	75	30	160	30	25	35	0	0
1	400	0	50	120	20	180	90	90	15	130	20	10	10	5	0
2	200	50	0	40	60	180	90	90	15	45	20	10	10	5	0
3	60	120	40	0	50	100	50	50	15	240	40	25	10	5	0
4	80	20	60	50	0	50	25	25	10	120	20	15	5	0	0
5	150	180	180	100	50	0	100	100	30	880	60	15	15	10	0
6	75	90	90	50	25	100	0	50	15	440	35	10	10	5	0
7	75	90	90	50	25	100	50	0	15	440	35	10	10	5	0
8	30	15	15	15	10	30	15	15	0	140	20	5	0	0	0
9	160	130	45	240	120	880	440	440	140	0	600	250	500	200	0
10	30	20	20	40	20	60	35	35	20	600	0	75	95	15	0
11	25	10	10	25	15	15	10	10	5	250	75	0	70	0	0
12	35	10	10	10	5	15	10	10	0	500	95	70	0	45	0
13	0	5	5	5	0	10	5	5	0	200	15	0	45	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Instances of Solution of the Problem

The solution instances of the problem are feasible solutions that meet all the problem's conditions. Figure 2 shows three instances as examples. It can be observed that in any instance, one can reach any node. Some passengers have direct trips without transfers, as in the case of the passenger traveling from node 0 to node 6 in instance a), using the green route (0-1-2-5-14-6). Others make trips with one transfer, such as the passenger traveling from node 8 to node 7 in instance b), using the dashed black route in the segment (8-14) and then the red route for the segment (14-7). And some passengers make trips with two transfers, like the passenger traveling from node 4 to 11 in instance c), using the red, green, and brown routes. The first transfer is made from the red line to the green line using the segments (4-3, 3-5) on the red line, then the green line with segments (5-7, 7-9, and 9-10), and finally, the second transfer is made from the green line to the brown line using the segment (10-11).



(a) (b) (c)

Figure 2: Instances of Feasible Routes, Based on the Mandl Network

Mathematical Model for Route Design

The mathematical model has two objectives: to minimize the average passenger travel time ($\overline{T}_V$) nd the total travel time of all vehicles on the routes (T_{RV}), The constraints ensure the fulfillment of all problem conditions. The model grows combinatorially with size

$$\text{Function Objective: FO} = \text{Minimize}(\overline{T}_V) + \text{Minimize}(T_{RV}); \text{ s.a} \quad (1)$$

$$\overline{T}_V = P_0\overline{T}_{V0} + P_1\overline{T}_{V1} + P_2\overline{T}_{V2} + \cdots + P_k\overline{T}_{Vk} \quad (2)$$

The parameters of the aforementioned equations are observed below:

$$\overline{T}_{V,k}=\left(\frac{\sum_{i=0}^{i=n-1}\sum_{j=0}^{j=n-1} D_{i,j,k}T_{i,j,k}}{\sum_{i,j=0}^{i,j=n-1} D_{i,j,k}}\right) \qquad \forall\ k = 0, 1, 2, \ldots, k \quad (3)$$

$$P_k=\left(\frac{\sum_{i,j=0}^{i,j=n-1} D_{i,j,k}}{D_T}\right) \qquad \forall\ k = 0, 1, 2, \ldots, k \quad (4)$$

$$D_T=D_{i,j,0}+D_{i,j,1}+D_{i,j,2}+\ldots+D_{i,j,k} \quad (5)$$

$$T_{RV}=T_{R0}+T_{R1}+T_{R2}+\cdots+T_{Rm} \quad (6)$$

$$T_{Rm}=\sum_{i,j=0}^{i,j=n-1} X_{i,j,m}\, T_{i,j,m} \quad (7)$$

Notation: Symbols and Variables

n = 15, number of nodes in the network, positive integer

m = number of routes in the route set, positive integer

T_{ij} = time in minutes to go from node i to node j, given by the time matrix

D_{ijk} = number of passengers traveling from node i to node j, using k transfers, k = 0, 1, 2, …, k

P_k = percentage of passengers traveling with k transfers, k = 0, 1, 2, …, k

$T_{V,k}$= average travel time with k transfers, k = 0, 1, 2, …, k

… = represents continuity from the previous way in the route of the number of transfers (k), k=0, 1, 2,…, k

T_{RV} = vehicle travel time on all routes

$\overline{T}_V$ = average travel time on the set of current routes

D_T = total passenger demand between node i and node j

T_{ijk} = Travel time from node i to node j with k transfers, k = 0, 1, 2, …, k; determined by finding the shortest route using the current route set and 5-minute time increments for each transfer

X_{ijk} = edge usage of the network from node i to node j on path m; m = 0, 1, 2, …, m, binary variable takes the value 1 when used and 0 when not

Additional Restrictions

Each node in any route should not repeat to avoid backtracking. An edge can be used more than once in the set of routes. Finally, the set of routes must allow reaching any node. These considerations are taken into account when designing the route generator.

Chromosome Design and Adaptive Function

An individual represents a solution to the problem, and thus, an individual is represented as a list of lists, with each inner list containing integers that identify nodes in the network. These nodes are implicitly connected, meaning that each pair of nodes has an edge and consequently a distance or time to travel between them. The number of nodes in each list, or rather in each route, is variable and is generated based on the number of routes, which can range from a minimum to a maximum, and a specific number of nodes in the route, which can also vary between a minimum and a maximum number. The instances shown above are some examples of individuals in which, in all cases, it is ensured that starting from a particular node, all other nodes can be reached through the routes. Additionally, nodes are not repeated in the routes to avoid backtracking, and the instance ends up being a subgraph of the original network because not all edges of the network are used.

Quantifying the quality of a set of routes is not straightforward, as there are conflicting objectives, as stated by Kiliç and Gök (2014) when they mention that 'Stakeholders often have conflicting interests.' In this algorithm, it is proposed that the quality function is a compromise of three important parameters, such as minimizing the total travel time of vehicles to transport the existing passenger demand in accordance with the demand matrix 'D,' minimizing the average passenger travel time, and maximizing the percentage of passengers with direct transportation without transfers. However, it is necessary to consider the movement of passengers through the designed routes. This involves quantifying the percentage of passengers who travel directly without making transfers, those passengers who make one transfer, two transfers, and more transfers. In this context, it was proposed that evaluating an individual should be in harmony with these percentages. Therefore, the quality function was defined as shown in the dimensionless equation (8):

$$\text{Quality}= \frac{0.9P_0 - 0.04P_1 - 0.03P_2 - 0.02P_3 - 0.01P_4 - \cdots\ 0.00P_k}{(\log(\overline{T}_V) + \log(T_{RV}))} \quad (8)$$

Logarithms help minimize the disparity between the average travel time and the travel time of all vehicles on all routes, so the denominator favors quality if this sum is small and penalizes it if it's large. Furthermore, if there are passengers with one or more transfers, the travel time from i to j increases by 5-time units for each transfer. This allows for a fair comparison of our algorithm with other proposals, ensuring that transfers are also considered in the fitness or quality function. In other words, if a transfer from one vehicle to another is necessary to reach the final destination, the travel time increases by 5 minutes. If two transfers are needed, the time increases by 10 minutes, and so on. This function requires determining the travel times for direct trips first, then for one transfer, two transfers, and so on. Additionally, the demand of passengers without transfers, with one transfer, with two transfers, etc.,

must be determined proportionally. This way, the quality of the three previous instances is evaluated as follows:

Instance a: [77.59, 20.87, 1.54] 12.98, 139.0; Quality = 9.196; This indicates that 77.59% of passengers travel directly, 20.87% of passengers use one transfer, and the remaining 1.54% make two transfers to reach the final destination. The average travel time for passengers from i to j is 12.98 min, and the travel time for vehicles on all routes is 139 min. Applying the formula in equation (5), the quality is 9.196. The percentages are obtained by determining the transportation costs.

Instance b: [85.48, 14.07, 0.45] 11.38, 153.0; Quality = 10.232; The second instance indicates that 85.48% of passengers travel without transfers, 14.07% make one transfer, and 0.45% of passengers make 2 transfers. The average travel time from i to j is 11.38 min, and the travel time for vehicles on the routes is 153 min. The quality of this instance is better than the previous one, at 10.232. Comparing it with the previous instance, you can see that the percentage of passengers not using transfers increased from 77.59% to 85.48%. Additionally, the average cost decreases from 12.98 to 11.38 min, while the travel time of vehicles increases from 139 to 153 min.

Instance c: [61.79, 37.06, 1.16] 13.21, 83.0; Quality = 7.728; In this instance, the percentage of direct passengers decreases to 61.79% compared to the previous ones. The percentage of passengers with one transfer increases to 37.06%, and the percentage of passengers with two transfers is 1.16%. The average travel time increases to 13.21, which is unfavorable for passengers. However, the travel time for buses decreases to 83.0. In these conditions, the quality of this individual is quite low, reaching 7.728."

Improving the Algorithm of the Computational Prototype

Python programming language was used to implement the required functionality in the genetic algorithm and to test different algorithm parameters to ensure its stability. As can be seen in the main genetic operators described below, regarding the mechanisms of selection, crossover, and mutation; the selection of individuals allows the algorithm to go through all individuals in groups of four. This means that the first four are selected, followed by the next four, and so on until the last group of four is reached. This necessitates having a population size that is a multiple of four. On each occasion of selection, the one with the best fitness is preserved. Then, pairs of individuals are established, such as 0 vs. 1, 1 vs. 2, and 2 vs. 3. Subsequently, within each pair of individuals, the longest route is identified in order to perform the crossover of individuals. This mechanism enables genetic transmission in an aggregated manner by exchanging the longest routes between each pair of individuals, resulting in 6 offspring, two per pair. The mutation stage of the offspring follows, using a method to reduce the size of the routes, which helps maintain diversity and obtain better individuals in each iteration. After mutation, the quality of the offspring is evaluated, and the top 3 offspring in terms of quality are selected. Consequently, there are 4 individuals: the one that was preserved at the beginning and the three selected offspring who then replace the 4 individuals initially selected. Figure 3 schematically illustrates these three stages of the algorithm.

Subsequently, for better clarity, the main functions implemented through pseudocode in the Genetic Algorithm are described, such as the generation of individuals, the crossover process, mutation, and the evaluation of the population, the output of which is necessary to consistently determine the quality of each individual in the population. Here we have the "Generate Initial Population" function: This function implements the generation of individuals. It takes as input the population size (TPo) and the matrix of times between nodes i and j (mTij), and it provides as output the population of individuals that meet all the problem's conditions. It uses a "generate_route" function and then evaluates that route to consider routes that have a length greater than 3 nodes. The route generation process doesn't end until all nodes are included, and there are no disconnected nodes, as shown in algorithm 1.

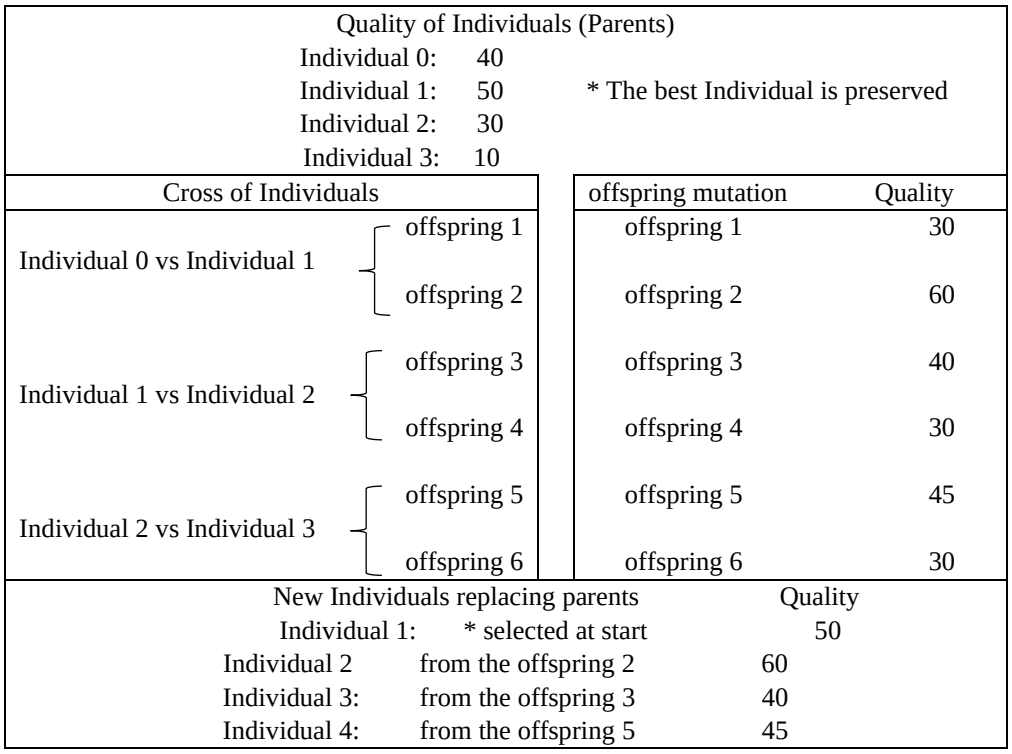


Figure 3: Diagram of the Stages of Selection, Crossing and Mutation of the AG

```
Function: Generate Initial Population
    Input: tpo: Population Size,
           mtij: Travel Time Matrix
    Output: Initial Population (poi)
1:      poi ← initialize Initial Population
2:      v ← initialize Individual
3:      for k=0 to tpo do:
4:          repeat
5:              route = generate_route
6:              if length(route) > 3 then:
7:                  add route to v
8:          until visited nodes = 15
9:          append v to poi
10:     return poi
```

Algorithm 1: Pseudocode that Generates the Initial Population of Individuals

Aggregated Individual Crossover: This operator performs the crossover between two individuals. It takes a pair of individuals as input. What's innovative about its genetic transmission functions is that it operates by selecting the longest route in each individual. In other words, it focuses on the network's connectivity when crossing individuals with greater geographical reach and a higher demand, minimizing the number of transfers. This results in an exchange of genetic information in an aggregated manner. The output of the function is two new individuals that contain genes from the other individual (see algorithm 2).

```
Function: Crossover Individuals
    Input: indiv1, indiv2
    Output: newindiv1, newindiv2
1:  first_route ← argmax(length(route)), route ∈ indiv1
2:  second_route ← argmax(length(route)), route ∈ indiv2
3:  newindiv1 ←append second_route to indiv1
4:  newindiv2 ←append first_route to indiv2
5:  return newindiv1, newindiv2
```

Algorithm 2: Pseudocode that Crosses Two Individuals in an Aggregate Way

Pruning Mutation: This operator changes the genetic structure, and its main function is to take an individual as input and return a mutated individual. The procedure involves selecting the route Indiv[0] from the individual and iterating through all the nodes of this initial route. During the iteration, the algorithm searches for the node in the rest of the routes with the condition that if it finds the node at position zero (0) or at the last position of any route, it removes the node where it was found. The functionality of this operator is to prune (generate a pruning) so that the result consists of individuals with strictly defined routes, avoiding the repetition of sections and minimizing the overall costs of the entire network. Additionally, another optimization function of this operator is to check for disconnected nodes, adding optimization value to the operator. This process is expressed in the pseudocode shown in algorithm 3.

```
1:  Procedure: Mutation
2:  Input: indiv
3:  Output: indivm
4:  indiv_evaluated ← indiv[0]
5:  for each node in indiv_evaluated:
6:      for each evaluated_route in indiv_evaluated:
7:          if node exists in endpoints of evaluated_route
8:              and there are no disconnected nodes then
9:              indiv_evaluated ← remove node from indiv_evaluated
10:         end
11:     end
12: indvm ← indiv_evaluated
13: return indivm
```

Algorithm 3: Pseudocode that Modifies an Individual's Chromosome

Evaluate Population: This function is responsible for decoding the chromosome of each individual in the population into alleles, which are expressed in terms of the percentage of travelers who have direct trips, i.e., without transfers, the percentage of travelers who travel with one transfer, the percentage of travelers who make two transfers, the percentage of travelers who make three transfers, four transfers, and/or k transfers. It is necessary to note that the sum of the percentages must be 100%. Additionally, the decoding determines the average travel time for the entire population, denoted as $\overline{T}_V$. Finally, the function also determines the total travel time of all vehicles on all routes, known as T_{RV}. This function returns, for each individual, an array $[[P_0, P_1, P_2, \ldots, P_k], \overline{T}_V, T_{RV}]$, representing the percentages of travelers with 0, 1, 2, …, k transfers, the average travel time, and the total travel time of all vehicles in the evaluated set of routes for each individual, as shown in Figure. There is an issue in this function between the percentages P_0 and P_1. When there are equal traveled pairs of nodes between direct trips without transfers and with one transfer, the algorithm does not prioritize direct trips. The same issue occurs between trips with one transfer and trips with two transfers. In this case, it should prioritize trips

with one transfer. This situation has been corrected as an improvement to avoid re-evaluating each individual until the best solution is found, which occurred with the algorithm in the previous proposed version.

```
Function: Evaluate Population
    Input: poi, mtij, mdemanda
    Output: evalua_poi
    1:  evalua_poi ← initialize as zero
    2:  for each route in poi:
    3:      mc ← costMatrix(route, preferences), for n transfers with n = 0,1,2,3,4,5
    4:      md ← demandMatrix(route), for n transfers with n = 0,1,2,3,4,5
    5:      mp ← userPercentageMatrix(route, mc, md), for n transfers with n =
            0,1,2,3,4,5
    6:      T̄V ← Average cost per trip(mp, mc, md)
    7:      TRV ← Total travel cost across all routes(mc, md)
    8:      append (mp, T̄V, TRV) to evalua_poi
    9:  end
    10: return evalua_poi
```

Algorithm 4: Pseudocode that Decodes the Chromosome of Each Individual

Fitness: A function that assesses the fitness of each individual. It takes as input the output of the Evaluate Population function and returns the fitness of each individual in the population according to equation (5), as shown in algorithm 5.

```
Function: Quality
    Input: evalua_poi
    Output: Population quality
1:  weights ← fixed weights per metric
2:  quality_poi ← empty list
3:  for each indiv in evalua_poi:
4:  (mp, T̄V, TRV) ← indiv
5:  quality ← mp(0) * weights(0)
6:  for k=1 to length(weights):
7:      quality ← quality - mp(k)*weights(k)
8:      quality ← quality / (log(T̄V) + log(TRV))
9:      append quality to quality_poi
10: end
11: return quality
```

Algorithm 5: Pseudocode that Returns the Quality of Each Individual

Improved Algorithm Stability

According to the factorial design outlined in the methodology: Number of generations (NG) with (50, 100, and 150), Population size (TPo) with (60, 120, and 180), the results are summarized in the analysis of variance Table 2, which shows that the TPo factor has a highly significant influence on the quality of the genetic algorithm individuals, while the NG factor is indifferent. This is because the algorithm's evolution manages to find very good individuals in fewer than 50 generations, with the exception of the NG=100, TPo=120 interaction, where in the first repetition, the optimal value was reached in generation 84, as shown in Table 3. This table reflects the NG at which the quality remains constant until the end of all generations. Additionally, the coefficient of variability is very small at 0.0978%, indicating that

the genetic algorithm is very precise in its results. To determine the best level of the TPo factor, the Duncan test was conducted with a coefficient of 0.05, revealing that, on average, the levels of 120 and 180 individuals statistically perform the same, with quality values of 11.27075 and 11.29442, respectively. In contrast, the level of 60 individuals has a lower average quality of 11.06750. Numerically, the best average was found with a population size of 180 individuals, which has an average quality of 11.29442.

Table 2: Analysis of Variance of the 2x3 Factorial Design for the AG

Factors	GL	SC	CM	Fc	SIG
Treatments	8	0.390387056			
NG	2	0.004833389	0.00241669	0.22052	
TPo	2	0.366042389	0.18302119	16.70019	**
NG x TPo	4	0.019511278	0.00487782	0.44509	
ERROR	27	0.295899250	0.01095923		
TOTAL	35	0.686286306			

Table 3: Pair (NG, Quality), from which the Quality does not Change

NG	TPo	I	II	III	IV
50	60	(13, 11.164)	(23, 11.048)	(45, 11.032)	(34, 10.978)
50	120	(22, 11.213)	(16, 11.275)	(36, 11.248)	(32, 11.330)
50	180	(32, 11.376)	(25, 11.190)	(43, 11.238)	(21, 11.235)
100	60	(09, 11.151)	(35, 11.160)	(44, 10.850)	(16, 11.255)
100	120	(84, 11.348)	(36, 11.333)	(36, 11.243)	(22, 11.137)
100	180	(21, 11.112)	(27, 11.392)	(18, 11.264)	(32, 11.317)
150	60	(17, 11.170)	(23, 10.970)	(16, 11.173)	(16, 10.859)
150	120	23, 11.237)	(21, 11.360)	(41, 11.233)	(34, 11.292)
150	180	(29, 11.359)	(31, 11.340)	(38, 11.286)	(21, 11.379)

Regarding the random generation of individuals in the Genetic Algorithm (GA), it has been found that the size of each individual can vary from 3, 4, 5, 10, 15, 20, 25, up to 28 routes. This variability is due to the fact that an individual is considered complete when all constraints are met, meaning that nodes are not repeated within the same route, and the generation process does not stop until all nodes are connected. During the evaluation of each individual's quality, an initial output is determined based on transfer percentages, $\overline{T}_V$, and T_{RV}, and quality is then calculated from this information.

During calibration tests of the route generation algorithm, it was observed that it consistently delivers highly satisfactory sets of routes. This is reflected in the high percentage of passengers traveling without transfers, and even when transfers are required, the percentage is very low. Additionally, the average travel time per passenger is minimized, as is the total travel time for all vehicles on all routes in the set. This means that if the algorithm's response is a set of 3 routes, these routes are highly satisfactory. Similarly, if it delivers a set of 4, 5, 6, 7, or 8 routes, they are also highly satisfactory. The algorithm has been optimized to prioritize routes without transfers as the first option, followed by routes with 1 transfer, and only if the result warrants it, routes with 2 transfers are considered.

Finally, an instance of the evolution of individuals regarding their quality across all generations is shown in Figure 4. It's important to note that the average execution time of the algorithm is 35 minutes. The quality of each individual is considered adimensional because it is the result of dividing the percentage of trips with 0, 1, 2, ..., k transfers by the sum of the logarithms of the average travel time and the logarithm of the total travel time of all buses, with time units in minutes.

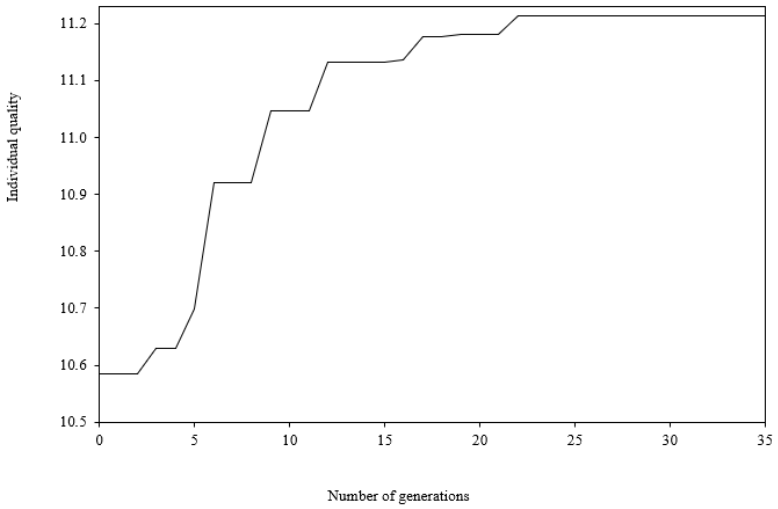


Figure 4: Behavior of the Quality of an Individual During the Evolution

Construction of the Second Phase Algorithm

This algorithm consists of two parts: the calculation of frequencies and the determination of the fleet. In the first part, the demand for each route is determined, and in the second part, the frequency for each route and the calculation of the necessary fleet are determined, both for the routes designed in the first phase.

The first part involves establishing an efficient distribution of demand for the set of routes of the optimal individual selected in the first phase. This calculation starts by distributing the demands that involve transfers, whether it's 1, 2, or more transfers, equitably among each route in the set of routes. For example, if the demand from node "i" to node "j" is 300 passengers and the journey includes one transfer because two routes are involved, then the demand is divided into 150 for the first route and 150 for the second route. Conversely, if the demand from "i" to "j" involves 3 routes, indicating two transfers, then the demand of 300 passengers is divided equally among the three routes, with 100 passengers for each route. After establishing the demands to be distributed among routes with transfers, these demands are inserted into node pairs that are part of the direct travel routes in the demand matrix. However, before insertion, it's checked how many times that node pair appears in the set of routes to distribute the demand evenly. This verification is done for each route in the individual's set of routes.

After determining the demand for different node pairs (i, j) or segments (i, j) for all trips with transfers, the original demand matrix is modified by adding the demand from the segments with transfers. Then, the demand is assigned to the routes in the individual's set. The demand assignment to the routes in the individual's set starts with the modified demand, and similarly, the number of times each node pair (i, j) or segment (i, j) appears in the individual's set of routes is checked to distribute the demand equally among all segments. Then, the demand distributed in each route is assigned as shown in the pseudocode in algorithm 6.

```
Algorithm: Demand Distribution
  Input: Set of Individual Routes
  Output: Total Demand per Route
  1: Individual ← Routes i, j + Routes j, i
  2: Total Segments of Individual ← Individual (combination, 2)
  3: For i from 0 to Number of nodes do
  4:     For j from 0 to Number of nodes do
  5:         If route cost and number of transfers ≥ 1 then
  6:             Routes with Transfer ← Shortest Path Algorithm (i, j)
  7:             For Routes with Transfer from 0 to number of routes with transfer do
  8:                 Number of Routes for Transfer ← number of transfers + 1
  9:                 Demand of Segments with Transfer ← (Demand with transfer (i, j) / Number of Routes for Transfer
 10:             end For
 11:         end If
 12:     end For
 13: end For
 14:
 15: Demand Matrix (i, j) ← Original Demand Matrix + Demand of Segments with Transfer
 16: If a segment occurs in every route > 1 then
 17:     Demand ← Demand / number of times the segment repeats
 18: end If
 19: For a from 0 to length of the route do
 20:     Total Demand of the route ← Demand of each node combination
 21: end For
 22: Output: Total Demand per Route
```

Algorithm 6: Assignment of the Demand to the Routes of the Individual Route Set

The second part of the allocation algorithm determines the frequency using the formula proposed by Ceder (1987), as reported by Arbex and da-Cunha (2015). This calculation is performed to ensure that the occupancy of buses does not exceed the most congested segment of the route in terms of vehicle occupancy. Occupancy is determined using an occupancy factor representing the proportion of standing passengers, calculated as the total number of passengers divided by the number of seats. The expression that provides the frequency calculation is given by equation (6).

$$\text{Frecuency} = \frac{Q_{\max}}{(\text{load factor}) \text{ x } (\text{bus capacity})} \quad (9)$$

Where:

- Qmax: maximum load per section of each route in the set of routes
- load factor: it is the parameter established for the estimation of additional load (standing passengers), in this article it is considered 1.25 (additional 25% of passengers), identical to other articles
- Bus capacity: It is the standard load of (seats), in this article it is considered 40, equal to other articles.

To calculate Q_max, the modified demand from the first part of the allocation algorithm is used. An exhaustive combinatorial analysis of all segments or node pairs obtained from the individual's set of routes was performed. This allowed for the determination of the demand for each segment in each route and for the entire set of routes. Q_max is determined for each route by summing all the accumulated demand for each segment. From all the accumulated demands, the highest demand for each route is selected. Once Q_max is obtained, equation 6 is applied to determine the frequency of each route. Consequently, the travel time for each route in the set of routes is evaluated. Finally, the required fleet for each route is calculated according to the formula in equation 7, as proposed by Capali and Ceylan (2020). The total fleet for the set of routes is determined as the sum of the fleets for each route,

considering both the outbound and return trips. The pseudocode in algorithm 7 shows the calculation process.

$$\text{Fleet} = \frac{\text{Frecuency} * \text{route cost}}{60} \quad (7)$$

```
Algorithm: Frequency and Fleet Determination
  Input: Demand, Individual
  Output: Frequency, Fleet
  1: For routes from 0 to length of Individual do
  2:      Demand per segment of each route ← demand for each node pair per route in Demand
  3: end For
  4: Maximum Load ← Maximum (Demand per segment of each route)
  5: Frequency ← Maximum Load / (load factor * Bus Capacity)
  6: Output: Frequency
  8:
  9: For route from 0 to length of Individual do
 10:      For elements in route from 0 to length of route - 1 do
 11:           Route Cost ← sum of costs of node pairs per route
 12:          Route Fleet ← (Frequency * Route Cost) / 60
 13:      end For
 14:  end For
 15: Fleet = sum (Route Fleet)
 16: Output: Fleet
```

Algorithm 7: Assignment of the Demand to the Routes of the Individual Route Set

3 Analysis and Discussion

Both the algorithms of the first phase used in the development of the computational prototype for route design and the algorithms of the second phase used for demand allocation and bus frequency to routes, as well as the calculation of the bus fleet, were implemented using Python and tested on the Swiss road network of Mandl, 1979. This network has 15 nodes and 21 edges, with a total network demand of 15,570. While the proposed algorithm is validated on the Swiss road network, as it is a reference in the literature, our proposal achieves better results in 90.70% of comparisons, with similar results in the remaining cases, and our proposal consistently performs with a shorter algorithm execution time.

Furthermore, even though the reference network is quite small compared to the networks in various cities worldwide, our algorithm will always respond adequately because it has a route generation procedure that complies with all constraints of the mathematical model. After calibrating the route generation algorithm and determining the best parameters, including characterizing the objective function structure and the behavior of genetic operators, specifically for these latter elements of the algorithm, they undergo thorough processing for the entire population generated from the first iteration. This is because their exploratory behavior captures the best characteristics of each individual from the inception of the solution search space. Therefore, at the end of the algorithm, individuals have gathered the best genetic information from the outset of processing.

We conducted 120 executions of the genetic algorithm, resulting in multiple efficient solutions: 3 routes with three segments, 17 routes with four segments, 17 routes with five segments, 33 routes with six segments, 33 routes with seven segments, and 17 routes with eight segments. All of these solutions showed remarkable efficiency, with a coefficient of variation of quality with respect to the mean of only

0.0978%. It is important to clarify that in Tables 4 and 5, results from other authors are shaded in gray, while unshaded results are obtained with our algorithm based on the routes presented by those authors for the purpose of fair and comprehensive evaluation. Therefore, these tables contain the results of our proposal. Table 4 analyzes and discusses solutions with 3, 4, 5, and 6 routes, while Table 5 deals with solutions of 7 and 8 routes. In both tables, both phases are integrated, and comparisons are made with all proposals from the literature.

Comparing our results with those of Fan and Mumford (2010) for 4, 6, 7, and 8 routes, it can be observed that in the solutions with 6, 7, and 8 routes, our algorithm outperforms them in all evaluated parameters, including fleet size. However, when comparing the 4-route solution, the percentages of direct and one-transfer trips are slightly lower in our algorithm. Still, $\overline{T}_V$ (average travel time per passenger) is better in our algorithm at 10.86 minutes compared to 11.37 minutes, and the T_{RV} (Total Route Length) is also better in our algorithm at 117 minutes compared to 147 minutes. Therefore, the quality is also superior. Similarly, the fleet size is much better in our algorithm, with a value of 78 buses, while the authors' proposal requires 103 buses. When Asadi-Bagloee and Ceder (2011) report their results, they only present results for 12 routes, which is why it is not reported in Table 4. In terms of our algorithm, they report the following: [[9,7,5,2,1,0], [9,6,14,8], [6,14,5,2,1], [6,14,5,3,4], [0,1,2,5,7,9,13], [12,9,7,5,2,1,0], [11,10,9,6], [10,9,7,5,2,1,0], [8,14,6,9,12], [3,5,7,9,10], [10,9,7,5,3,4], [9,10,11]]. They also report that 83.66% of passengers travel directly without transfers, 15.21% use one transfer, 0.95% use two transfers, and there is an unsatisfied demand of 0.18%. They also report a fleet size of 87 buses for the 12 proposed routes, with an $\overline{T}_V$ of 11.52 and T_{RV} of 261. Comparing these results with our route generator, there is a significant gap, as our algorithm achieves 96.15% direct passenger trips, with only 3.85% using one transfer, completely satisfying the demand. Additionally, our parameters $\overline{T}_V$=10.38 and T_{RV}=191 are better than those of the authors. Furthermore, the fleet size in their proposal is 87 buses, which is higher compared to our proposal of 78 buses. Cipriani et al. (2012) are not analyzed because they do not validate their proposal on the Mandl network.

Regarding the results presented by Nikolić and Teodorović (2013), firstly, they do not provide solutions with three routes, whereas our genetic algorithm efficiently does so. For solutions with 6, 7, and 8 routes, their algorithm has slightly better percentages of direct trips without transfers, such as 95.63%, 98.52%, and 98.97%, compared to ours, which are 94.67%, 96.15%, and 95.70%. The difference corresponds to trips with one transfer. However, in terms of $\overline{T}_V$, they achieve 10.23, 10.15, and 10.09 minutes, respectively, compared to our genetic algorithm with $\overline{T}_V$ of 10.42, 10.38, and 10.49 minutes. This apparent disadvantage of our algorithm is greatly compensated for by T_{RV}. Our algorithm has lower total travel times in all solutions, with values of 183, 191, and 195 minutes, whereas they report 224, 247, and 288 minutes, respectively. As a result, the quality of our solutions is superior to those presented by them. Regarding fleet size, although they do not report this parameter for solutions with 6, 7, and 8 routes, when we evaluate their routes with our algorithm, we obtain fleet sizes of 94, 90, and 106 buses for the respective solutions. These values are higher than those provided by our algorithm, which obtains 72, 78, and 74 buses for the same solutions, as shown in Table 4. As for the 4-route solution, their reported results in Table 4 do not correspond. Instead, the correct values should be: 88.76% direct trips, 10.15% with one transfer, 1.09% with two transfers, $\overline{T}_V$=10.79 minutes, and T_{RV}=146 minutes, with a quality of 10.791. These results were obtained from the routes [[0,1,2,5,7,9,10,11], [1,4,3,5,7,9,12,10], [8,14,6,9,7,5,3,11], [3,1,2,5,14,6,9, 13]], compared to our results of 92.94% direct trips, 7.06% with one transfer, $\overline{T}_V$=10.86 minutes, T_{RV}=117 minutes, and a quality of 11.664. Our results are better than theirs, and our algorithm requires 89 buses, while theirs requires 78 buses, making it a more efficient solution. It is worth noting that the authors attempt to

achieve a lower $\bar{T}_V$ in route sets by incurring a high T_{RV} or increased travel time for all vehicles on all routes, which affects the number of buses required.

The results presented by Kiliç and Gök (2014) are based on an algorithm that uses Tabu Search as a metaheuristic and Simulated Annealing as part of the solution. In Table 4, the best results of their algorithm are shown, indicating that they do not provide solutions for 3 routes, and the solutions they provide for 4 routes are inferior in all parameters compared to our genetic algorithm, except for $\bar{T}_V$, where they achieve $\bar{T}_V$=10.56 minutes, and our algorithm achieves $\bar{T}_V$=10.86 minutes. However, they have a T_{RV}=148 minutes in their algorithm compared to our T_{RV}=117 minutes, resulting in a better-quality solution in our case. As for the solutions with 6, 7, and 8 routes, they achieve slightly better results than our genetic algorithm, except for T_{RV}, where our algorithm reports lower travel times, such as 183, 191, and 195 minutes compared to their 212, 250, and 272 minutes, respectively. This makes the quality of our solutions superior. Regarding fleet size, they do not report this parameter, but we calculated it based on the routes they show, resulting in fleet sizes of 102, 98, 100, and 97 buses for solutions with 4, 6, 7, and 8 routes, respectively. In comparison, our algorithm requires 78, 72, 78, and 74 buses for the same solutions, as shown in Table 4.

While Mahdi-Amiripour et al. (2015) proposed a practical method based on genetic algorithms to solve the bus network design problem and achieved good results, when they test their algorithm on the Mandl's Swiss Road network, they only report results for 4 routes: [[6,14,5,2,1,3,11], [11,10,9,6,14], [0,1,2,5,14,6,9], [4,3,5,7,9,13,12,10]]. This solution led to the following result: 79.38% of passengers travel directly without transfers, 20.62% make one transfer, but they do not report $\bar{T}_V$ or T_{RV}. They do report a fleet size of 66.95 buses for this 4-route solution. In this case, the missing values were obtained using their known routes and our algorithm, resulting in the following values: $\bar{T}_V$ = 11.19 minutes and T_{RV} = 104 minutes. Compared to our genetic algorithm, their solution falls significantly short, as our generator achieves 92.94% of passengers traveling directly without transfers and only 7.06% using one transfer. $\bar{T}_V$ is 10.86 minutes, and T_{RV} is 117 minutes, resulting in a higher quality of 11.664. Our algorithm also requires 80 buses, which is unusual compared to their reported fleet size.

The results of extensive computational experiments by Arbex and da-Cunha (2015) are reported using both the original Mandl reference set and instances with different demands and travel times to determine Pareto fronts of optimal solutions since user and operator costs are conflicting objectives. However, their reports are incomplete in Table 4 because they do not report T_{RV} or the route solutions. This is unfortunate because if the route solutions existed, we could have reconstructed all the parameters. In this case, we can only comment that, as per their statements, our route generator using genetic algorithms achieves lower average travel times in all solutions. For example, we have $\bar{T}_V$ values of 10.86, 10.72, 10.42, and 10.38 minutes for their respective 4, 6, 7, and 8-route solutions, while they report 10.23, 10.15, 10.09, and 10.26 minutes. However, our algorithm reports much lower T_{RV} values for the same solutions, such as 183, 191, 195, and 194 minutes, compared to their 224, 247, 288, and 257 minutes. As a result, the quality of our solutions is superior. Regarding fleet size, they do not report it for these solutions, which limits the comparison.

The best results from the application of two mixed-integer linear programming models proposed by Cancela et al. (2015) cannot be reported in Table 4 because they propose solutions with 20 routes in Model I and 12 routes in Model II. The same applies to their comparative part with Asadi-Bagloee and Ceder (2011), where their solution also corresponds to a set of 12 routes. However, we can discuss these solutions by showing the routes they provide. For 20 routes: [[9,10], [9,13,12,10], [9,10,12], [9,10,11], [9,13,12,10,11], [9,10,12,13], [0,1,2,5,7,9], [0,1,2,5,14,6,9], [8,14,6,9], [1,3,5,7,9], [4,3,5,7,9], [4,1,2,5,7,9], [4,3,1,2,5,7,9], [10,9,13,12], [1,2,5,14,6], [0,1,3,4], [0,1,2,5,7,14,6], [0,1,2,5,7,9,6],

[11,10,12], [11,10,9,13,12]]. For 12 routes: [[9,10,12], [3,5,7,9], [0,1,2,5,7,9], [0,1,2,5,14,6,9], [8,14,6,9], [4,3,5,7,9], [4,3,1,2,5,7,9], [10,9,13,12], [0,1,3,4], [0,1,2,5,14,6], [11,10,12], [11,10,9,13,12]]. Their 20-route solutions provide the following results: [90.37%, 9.31%, 0.32%], 10.74, 354, quality=9.82, while their 12-route solutions yield: [89.53%, 9.96%, 0.51%], 10.83, 217, quality=10.327. Their comparative 12-route solution shows: [86.90%, 12.52%, 0.58%], 11.52, 261, quality=9.701. All three of these proposed solutions are dominated by the best solution from our algorithm in all its parameters, as shown by [95.89%, 4.11%], 10.38, 191, quality=11.345. It's worth highlighting that the authors' proposal has users making up to 2 transfers, while our solution has 95.89% of travelers making direct trips, with only 4.11% needing to make a single transfer. Our solution also achieves a lower average travel time per passenger of 10.32 minutes and a lower total route travel time of 191 minutes, resulting in a significantly better-quality solution. Regarding the number of buses, our algorithm indicates that Cancela et al. (2015) would require 64 and 62 buses for their 20-route and 12-route solutions, respectively, while their comparative solution would require 61 buses. This can be explained by the fact that, when multiple routes exist in a solution, the maximum load (Qmax) traveling on the routes is diluted or reduced due to multiple travel alternatives and repeated sections. As a consequence of this decrease in Qmax, the fleet size becomes smaller, and as a result, the travel time of the buses on all routes is much longer. For example, in the case of 8 routes from our algorithm, with a travel time of 195 minutes, the 20-route proposal's travel time increases to 354 minutes, equivalent to an 81.54% increase.

Table 4: Report of the Comparison of Algorithm Results of 11 Proposals on Route Design in the Swiss Road Reference Network of Mandl

Authors	Number of routes									Fleet	Percentage (%)			$\bar{T}_V$	T_{RV}	Quality
	d/f	1	2	3	4	5	6	7	8		P_0	P_1	P_2			
Proposal	dem.	5468	5583	4529	-	-	-	-	-	96	93.13	6.81	0.06	10.93	125	11.571
	frec.	23	25	18	-	-	-	-	-							
11	dem.	7354	4284	3988	-	-	-	-	-	124	93.67	5.43	0.90	10.50	150	11.4
	frec.	34	19	18	-	-	-	-	-							
Proposal	dem.	6106	2364	2706	4394					78	92.94	7.06	0	10.86	117	11.664
	frec.	24	13	14	26											
1	dem.	4252	3926	3642	3755					103	93.26	6.74	0	11.37	147	11.268
	frec.	26	16	17	23											
2	dem.	5248	4542	2686	3120					89	92.10	7.19	0.71	10.51	146	10.791
	frec.	23	20	14	15											
3	dem.	4488	4574	4759	1829					102	91.33	8.16	0.51	10.56	137	11.245
	frec.	21	29	22	11											
4	dem.	2236	2630	3025	7678					80	79.38	20.62	0	11.19	104	10.0
	frec.	13	14	21	40											
5	-	-	-	-	-					79	98.27	1.73	0	11.13	-	-
	-	-	-	-	-											
6	-	-	-	-	-					40	92.29	7.71	0	15.67	91	11.4
	-	-	-	-	-											
7	-	-	-	-	-					-	91.84	8.15	0	10.48	148	11.2
	-	-	-	-	-											
8	dem.	4033	4948	3232	3357					92	92.23	7.71	0.02	10.84	151	11.2
	frec.	19	22	20	21											
9	-	-	-	-	-					-	91.07	8.22	0.71	10.56	124	-
	-	-	-	-	-											
10	dem.	2915	3183	5816	3657					86	91.52	7.77	0.71	10.54	143	-
	frec.	17	18	24	15											
11	dem.	3973	4298	3800	3499					97	91.84	8.16	0	10.48	148	11.2
	frec.	16	21	21	21											
Proposal	dem.	4959	1240	3175	4003	2193				72	93.38	6.62	0	10.72	141	11.444
	frec.	24	7	16	18	12										

11	dem.	4374	3709	4141	2295	1052				82	94.73	5.14	0.13	10.31	180	11.3
	frec.	16	18	19	12	4										
Proposal	dem.	2979	3424	1571	3752	2673	1171			72	94.67	5.33	0	10.42	183	11.252
	frec.	11	18	9	19	11	8									
1	dem.	2947	3125	831	4453	1778	2437			94	91.52	8.48	0	10.48	215	10.6
	frec.	15	15	4	23	8	12									
2	dem.	3349	3227	2739	3175	2050	1030			94	95.63	4.37	0	10.23	224	11.1
	frec.	14	17	15	17	11	6									
3	dem.	870	3479	591	3024	2591	5016			98	95.50	4.50	0	10.29	216	11.1
	frec.	5	14	3	20	15	18									
5	-	-	-	-	-	-	-			77	98.20	1.80	0	11.55	-	-
	-	-	-	-	-	-	-									
6	-	-	-	-	-	-	-			70	84.77	15.23	0	11.38	112	10.6
	-	-	-	-	-	-	-									
7	-	-	-	-	-	-	-			-	97.17	2.82	0	10.18	212	11.4
	-	-	-	-	-	-	-									
8	dem.	2563	1166	2135	2805	4223	2678			90	94.93	5.07	0.02	10.84	151	11.2
	frec.	16	7	11	15	21	17									
9	-	-	-	-	-	-	-			-	96.92	3.08	0.00	10.19	195	-
	-	-	-	-	-	-	-									
10	dem.	3075	3390	3632	2485	2051	942			96	96.27	3.60	0.13	10.22	230	10.9
	frec.	15	16	14	16	10	4									
11	dem.	3667	2636	2812	1495	2302	2659			100	97.17	2.83	0.00	10.18	220	11.3
	frec.	16	13	15	9	12	16									

The comparative analysis of our route generator against Owais and Osman (2018), even though they do not show the routes, is presented in Table 4. It can be observed that, in all cases of 4, 6, 7, and 8 routes, the percentages of passengers traveling directly without transfers are better in our route generator. The average cost of travel per passenger ($\overline{T}_V$) is much lower in our route generator. However, the T_{RV} parameter is better in all cases for the authors' proposal, indicating that their objective is to favor operators by reducing the cost of vehicle travel, at the expense of the average cost per passenger, and they do not emphasize maximizing direct passenger transfers. Nevertheless, the qualities of the solutions are always better in our route generator, as our quality function equally weights $\overline{T}_V$ and T_{RV}. In the comparative analysis related to fleet size, they report that for solutions with 4, 6, 7, and 8 routes, the required fleet sizes are 40, 70, 88, and 95 buses, respectively. Based on Table 4, we can only say that they achieve better results in the 4 and 6-route solutions, while our results are better for the 7 and 8-route solutions. However, it is strange that they report a fleet requirement of only 40 buses for the 4-route solution.

Table 5: Report of the Comparison of Algorithm Results of 11 Proposals on Route Design in the Swiss Road Reference Network of Mandl

Authors	Number of routes										Fleet	Percentage (%)			$\overline{T}_V$	T_{RV}	Quality
	d/f	1	2	3	4	5	6	7	8	9		P_0	P_1	P_2			
Proposal	dem.	1813	3463	924	4282	670	1932	2486			78	96.15	3.85	0	10.38	191	11.4
	frec.	9.8	18.8	5.6	14.9	3.7	13.6	12.9									
1	dem.	2348	2104	683	2586	2090	1420	4338			96	93.32	6.36	0.32	10.42	231	10.8
	frec.	11.4	10.1	3.86	14.8	11.7	11.8	23.3									
2	dem.	2787	2842	2384	2857	2669	1153	879			90	98.52	1.48	0	10.15	247	11.3
	frec.	11.1	15.2	13.1	13.9	13.4	6.07	3.64									
3	dem.	1894	1582	2592	1932	2932	2568	2080			100	97.04	2.83	0.13	10.23	274	11.0
	frec.	9.8	7.7	11.4	8.3	16.3	9.6	14.5									
5	-	-	-	-	-	-	-	-			77	98.52	1.48	0	11.91	-	-
	-	-	-	-	-	-	-	-									
6	-	-	-	-	-	-	-	-			88	89.21	11.79	0	11.25	125	11.0
	-	-	-	-	-	-	-	-									
7	-	-	-	-	-	-	-	-			-	98.84	1.15	0	10.10	250	11.3
	-	-	-	-	-	-	-	-									
8	dem.	3482	1879	2465	723	2381	1852	2789			84	93.38	6.62	0	10.62	237	10.7
	frec.	20.0	8.4	16.6	3.8	11.0	10.5	12.9									

9	-	-	-	-	-	-	-	-			-	97.43	2.57	0	10.15	209	-
	-	-	-	-	-	-	-	-									
10	dem.	2997	2572	916	2853	2458	831	2764			106	98.01	1.99	0	10.1	246	11.3
	frec.	12.0	13.5	6.1	14.6	14.5	4.8	17.9									
11	dem.	3511	2522	2728	1350	2109	804	2547			104	98.97	1.03	0	10.1	259	11.3
	frec.	15.3	12.6	14.8	7.7	11.4	4.5	15.9									
Proposal	dem.	2278	1883	3630	425	503	2581	3841	429		74	95.70	4.3	0	10.49	195	11.3
	frec.	13.3	9.7	14.5	3.6	5.0	13.2	14.2	2.2								
1	dem.	1551	3324	1430	1280	832	1856	2080	3223		106	94.54	5.46	0	10.36	283	10.6
	frec.	11.3	15.6	11.3	7.1	5.1	11.3	11.8	13.0								
2	dem.	3110	1304	2415	2497	528	2098	2675	941		106	98.97	1.03	0	10.09	288	11.2
	frec.	13.2	7.7	14.6	11.4	3.4	14.2	15.1	4.9								
3	dem.	2157	2243	2161	2887	2835	1760	1066	461		97	97.37	2.63	0	10.20	298	10.9
	frec.	10.0	9.4	16.6	15.0	10.3	9.6	5.2	2.8								
5	-	-	-	-	-	-	-	-	-		74	98.65	1.35	0	11.24	-	-
	-	-	-	-	-	-	-	-	-								
6	-	-	-	-	-	-	-	-	-		95	90.68	9.32	0	11.35	154	10.9
	-	-	-	-	-	-	-	-	-								
7	-	-	-	-	-	-	-	-	-		-	99.16	0.83	0	10.08	272	11.0
	-	-	-	-	-	-	-	-	-								
8	dem.	1418	164	3700	1079	2719	2255	979	3256		89	94.54	5.46	0	10.32	251	10.8
	frec.	7.1	0.9	16.7	7.0	19.6	15.2	5.7	18.4								
9	-	-	-	-	-	-	-	-	-		-	98.59	1.41	0	10.09	233	-
	-	-	-	-	-	-	-	-	-								
10	dem.	2642	2275	3123	872	2167	2315	1605	572		101	98.97	1.03	0	10.08	280	11.2
	frec.	12.7	10.9	12.5	5.8	15.6	13.7	8.2	3.4								
11	dem.	3024	2003	2260	1702	1664	762	2127	2028		100	99.23	0.77	0	10.07	302	11.1
	frec.	11.3	9.6	10.1	9.2	9.3	4.2	12.6	12.5								

In Tables 4 and 5, the results of 11 proposals from 2010 to 2021 are reported, in addition to our proposal, in the following order: 1) Fan and Mumford (2010), 2) Nicolić and Teodorović (2013), 3) Kiliç and Gök (2014), 4) Mahdi-Amiripour et al. (2015), 5) Arbex and da-Cunha (2015), 6) Owais and Osman (2018), 7) Ahmed et al. (2019), 8) Buba and Lee (2019), 9) Yang and Jiang (2020), 10) Katsaragaskis et al. (2020), and 11) Vermeir et al. (2021). The reported parameters are: P_0 = percentage of passengers traveling without transfers, P_1 = percentage of passengers making 1 transfer, P_2 = percentage of passengers making 2 transfers, $\overline{T}_V$ = average passenger travel time in minutes, T_{RV} = total travel time of all vehicles on all routes in minutes, Quality = quality of each solution is dimensionless; it also shows the demand satisfied on each route, frequency, and fleet size.

In the following analysis and discussion of results, three proposals are grouped together due to having similar results, namely Buba and Lee (2019), Yang and Jiang (2020), and Katsaragaskis et al. (2020) for a fair comparison. We have taken the routes proposed by Buba and Lee (2019) and calculated the demand served by each route and the frequencies using our algorithm. In the case of Yang and Jiang (2020), they do not report routes, so their results are reported as follows. In the solution of 4 routes, our algorithm achieves better results in all evaluated parameters except $\overline{T}_V$, which is higher by 0.02, 0.030, and 0.32 minutes for each of the respective authors. However, despite this, the quality of our result is better (11.664), as the percentage of trips without transfers and T_{RV} (92.94% and T_{RV} = 117) are better than their proposals. In the case of the 6-route solution, our algorithm outperforms Buba and Lee (2019) in all parameters except for 0.26% of passengers who make trips with a transfer and T_{RV}. As for the other two authors in the same 6-route solution, they achieve better results by small differences in all parameters except for T_{RV}, where our route generation algorithm consistently records lower values, namely 183 minutes compared to 195 and 230 minutes, respectively. In the set of 7 routes, our route generation algorithm records better results than those shown by Buba and Lee (2019), while the other two authors achieve better results than our algorithm. However, it is worth noting that our algorithm consistently minimizes T_{RV}, and this time the value was 191 minutes compared to 209 and 246 minutes. Finally, in the 8-route solution, our route generation algorithm outperforms Buba and Lee (2019) except for the parameter $\overline{T}_V$, where our algorithm delivers a value of 10.49 minutes, which is higher than the

proposed 10.32 minutes. Regarding the other authors, they achieve better results than our route generator, but these better results come at the expense of a high T_{RV}, as can be seen that our algorithm uses 195 minutes in this set of routes, while the other two authors use 233 and 279 minutes, respectively. Analyzing the fleet and comparing our proposal for the 4, 6, 7, and 8-route solutions, we found that we require 78, 72, 78, and 74 buses, respectively. However, authors Buba and Lee (2019) and Katsaragaskis et al. (2020) report much larger fleets ranging from 84 to 101 buses. A comparison cannot be made with Yang and Jiang (2020) because they did not report this parameter. Finally, a comparison is made between the proposal of Vermeir et al. (2021) and our genetic algorithm-based route generator. It is worth noting that the comparison can be made in the 3, 4, 5, 6, 7, and 8-route sets because there are routes in both proposals. For a fair comparison, the version proposed by Vermeir et al. (2021) for the 3-route solution, considering routes with more than 8 nodes on each route, and for the solutions of 4, 5, 6, 7, and 8 routes, the version that contains up to 8 nodes in the routes to ensure consistency in the comparison has been considered. Also, it should be noted that the author does not indicate the fleet size. The comparison with the 3-route solution shows that the author's proposal is apparently better than ours in terms of achieving slightly better results in the parameters, except for T_{RV}. Our proposal reports 125 minutes, and the authors report 150 minutes. This difference means that their proposal requires a fleet of 124 buses, whereas our proposal only requires 96 buses, which is reflected in the fact that our proposal has better quality, namely 11.571, while they have 11.418. With the 4-route solution, our proposal performs better in all parameters except $\overline{T}_V$ = 10.86 minutes compared to $\overline{T}_V$ = 10.48 minutes of the other authors. In this case, the fleet required by the authors is 97 buses, while in our proposal, only 78 buses are needed. For the solutions with 5, 6, 7, and 8 routes, both proposals show decent performance, with a high percentage of passengers traveling directly. However, the authors' proposal, in an attempt to reduce the average passenger travel time ($\overline{T}_V$), disproportionately increases the travel time of all buses or the fleet, which in the worst case has increased from 195 to 302 minutes in the 8-route solution, representing an increase of up to 54.82%. This increase does not compensate for the reduction achieved in $\overline{T}_V$, as they report $\overline{T}_V$ = 10.31, $\overline{T}_V$ = 10.18, $\overline{T}_V$ = 10.10, and $\overline{T}_V$ = 10.07 minutes for the mentioned solutions, while our algorithm reported values such as $\overline{T}_V$ = 10.72, $\overline{T}_V$ = 10.42, $\overline{T}_V$ = 10.38, and $\overline{T}_V$ = 10.49 minutes in the same solutions, demonstrating a reduction, which in the best case is 4.17%, and an overall reduction of 12.87%. Regarding the fleet, even though they do not provide this information, our algorithm has calculated that these solutions require 82, 100, 104, and 100 buses, while our proposal requires 72, 72, 78, and 74 buses, making our proposal significantly better. In conclusion, the route generator in the first phase always delivers efficient solutions of very high quality thanks to innovative crossover methods that we have called "aggregated reproduction," which avoids rejecting individuals as unviable and the mutation method, which not only alters the genetic material of an individual but also prunes the route if necessary, allowing for adequate exploration of the search space. As a result, the route generation algorithm does not allow convergence stagnation in local optima. The results of 11 comparisons reported in Table 4 and others not reported in the table but discussed here are highly competitive. Furthermore, our proposal comes from a high-precision algorithm with a coefficient of variation of quality relative to the mean in the order of 0.0978%. Another advantage of our algorithm is the execution time, which is 35 minutes. It is important to note that our proposal has a unique characteristic in that it does not require an increase in T_{RV} to achieve a good solution quality. For the algorithm's execution in both the first and second phases, a laptop with an Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz 2.30 GHz processor and 8.00 GB (7.60 GB usable) of installed RAM was used. In Table 6, the solutions provided by our genetic algorithm-based route generator in each set of routes are reported. In particular, the genetic algorithm's behavior reports solutions, i.e., individuals, ranging from 3 routes to 8 routes, all of which are efficient.

Table 6: Set of Routes Generated by the Route Generator Algorithm

Number of Routes	List of Solutions	Number of Routes	List of Solutions
3	0,1,2,5,7,14,6,9,12	7	0,1,3,5,7,14,6
	0,1,3,11,10,9,7,5,14,8		4,1,2,5,14,6,9,10
	4,3,5,14,6,9,10,12,13		8,14,7,5,2,1,3
4	11,10,9,7,5,2,1,0		13,12,9,7,5,2,1,0
	4,3,1,2,5,7,14,6		3,11,10,12
	8,14,5,7,9,13,12		4,3,5,7,9,13
	12,10,9,6,14,5,3,4		8,14,5,7,9,10,11
5	0,1,2,5,7,14,6,9	8	4,3,5,7,14,6,9,13
	11,3,1,2,5,7		9,6,14,5,2,1,4
	4,3,5,14,6,9,12,13		0,1,3,5,14,6,9,10
	12,10,9,7,5,2,1,4		11,10,12
	8,14,6,9,10,11		9,12
6	0,1,2,5,7,9,13,12		11,10,9,7,5,14,8
	0,1,2,5,14,6,9,10		12,13,9,7,5,2,1,0
	11,3,1,2,5,7,14,6		2,5,3,11
	0,1,4,3,5,7,9,10		
	8,14,5,7,9,13,12,10		
	13,9,10,11		

4 Conclusion

Based on the results presented and the discussion in comparison with findings from other authors and the detailed analysis provided, the following conclusions can be drawn: 1) In the first phase, the problem of urban transportation route design has been mathematically modeled. This mathematical model includes detailed parameters and variables related to a multi-objective function, three explicit constraints with their corresponding equivalences, and additional constraints described narratively. The intention behind this mathematical modeling, as with any proposal, is to create an approximation to the real world for practical application. It involves constraining the variables and parameters that ultimately define the problem's context into a model that is viably applicable to real-world cases when the algorithm converges. 2) Likewise, a computational prototype for generating urban transportation routes has been constructed with precise, efficient results and rapid execution time. This prototype was implemented in the Python programming language and is based on a metaheuristic inspired by the theory of evolution in living beings known as Genetic Algorithms. The prototype has been tested on the Swiss Road network by Mandl, demonstrating better results than those presented by authors in previous works. 3) In a single run, the algorithm delivers a highly efficient solution that belongs to a set of solutions ranging from 3 to 8 routes. 4) The convergence of the algorithm is attributed to innovative mechanisms in the crossover and mutation operators. These mechanisms trim redundant routes, preventing the algorithm from getting stuck in local optima. 5) The quality function ensures the satisfaction of passenger demand with a high percentage of direct, non-transfer trips, while simultaneously minimizing the average passenger travel time and the total time incurred by all vehicles traveling on all routes. 6) Importantly, the algorithm has demonstrated that achieving high efficiency doesn't require increasing the total route travel time (T_{RV}), as is the case with some other proposals. 7) In the second phase, an assignment algorithm is used to allocate passenger demand to the routes designed in the first phase, determining the demand served by each route. Fleet size and bus frequencies for each route, as well as the total fleet size, were calculated. 8) It was verified that computational complexity is effectively addressed in the first phase.

References

[1] Ahmed, L., Mumford, C., & Kheiri, A. (2019). Solving urban transit route design problem using selection hyper-heuristics. *European Journal of Operational Research*, *274*(2), 545-559.

[2] Arbex, R.O., & da Cunha, C.B. (2015). Efficient transit network design and frequencies setting multi-objective optimization by alternating objective genetic algorithm. *Transportation Research Part B: Methodological*, *81*, 355-376.

[3] Bagloee, S.A., & Ceder, A.A. (2011). Transit-network design methodology for actual-size road networks. *Transportation Research Part B: Methodological*, *45*(10), 1787-1804.

[4] Bourbonnais, P.L., Morency, C., Trépanier, M., & Martel-Poliquin, É. (2021). Transit network design using a genetic algorithm with integrated road network and disaggregated O–D demand data. *Transportation*, *48*, 95-130.

[5] Buba, A.T., & Lee, L.S. (2019). Hybrid differential evolution-particle swarm optimization algorithm for multiobjective urban transit network design problem with homogeneous buses. *Mathematical Problems in Engineering*, *2019*, 1-16.

[6] Cancela, H., Mauttone, A., & Urquhart, M.E. (2015). Mathematical programming formulations for transit network design. *Transportation Research Part B: Methodological*, *77*, 17-37.

[7] Capali, B., & Ceylan, H. (2020). A multi-objective meta-heuristic approach for the transit network design and frequency setting problem. *Transportation Planning and Technology*, *43*(8), 851-867.

[8] Ceder, A. (1987). Methods for creating bus timetables. *Transportation Research Part A: General*, *21*(1), 59-83.

[9] Cipriani, E., Gori, S., & Petrelli, M. (2012). Transit network design: A procedure and an application to a large urban area. *Transportation Research Part C: Emerging Technologies*, *20*(1), 3-14.

[10] Doostie, S., Nakashima-Paniagua, T., & Doucette, J. (2021). A Novel Genetic Algorithm-Based Methodology for Large-Scale Fixed Charge Plus Routing Network Design Problem with Efficient Operators. *IEEE Access*, *9*, 114836-114853.

[11] Fan, L., & Mumford, C.L. (2010). A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics*, *16*, 353-372.

[12] Jiménez-Carrión, M. (2016). Genetic Algorithms from Operations Research, 1st Edition, 111-133. *Editorial Académica Española*.

[13] Jiménez-Carrión, M. (2018). Simple genetic algorithm to solve the job shop scheduling problem (job shop scheduling). *Información tecnológica*, *29*(5), 299-314.

[14] Katsaragakis, I.V., Tassopoulos, I.X., & Beligiannis, G.N. (2020). Solving the urban transit routing problem using a cat swarm optimization-based algorithm. *Algorithms*, *13*(9), 1-27.

[15] Kılıç, F., & Gök, M. (2014). A demand-based route generation algorithm for public transit network design. *Computers & Operations Research*, *51*, 21-29.

[16] Lotero Vélez, L., & Hurtado Heredia, R.G. (2014). Vulnerability of complex networks and applications to urban transportation: a literature review. *Revista EIA*, (21), 67-78.

[17] Mahdi Amiripour, S.M., Mohaymany, A.S., & Ceder, A. (2015). Optimal modification of urban bus network routes using a genetic algorithm. *Journal of Transportation Engineering*, *141*(3).

[18] Manser, P., Becker, H., Hörl, S., & Axhausen, K.W. (2020). Designing a large-scale public transport network using agent-based microsimulation. *Transportation Research Part A: Policy and Practice*, *137*, 1-15.

[19] Nazif, H. (2022). A fuzzy logic-based method for designing an urban transport network using a shark smell optimisation algorithm. *Journal of Experimental & Theoretical Artificial Intelligence*, *34*(4), 673-694.

[20] Nikolić, M., & Teodorović, D. (2013). Transit network design by bee colony optimization. *Expert Systems with Applications*, *40*(15), 5945-5955.

[21] Nisperuza, P.A., López, J.M., & Hernández, H.E. (2019). A Metaheuristic based on the Non-Dominated Genetic Sorting Algorithm II, applied to the Perishable Products Vehicle Routing Problem. *Información tecnológica*, *30*(6), 223-232.
[22] Owais, M., & Osman, M.K. (2018). Complete hierarchical multi-objective genetic algorithm for transit network design problem. *Expert Systems with Applications*, *114*, 143-154.
[23] Possel, B., Wismans, L.J., Van Berkum, E.C., & Bliemer, M.C. (2018). The multi-objective network design problem using minimizing externalities as objectives: comparison of a genetic algorithm and simulated annealing framework. *Transportation*, *45*, 545-572.
[24] Saenko, I., & Kotenko, I.V. (2014). Design of Virtual Local Area Network Scheme Based on Genetic Optimization and Visual Analysis. *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, *5*(4), 86-102.
[25] Taha, H. A., (2012). Investigación de operaciones, Novena ed., 9-11, Pearson Educación, México, México (2012)
[26] Vermeir, E., Engelen, W., Philips, J., & Vansteenwegen, P. (2021). An exact solution approach for the bus line planning problem with integrated passenger routing. *Journal of Advanced Transportation*, *2021*, 1-18.
[27] Yang, J., & Jiang, Y. (2020). Application of modified NSGA-II to the transit network design problem. *Journal of Advanced Transportation*, *2020*, 1-24.

Authors Biography

Miguel Jiménez-Carrión
Miguel Jiménez-Carrión Doctor in Industrial Engineering from the National University of Piura, Peru, 2013. He was the winner of a mentoring contest to advise a participant with her thesis in Economic Engineering at the National University of the Altiplano of Peru. He is currently a Full Professor of the Operations Research Department of the Faculty of Industrial Engineering, and a RENACYT researcher; External Evaluator for CONCYTEC and Innovate Perú. His research interests include artificial neural networks, genetic algorithms, fuzzy logic, fuzzy controllers, shape recognition, mathematical modeling, and optimization.

Gustavo Alexis Flores Fernandez
Gustavo Alexis Flores Fernandez is industrial engineer graduate from the Universidad Nacional de Piura and a master's student in economics at the Universidad San Antonio Abad del Cusco. He has served as a methodological advisor for academic research projects. His areas of interest include algorithmic optimization models, the use of agile methodologies for process optimization, and technological innovation projects.

Alejandro Benjamin Jimenez Panta
Alejandro Benjamin Jimenez Panta holds a Bachelor’s Degree in Mechatronic Engineering from the Pontifical Catholic University of Peru. With three years of experience in the Machine Learning field, he currently serves as a Machine Learning Implementation Engineer, focusing on solving problems related to retention, cross-sell, and conversion through self-learning mechanisms. He co-authored the paper "Unpaired Faces to Cartoons: Improving XGAN," which was presented at CVPRW 2022. His current research interests include Reinforcement Learning and 3D object reconstruction.