A multiobjective bilevel approach based on global-best harmony search for defining optimal routes and frequencies for bus rapid transit systems

Edgar Ruano-Daza[1,*], Carlos Cobos[1,*], José Torres-Jiménez[2], Martha Mendoza[1], Alexander Paz[3]

[1] *Information Technology Research Group, University of Cauca, Popayan, Colombia*

[2] *Information Technology Laboratory, CINVESTAV-Tamaulipas, Victoria City, México*

[3] *Transportation Research Center, University of Nevada, Las Vegas, US*

[*] *Corresponding authors*

eruano@unicauca.edu.co, ccobos@unicauca.edu.co, jtj@cinvestav.mx, mmendoza@unicauca.edu.co, apaz@unlv.edu

Abstract

The operation of a Bus Rapid Transit System (BRTS) requires solving complex design problems, including the selection of routes and the corresponding frequency of service, among others. One challenge is the large number of potential solutions, with multiple routes having many frequency options. At least, the design should seek to minimize the total time spend by users in the system as well as the operational cost to pursue sustainability and profitability. The literature includes a wide variety of approaches to solve this Transit Network Design and Frequency Setting Problem (TNDFSP) for a BRTS. This study proposes a single framework that simultaneously considers restrictions and objectives of the users and the operator of the system. A Multi-objective Global-Best Harmony Search (MOGBHS) heuristic algorithm was implemented and tested successfully. The algorithm is based on three main components: 1) a Global-Best Harmony Search as the heuristic optimization strategy, 2) ordering of non-dominated solutions as a multi-objective optimization strategy, and 3) simulation of discrete events to evaluate solutions. A bi-level implementation of MOGBHS was adopted. At the external level, the algorithm searched the best configuration of routes, while at the internal level, the algorithm searched the best frequency for a solution for a specific route. Experiments were performed using a simulation model of an actual BRTS located in Pereira, Colombia, known as Megabus. Routes and frequencies were searched for this BRTS by minimizing waste bus capacities (operation costs) and minimizing users' travel time (maximizing satisfaction). Results using the proposed algorithm where superior to those obtained using comparable alternatives, including NSGA-II and MOEA/D.

Keywords: Bus Rapid Transit Systems, Transit Network Design and Frequency Setting Problem, Multiobjective Optimization, Global-Best Harmony Search, Covering Arrays, NSGA-II, MOEA/D.

1 Introduction

Bus Rapid Transit Systems (BRTS) have been shown to be a viable solution to address growing transportation needs of populations in large urban centers and to reduce negative externalities created by conventional automobiles, such as emissions and noise. BRTS provide similar cost and efficiency as do urban railway systems [1, 2]. However, the operation of a BRTS requires addressing various challenges including: 1) design of the network, 2) design of the routes, 3) definition of service frequency by route, 4) assignment of buses, 5) assignment of personnel, 6) consideration of induced demand, 7) design of traffic controls, and 8) generation and deployment of travel information strategies. The growing use of BRTS as well as diverse and changing conditions in which passengers travel make the process of determining service routes and frequencies essential to provide effective service. It also is critical to consider the needs of the service providers, who seek the greatest profitability and sustainability of the system over the short, medium, and long term [3-5].

Ideally, the design of a BRTS needs to reach desired levels of user satisfaction, sustainability, and profitability. This is evident in the literature regarding problems involving "design of transit networks and routes", "programming of frequency", and "programming of schedules" for the design of BRTS. These problems have been studied in detail within a global framework, the Transit Network Design and Scheduling Problem (TNDSP) [4].

Given the importance of transit systems, the above problems have been addressed using a variety of approaches, including analytical and heuristic frameworks [4-6]. Within the set of heuristics, Multi-objective Optimization (MO) [7] has provided significantly good results when solving one or more sub-problems of the TNDSP. MO enables the explicit consideration of multiple objectives at once in order to find an array of best feasible solutions. It is the designer or decision-maker who selects a solution from the list of alternatives provided by the MO algorithm [7-9]. The optimization process depends on feedback obtained from evaluating feasible solutions found by the algorithm. In the case of TNDSP, the cost involved in searching for solutions increases by the amount of restrictions that must be considered. To address this challenge, this current study proposes to use simulation tools for discrete events to facilitate the generation of necessary data to calculate an aptitude or quality of the solutions based on the objectives to be optimized. The existing literature reports substantial results when using this proposed approach to address other similar type of problems [10].

Considering the quantity of the objectives to optimize as well as their conflicts, this paper proposes a new multi-objective bi-level algorithm to solve the Transit Network Design and Frequency Setting Problem (TNDFSP), in which the configuration of routes and frequencies are searched at the external (leader) and internal (follower) levels, respectively. The algorithm is based on:

(1) A global-best harmony search [11], which is a metaheuristic that is widely known and used to solve diverse complex problems for continuous and discrete optimization; and
(2) The concept of ordering of non-dominates [12] to compare and evaluate the best solutions found in a problem that has multiple objectives.

The implementation includes the use of covering arrays [13] as a tool to reduce the search space of the routes to be defined at the external level of the algorithm.

Diverse solutions to problems related to TNDSP have been proposed. However, most studies have focused on optimizing only one objective to satisfy the needs of one group, the users, or the operators, without considering other objectives. Hence, solutions obtained using the existing literature are sub-optimal or unsustainable for real operations. To the best knowledge of the authors, an approach such as the one proposed in this study is not available in the literature.

The remaining of this paper is organized as follows. Section 2 summarizes a set of studies related to TNDSP, multi-objective bi-level optimization, and the Global-best Harmony Search (GBHS) metaheuristic. Section 3 provides details about the proposed algorithm, Multi-objective Global Best Harmony Search (MOGBHS), including its evaluation using instances of the multi-objective optimization test [14] from the Congress on Evolutionary Competition (CEC) of the Institute of Electrical and Electronics Engineers (IEEE). This section also explains how the MOGBHS was adapted, using two search levels for the solution of TNDFSP. Section 4 describes the BRTS model used in the experiments, the process of refining the parameters of the proposed algorithm, the results of experiments, and a comparison with the algorithms, Non-dominated Sorting Genetic Algorithm (NSGA-II) and Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D). Finally, Section 5 presents conclusions and recommends future directions for research.

2 Related Studies

2.1 Transit Network Design and Scheduling Problem

In 2007, Mauttone and Urquhart [3] proposed an approach to solve the Transit Network Design Problem (TNDP) by using an adaptation of the Greedy Randomized Adaptive Search Procedure (GRASP), a metaheuristic approach designed to solve combinatorial optimization problems. In contrast to GRASP, the GRASP-TNDP algorithm tries to obtain an approximate Pareto front with multiple solutions. In that study, the evaluation was performed successfully using information from the transit system of the City of Rivera in Uruguay. However, implementation and testing of the algorithm included simplifications of the problem that might make the solutions unusable for real applications. For example, the capacity of automobiles in the system was not considered. To address this limitation, the algorithm proposed is this study used discrete simulation models, which made it possible to include more dimensions in the experiments and enable future improvements.

Beltran et al. [15] used a genetic algorithm to determine the frequency of service for a public transit system having Zero Emission Vehicles. Their study sought to minimize operation costs, the cost to users, and external costs by considering the elasticity of the demand and the number of vehicles. However, solutions for routes were generated independently of the genetic algorithm that sought the frequencies. Hence, this approach did not make use of information about the quality (fitness) of the solutions.

Mauttone and Urquhart [16] proposed a multi-objective metaheuristic based on GRASP to validate TNDSP results by using graphs with models on the scale of cities. This study demonstrated that by using the same computational effort as a weighting method, the proposed algorithm produced more non-dominated solutions. In addition, the study demonstrated that the optimization of routes and frequencies was a solvable multi-objective problem. In contrast to the algorithm used by Mauttone and Urquhart, the algorithm proposed in this current study uses simulation models based on existing transit systems to obtain the necessary information to calculate the quality of the solutions. This approach makes it possible to consider more dimensions and, therefore, facilitate its implementation for real-world systems.

Using Swarm Intelligence, Cipriani et al. [17] formulated a design of transit networks for urban buses as a problem of optimization of resources and costs. The final objective was the design of routes, including some restrictions on bus capacity and feasibility. This approach could be categorized as multi-objective optimization. However, it does not address the problem of finding optimal bus frequencies.

Calculating the fitness of each solution generated with respect to N objectives has led to using simulation to enable an efficient evaluation of the system under consideration. The strategy consists of feedback loops that use solutions generated by search algorithms on microscopic, macroscopic, mesoscopic models of the system. An example of this was illustrated by Wang et al. [18], where a fee structure and service frequency was found by using a genetic algorithm and a Simulated Annealing algorithm to obtain a supply-and-demand equilibrium for a transit system. To test the model and the suggested algorithms, the best solutions were applied to a real system in Guangzhou, China.

Using an $\in$-constraint method, Ibarra-Rojas et al. [19] developed a bi-level objective formulation to program schedules and vehicles in order to analyze the relationship between the level of service and operation costs of transit networks. Zhao and Jiang [20] proposed a memetic algorithm to configure routes and frequencies in order to minimize the cost to passengers and reduce the unmet demand. However, this approach sought only the satisfaction of the users without considering the operation costs. In contrast, this current study used an approach in which both users and operators were explicitly considered, making the proposed approach applicable for real bus rapid transit systems.

Giesen et al. [21] applied multi-objective optimization for scheduling the frequency in a BRTS. They proposed an adjustment to the Tabu Search algorithm [22, 23] in order to consider two objectives simultaneously, minimize the total travel time for the passengers, and minimize the size of the fleet.

The existing literature provides examples of multi-objective optimization used to solve one or more sub-problems of TNDSP. However, there is no single framework that can search simultaneously for routes and frequencies while also considering the objectives of the operators and users. For example, some implementations focused on the frequency of service and/or considered only the objectives of the users. Furthermore, the models used to evaluate the quality of the solutions were complex to build, adapt, or expand.

This research addressed these limitations to search simultaneously for routes and frequencies while minimizing travel time and operational costs for BRTS. A multi-objective bi-level algorithm based on GBHS was implemented and tested to determine the solution to the proposed problem. The implementation used simulation models of discrete events to evaluate the quality of the solutions and facilitate their use for real bus rapid transit systems.

2.2 Multi-objective Bi-level Optimization

Multi-objective bi-level optimization problems are complex, and require solutions for the lower level (follower) considered during the search of solutions for the upper level (leader) [24]. Various techniques have been proposed to solve this type of problems, including:

1) Classical bi-level approaches that use, for example, numerical optimization at the lower level and adaptations of an exhaustive search at the upper level [25];
2) Approximated set-valued mapping procedures for evolutionary approaches [24];
3) Transformation of the original multi-objective bi-level problem into a multi-objective single-level optimization task with complementarity constraints [26];
4) Multi-objective mixed-integer programming [27];

5) Evolutionary algorithms [28];
6) An approach based on adaptive scalarization and evolutionary algorithms with minimum modifications [29];
7) Multi-objective particle swarm optimization [30]; and
8) Hybrid particle swarm optimization with crossover operators [31].

Important applications of multi-objective bi-level optimization include, among others, transportation planning and management [32], assignment of resources, supply-chain planning, environmental economics, structural optimization and engineering design [33], game theory, and decision making that involves qualitative variables [26] [29].

2.3 Global-Best Harmony Search

The Harmony Search (HS) optimization algorithm, originally proposed by Lee and Geem [34], is based on the improvisation process used by jazz musicians in search of the perfect harmony. HS randomly generates a set of solutions, evaluates them, and then organizes a population of solutions in a Harmony Memory (HM). For a certain number of improvisations (or iterations), the algorithm generates a new harmony. Each variable of the new harmony can be completely random or taken from the HM based on a parameter known as the Harmony Memory Consideration Rate (HMCR). If taken from harmony memory, depending on a parameter known as the Pitch Adjustment Rate (PAR), the variable may or may not be altered by adding or subtracting a value defined by the Bandwidth (BW) parameter.

Variations or improvements of the HS algorithm have been proposed. Some of the most relevant include the Improved Harmony Search [35], GBHS [11], the Self-adaptive Harmony Search [36], Self-adaptive Global Best Harmony Search [37], Global-best Harmony Search + Learnable Evolution Models (GHS+LEM) [38], Global Dynamic Harmony Search [39], and Improved Global Best Harmony Search [40]. There are many applications of HS, including traffic signal optimization [41], economic load dispatching [42], routing in wireless sensor networks [43], and artificial neural network optimization [44]. However, some factors that need to be considered are:

(1) GBHS uses five parameters to fine-tune performance;
(2) GBHS provides rapid convergence, requires few calculation resources, and has a low probability of being trapped in suboptimal solutions [34, 37];
(3) GBHS improves the results of an improved harmony search (IHS), which in turn improves the precision and convergence of HS; and
(4) GBHS works efficiently both with discrete and continuous problems.

This research used GBHS as the base for the proposed multi-objective algorithm. In contrast to HS, GBHS makes the value of PAR depend on the number of improvisations (or iterations). This guarantees a greater exploration and exploitation at the beginning and end of the search, respectively. The original paper on GBHS by Omran and Mahdavi [11] explains the behavior (exploration/exploitation) of the algorithm by means of the search process (i.e., improvisations). The number of improvisations (or iterations) is an algorithmic parameter that is defined by the user. , Among other factors, this number depends on the maximum execution time and the minimum quality of the expected results. This approach replaces adjusting for the addition or subtraction of the BW parameter with the extraction of a value from the best harmony stored in HM. With this replacement, the algorithm incorporates swarm characteristics, as in Particle Swarm Optimization (PSO).

HS and its improvements, including GBHS, originally were designed to work with only one objective. Subsequently, Sivasubramani and Swarup [45] adapted HS to allow for the simultaneous optimization of various objectives, making use of the ordering of nondominated solutions and Crowding Distance. This current study presents a multi-objective version of GBHS, using a similar approach.

Most MO optimization algorithms use one of the following methods to improve a population of solutions: 1) Pareto domination, 2) decomposition, 3) preference, 4) indicator-based selection, 5) hybrid, 6) memetic, and 7) coevolution. Adequate results usually are obtained by using methods based on Pareto domination by means of ordering nondominated solutions [12, 15-18, 46]; this method was the one selected for this study. Furthermore, Crowding Distance was included as a mechanism to guarantee greater diversity and to explore solutions during the evolution of the algorithm.

3 Proposed Algorithm

In order to solve the TNDFSP, a multi-objective bi-level algorithm based on GBHS was developed and coupled with a simulation model for discrete events. This section first describes the multi-objective adaptation of GBHS (MOGBHS), and then describes its evaluation, using instances of the multi-objective optimization test from the Congress on Evolutionary Competition of the IEEE (IEEE-CEC) [14]. Finally, the fitness function for TNDFSP and the bi-level adaptation of MOGBHS for the solution of TNDFSP in two levels is presented.

3.1 Multi-objective Global Best Harmony Search

The proposed algorithm, MOGBHS, randomly generates a set of harmonies and stores them in the HM, evaluates all the objectives for each element in the HM, and then sorts them by Pareto front and Crowding Distance. Subsequently, the algorithm performs several evolutionary iterations. In each iteration, the following steps are executed:

1) A new harmony is generated applying the logic of the GBHS algorithm. If the new harmony is a valid solution and it does not exist in HM, the algorithm moves to Step 2; otherwise, the solution is discarded and MOGBHS continues with the next improvisation (or iteration).
2) The new harmony is evaluated, using all the objectives to be optimized.
3) The new harmony is added to the existing HM.
4) The HM is organized by Pareto fronts and Crowding Distance.
5) The worst element contained in the HM that makes the HM exceed its maximum size, defined by the Harmony Memory Size (HMS) parameter, is eliminated.

The variables used during the implementation of the MOGBHS algorithm are:

HM:	An arrangement that stores all the solutions proposed by the algorithm. The HMS parameter determines its maximum size.
I:	A counter of improvisations performed by the algorithm.
J:	A counter of the variables of one harmony in each improvisation.
PAR:	The Pitch Adjustment Rate determines the probability that a value previously taken from the harmony memory is replaced by a value from the best existing harmony in HM. PAR for each improvisation is determined by:

$$PAR = PARmin + (PARmax - PARmin) * (I / NIter) \quad (1)$$

where PARmin, PARmax, and NIter are parameters of the algorithm.

The parameters necessary for the implementation of MOGBHS are:

HMS:	Harmony Memory Size, maximum size that the harmony memory can reach.
HMCR:	Harmony Memory Consideration Rate, determines the probability of selecting values for the new harmonies from the HM.
NVariables:	Number of variables that involve each harmony; it depends on the configuration used to represent the solutions to the problem.
NIter:	Maximum number of iterations/improvisations to be performed for the algorithm.
NObjectives:	Number of objectives that will be optimized.

Figure 1 shows the MOGBHS pseudocode, whose functions will be described in this section. Function *PopulationRandomInitialize* is responsible for the random generation of an initial population of harmonies of size HMS. In the generation of each harmony, its viability was evaluated by considering the restrictions of the model and the problem to optimize. If viable, the fitness values were calculated for each one of the objectives; otherwise, the harmony was discarded. **Figure 2** presents the pseudocode for this function.

```
PopulationRandomInitialize(HM)
NonDominatedOrderCalculate(HM)
CrowdingDistanceCalculate(HM)
```

```
Sort(HM)
for I = 1 to NIter do
    NewHarmony = null
    PAR = PARMin + (((PARMax - PARMin) / NIter) * I)
    for J = 0 to NVariables - 1 do
        if Random(0,1) < HMCR then
            NewHarmony.Variable[J] = HM.Harmonies[Random(0,HMS)].Variable[J]
            if Random(0,1) < PAR then
                NewHarmony.Variable[J] = HM.FromBestHarmony(Random(0, NVariables))
            end if
        else
            NewHarmony.Variable[J] = Random(MinDomainValue, MaxDomainValue)
        end if
    end for
    if InPopulation(HM, NewHarmony) = false AND IsViable(NewHarmony) then
        Evaluate(NewHarmony)
        Add(HM, NewHarmony)
        NonDominatedOrderCalculate(HM)
        CrowdingDistanceCalculate(HM)
        Sort(HM)
        RemoveTheWorst(HM)
    end if
end for
return the Pareto front in HM
```

Figure 1 Pseudo-code of the MOGBHS

```
count = 1
while count < HMS
    for k = 0 to NVariables - 1 do
        NewHarmony.Variable[k] = Random(MinDomainValue, MaxDomainValue)
    end for
    if IsViable(NewHarmony) then
        Evaluate(NewHarmony)
        Add(HM, NewHarmony)
        count = count + 1
    end if
end while
```

Figure 2 Pseudo-code of the function *PopulationRandomInitialize*

The function *Evaluate* establishes the fitness values that the harmony obtains when it is evaluated against all the objectives. These values are stored in the vector Evaluations of size NObjectives that each harmony should have. The behavior of this function depends on the domain on which the algorithm is being applied and the objectives that are being optimized. In this study, the domain corresponds to TNDSP.

The function *NonDominatedOrderCalculate* [12] establishes, for each element of the HM, its number of Pareto front (rank). Each harmony has the following attributes:

Front:	Determines its number of Pareto front or rank.
BossCount:	Stores the number of solutions of the population that dominate the actual harmony.
Dominated:	A list that makes it possible to store the elements dominated by the actual harmony.

The function *Dominates* is used to calculate the ordering of nondominated solutions. It returns true or false to indicate when a harmony dominates another, considering all objectives.

The function *CrowdingDistanceCalculate* [12] takes the existing population of solutions and establishes their Crowding Distance to the harmonies located in the same Pareto front.

3.1.1 Evaluation and Comparison using IEEE-CEC Problems for Multi-objective Optimization Competition

Before comparing the proposed algorithm against other MOEAs in a specific TNDFSP, an evaluation and comparison was performed using 9 and 12 multi-objective continuous problems with and without constraints, respectively. These problems (i.e., test instances) were taken from the multi-objective optimization competition of the IEEE-CEC [14]. MOGBHS was implemented within the MOEA framework, which facilitated performing a comparative analysis with other widely used algorithms, including NSGA-II, MOEA/D, Multiple Single Objective Pareto Sampling (MSOPS), and Strength Pareto Evolutionary Algorithm 2 (SPEA2) [47]. The study sought to determine the effects of the number of evaluations (2,000, 5,000, 10,000 and 20,000) of the objective function (EOFs) for different types of problems with and without constraints. The comparative analysis was performed using the Inverted Generational Distance metric commonly accepted by the scientific community [48] and the nonparametric statistical tests of Friedman and Wilcoxon.

General results of this comparison are provided in **Table 1**. MOGBHS and NSGA-II obtained the best average results followed in order by MOEA/D, MSOPS and SPEA2. At 2,000 EOFs best results are found by MSOPS (first place) with a ranking of 2.1 using the Friedman test. The Wilcoxon test shows, with a level of significance of 0.95, that MSOPS outperform all other algorithms with this amount of EOFs. NSGA-II and MOEA/D outperforms the results obtained with SPEA2 while NSGA-II outperforms the results obtained with MOGBHS. Rankings obtained at 5,000 EOFs are not accepted by the Friedman test because the p-value is not less than 0.05 but the Wilcoxon test shows, with a level of confidence of 0.95, that NSGA-II obtained better results than SPEA2. At 10,000 and 20,000 EFOs, the results are similar, and the Wilcoxon test shows the same results. That is, MOGBHS and NSGA-II outperforms the results obtained by MSOPS and SPEA2 with a level of significance of 0.95.

Table 1 Inverted Generational Distance Rankings for MOGBHS, NSGA-II, MOEA/D, MSOPS, and SPEA2 with Various Numbers of Evaluations of the Objective Function

EOFs	**MOGBHS**	**NSGA-II**	**MOEA/D**	**MSOPS**	**SPEA2**	**Chi^2**	**p-value**	**Wilcoxon Conclusions with 95% of Significance**
2,000	3.60 (4)	2.45 (2)	2.65 (3)	**2.10 (1)**	4.20 (5)	24.28	7.01 E-5	MSOPS > MOGBHS, NSGA-II, MOEA/D, MSOPS NSGA-II, MOEA/D > SPEA2 NSGA-II > MOGBHS
5,000	2.80 (2)	**2.6**5 (1)	2.90 (3)	2.90 (3)	3.75 (4)	5.96	0.2022	NSGA-II > SPEA2
10,000	**2.00 (1)**	2.50 (2)	2.90 (3)	3.80 (4)	3.80 (4)	20.32	4.31 E-4	MOGBHS, NSGA-II > MOPS, SPEA2
20,000	**2.00 (1)**	2.80 (2)	2.80 (2)	3.95 (5)	3.45 (4)	17.48	0.0016	MOGBHS, NSGA-II > MOPS, SPEA2
Avg.	**2.6 (1)**	**2.6 (1)**	2.8 (3)	3.2 (4)	3.8 (5)			

3.2 Objectives for TNDFSP

Considering various limitations of the existing TNDFSP literature [4, 49-56], a desirable objective is the simultaneous consideration of users and operator costs. Hence, the following were the optimization objectives during this study:

- Minimize the time spent in the system by the users, and
- Minimize the operation costs. It was assumed that the system-wasted transit capacity negatively influences its operation costs. Therefore, the operation costs are minimized by minimizing the wasted capacity of the buses in transit.

These two objectives are in conflict. Based on a constant fleet size value and the same conditions for passengers, several scenarios could arise. On one hand, a high frequency of service reduces user travel time but the wasted capacity of the buses increases. On the other hand, the low frequency of service increases user time, but there is a direct decrease in wasted capacity. If a very high frequency of service is used, congestion and extra waiting may result, and user time in the system might not necessarily be reduced as much as expected. If the operator could increase the fleet size and use new vehicles in a reasonably intelligent way, less user travel time in the system might be achieved. In contrast, if the fleet size is reduced, it is expected that the users would spend more time in the system.

Considering that the proposed multi-objective algorithm uses the ordering of nondominated solutions and Crowding Distance to classify and select the best feasible points, the objectives to be minimized were evaluated independently. In the experiments to evaluate the solutions generated by the algorithm, a discrete-events simulation model was created for a specific BRTS. This type of simulation model is relatively simple to implement, expand, and adapt. Hence, the proposed methodology could be used to determine optimal routes and frequencies for other BRTS.

Given the optimization objectives, the corresponding measures to obtain from the simulation model were:

F1: The time users spent in the system, which corresponds to the average time measured between entering and exiting stations at the origin and destination.

F2: The wasted capacity of the buses, which corresponds to the average percentage of unused space in the buses during the entire operation. The percentage of wasted capacity is defined as the available capacity divided by the total capacity of the bus.

To perform the simulation and capture the required measures, ARENA® (Rockwell Software) was used [57-59]. ARENA is based on SIMAN [60], a general-purpose simulation language.

The variables referring to the BRTS infrastructure were defined as follows for stops, stations (single or double), and routes between stations (represented by average transit time between them):

PNo: The total number of stops (points of entry/exit to/from buses) of a BRTS.

E: The set of size *PNo* of stops, where *E[i]* represents a stop to get on or off the bus, such that $i \in [1, \dots, PNo]$.

T: A two-dimensional matrix of size *PNo x PNo* with real values, in which the average trip times between the BRTS stations are defined. A matrix representing the graph of possible trips on the BRTS is expressed as:

$$T\ (i,j) = \begin{cases} 0, if\ stops\ i\ and\ j\ are\ in\ the\ same\ station \\ NaN, if\ there\ is\ no\ direct\ connection\ between\ stops \\ \mathbb{R}^{+}, in\ other\ case, representing\ the\ average\ time\ of\ trip\ from\ station\ i\ to\ j \end{cases} \quad (2)$$

The variables related to routes and frequencies that buses travel using the infrastructure previously defined for the BRTS include:

RNo: The total number of routes defined for the BRTS.

R: The set of routes defined in the system, where each *R[i]* contains the information necessary to determine the order of trips between stops and the action to be performed in each stop, such that $i \in [1, \dots, RNo]$.

R[i]: Vector of elements composed of <*E[i], action*>, which represents the order of the stops visited by each bus that follows route *R[i]*. Each element indicates the station included in the trip and the action to be performed (stop or continue).

F: A set of size *RNo*, where each *F[i]* defines the frequency in minutes of departure of buses corresponding to route *R[i]*, where $1 \leq F[i] \leq 30$ for each $i \in [1, \dots, RNo]$.

The variables related to the quantity of buses, their carrying capacity, and the routes they follow are:

TF: Size of flee, total number of available buses.

C: Carrying capacity of each bus. Maximum number of users that can be transported along a route from station *i* to *j* at a given moment.

B: Indicator matrix of size *TF* x 3. $B[i][1]$ indicates whether the *i*-th bus is available or not. A value 1 indicates that the bus is available; O otherwise. $B[i][2]$ indicates the current station, or that bus *i* has departed from that station. $B[i][3]$ indicates the target station, i.e. that the bus has departed from station $B[i][2]$ and is on its way to station $B[i][3]$. If $B[i][2] = B[i][3]$, the bus is not moving, i.e. it is at station $B[i][2]$. The stations involved in a specific bus status must belong to one of the routes specified in vector *R* and their status must be “stop station” as defined in vector *E*.

The variable related to the users of the BRTS and their time in the system is:

P: Set of users who enter through a station in set *E* and have as destination a different station in *E*.

The variables related to the implementation of the simulation are:

TR: Number of trips completed during simulation time (*ST*). A trip takes place when a Bus *B*[*k*] leaves station *E*[i] to another station *E*[*j*], where *i* <> *j*, following a specific route.

CO: The set of all the *TR* trips completed between stations by the entire fleet in circulationm fulfilling the configuration of routes and frequencies defined in *R* and *F* during the *ST* and on the topography defined by *E* and *T*. Each *CO[i]* represents the vehicle carrying capacity, that is, the number of passengers that the bus carries while it moves from one station to another.

CD: The set of size *TR*, which represents the capacity wasted by the bus during the trip, where each *CD[i]* is equivalent to *C – CO[i]*.

TP: Number of users who used the BRTS during *ST*.

TSP: The set of size *TP*, which registers the time of service (time of egress - time of entry) of each BRTS user, using the topography defined by *E* and *T*, with the configuration of routes and frequencies defined in *R* and *F*, and during the *ST*. Each *TP[i]* >= 0.

The objective functions to be minimized are

$$F1 = \frac{1}{TP}\sum_{i=1}^{TP} TSP[i] \tag{3}$$

$$F2 = \frac{1}{TR}\sum_{i=1}^{TR} CD[i] \tag{4}$$

The algorithm proposed in this paper generates the values for *R* and *F* in order to minimize *F1* and *F2* simultaneously.

3.3 MOGBHS-TNDFSP

The algorithm described in this section has the goal of finding the best sets of bus routes and frequencies for each route selected as part of each solution condition to a predefined structure of stations and road corridors. To solve this TNDFSP using the MOGBHS, in addition to the variables and parameters described in the previous section, it is necessary to incorporate the following elements:

- Files of SIMAN source code generated from the simulation model created in ARENA to represent the BRTS to be optimized.
- A database with a set of feasible routes for the BRTS. Each record in the database included, at minimum, the following information:
    - Stations that are part of the route,
    - Stations where buses stop along each route, and
    - A unique identifier.

The database of feasible routes could be built using multiple approaches. In this study, Covering Arrays (CAs) were used [13, 61]. The process of creating the database of routes using CAs is described in Section 3.3.1. Once all possible routes for the BRTS are defined, it is necessary to assign an entire unique and consecutive identifier to all of them. A pair of parameters were included within MOGBHS to determine the minimum and maximum identifiers of the routes in the database. This facilitates the selection of routes when generating harmonies. Minimum and maximum parameters were included to define the quantity of routes that a harmony generated by MOGBHS could provide. The implementation takes into consideration that:

1) The quantity of feasible routes, the quantity of possible combinations for the configuration of routes, and the quantity of frequencies for each route creates a very large search space; and
2) State-of-the-art practices includes successful cases that used multi-level algorithms, in which each level solves a particular problem from the set of TNDSP.

As described in Section 3.3.3 of this paper, MOGBHS-TNDFSP corresponds to an algorithm that searches for the best sets of routes. It relied on MOGBHS-TNFSP to find the best frequencies of bus departures along the selected routes and to perform the evaluation of the quality (fitness) of the harmony (solution) with respect to the optimization objectives.

A harmony generated by MOGBHS-TNDFSP contains the number of routes selected, the list of identifiers of the selected routes, and the frequencies of bus departures for each route. The structure of one harmony is presented in **Figure 3**, where N corresponds to the number of routes that formed the harmony, R_1 to R_N correspond to the identifiers of the selected routes, and F_1 to F_N are the frequencies of bus departures for the corresponding routes. The length of each harmony is variable depending on the number of included routes, and corresponds to $2N + 1$, with N being the number of routes.

0	1	2	…	$N-1$	N	$N+1$	$N+2$	…	$2N-1$	$2N$
N	R_1	R_2	…	R_{N-1}	R_N	F_1	F_2	…	F_{N-1}	F_N

Figure 3 Representation of a MOGBHS-TNDFSP solution.

The process of building each harmony involves two phases, both using the improvisation logic of GBHS, as follows:

1) The selection of the number of routes that would form the harmony, and
2) The selection of routes from the database, taking into consideration the number of routes previously selected.

The generation of frequencies for the selected routes and the evaluation of quality of the harmony with respect to the objectives to be optimized were performed by implementing the internal algorithm, MOGBHS-TNFSP. MOGBHS-TNDFSP generates an initial harmony memory that is organized by a Pareto front and Crowding Distance. Then, it runs a predetermined number of iterations to generate new harmonies, which were added to the existing HM if they are considered viable based on such design restrictions as the available fleet. At the end of each iteration, if a feasible solution was found and added to the HM, the existing harmony memory is reorganized based on Pareto front and Crowding Distance in order to remove the worst harmony, which is the one that had the greatest number in the Pareto front and the least Crowding Distance. **Figure 4** shows the pseudocode of MOGBHS for TNDFSP.

```
PopulationRandomInitialize(HM)
NonDominatedOrderCalculate(HM)
CrowdingDistanceCalculate(HM)
Sort(HM)
for I = 1 to NIter do
    NewHarmony = null
    PAR = PARMin + (((PARMax - PARMin) / NIter) * I)
    //selection of the number of routes
    if Random(0,1) < HMCR then
        NewHarmony.Variable[0] = HM.Harmonies[Random(0,HMS)].Variable[0]
        if Random(0,1) < PAR then
            NewHarmony.Variable[0] = FromBestHarmony(HM, 0)
        end if
    else
        NewHarmony.Variable[0] = Random(MinRoutesNumber,MaxRoutesNumber)
    end if
    //selection of the routes
    for J = 1 to NewHarmony.Variable[0] do
        if Random(0,1) < HMCR then
            NewHarmony.Variable[J] = HM.Harmonies[Random(0,HMS)].Variable[J]
            if Random(0,1) < PAR then
                NewHarmony.Variable[J] = FromBestHarmony(HM, Random(0, NVariables))
            end if
        else
            NewHarmony.Variables[J] = Random(MinRouteId, MaxRouteId)
        end if
    end for
    SolutionComplete(NewHarmony)
    if InPopulation(HM, NewHarmony) = false AND IsViable(NewHarmony) = true then
        Evaluate(NewHarmony)
        Add(HM, NewHarmony)
        NonDominatedOrderCalculate(HM)
        CrowdingDistanceCalculate(HM)
        Sort(HM)
        RemoveTheWorst(HM)
    end if
end for
return the Pareto front in HM
```

Figure 4 MOGBHS adapted to solve TNDFSP.

Once a new harmony is generated, whether during the initialization of the harmony memory or during the improvisations of the algorithm, it is necessary to validate its feasibility. A harmony is considered feasible when the configuration of the selected routes applied to the BRTS allows a user to go from any station to any other, even when making one or more transfers. If a harmony is not viable, then the process *SolutionComplete* is used, which consists of:

1) Identifying isolated stations or those that have the greatest connection problems with the configuration of the routes defined in the harmony.
2) Identifying an ‘easy’ route, i.e., the route that stops at all the stations for a trip; this covers the greatest quantity of stations identified above in Step 1.
3) Including an identified ‘easy’ route in the current harmony and evaluating if it meets the feasibility condition, such that it is possible to reach any station from any other, even when making one or more transfers. In the case that the route is not feasible, the process is repeated from Steps 1 to 3; otherwise, the process is completed.

Figure 5 presents the pseudocode of function *SolutionComplete*, which is used to apply the process described previously.

```
lstEasyRoutes = LoadEasyRoutes(DBManager)
lstTroubleStations = IsViable(Harmony)
While lstTroubleStations.Count > 0 Then
    lstCountStations = null // create a new dictionary
    for I = 0 to lstEasyRoutes.Count – 1 do
        Add(lstCountStations, lstEasyRoutes[I].ID, 0)
        for J = 0 to lstTroubleStations.Count – 1 do
            if Contains(lstEasyRoutes[I].Stations, lstTroubleStations[J]) then
                lstCountStations[lstEasyRoutes[I].ID]++
            end if
        end for
    end for
    SortByValue(lstCountStations)
    Add(Harmony.Routes, lstCountStations.first().ID)
    lstTroubleStations = IsViable(Harmony)
end while
```

Figure 5 Function *SolutionComplete.*

Figure 6 shows the pseudocode of function *IsViable*, which is an aid to *SolutionComplete*. This function is responsible for returning a list with identifiers for problematic stations.

```
lstTroubleStations = null / create a new dictionary
for I = 0 to NStations -1 do
    lstDestines = FindDestinesFrom(I)
    for J = 0 to NStations – 1 do
        if I <> J and Not Contains(lstDestines, J) then
            if ContainsKey(lstTroubleStations, J) then
                Add(lstTroubleStations, J, 1)
            else
                lstTroubleStations[J] ++
            end if
        end if
    end for
end for
SortByValueDesc(lstTroubleStations)
return getFirstKeys(lstTroubleStations)
```

Figure 6 Function *IsViable*

In the simulation model, as in a real BRTS, the behavior of the passengers depends on the available routes. Users usually tend to use the shortest routes, in terms of time, to reach a destination. Taking into account that each harmony defines a different set of available routes, it becomes necessary to update the ‘matrix of short routes’ in the simulation model so that the behavior of the passengers is consistent with the available routes. In order to calculate the short routes for each harmony generated in the course of the implementation of MOGBHS-TNDFSP, a variation of the Dijkstra algorithm [62] is proposed (see details in Section 3.3.2). Once the matrix of short routes is generated, the source files of the simulation model are updated. Function *Evaluate* of MOGBHS-TNDFSP is responsible for:

- Invoking the calculation of short routes among all the stations in the system, considering the set of routes selected to form the harmony that is being evaluated.
- Inserting the configuration of routes and corresponding short routes to the solution that is being evaluated into the SIMAN source code files.

- Invoking MOGBHS for TNFSP (follower level) to find the best combinations of bus departure intervals for the simulation model, using the routes and corresponding short routes to the harmony that is being evaluated.

This adaptation of MOGBHS for TNFSP is described in Section 3.3.3.

Figure 7 shows function *Evaluate* of MOGBHS for TNDFSP. In this figure, function *ReplaceInFile* receives three parameters:

- The name of the file in which the search should be performed,
- The searched text to be replaced, and
- The new value to be established in the SIMAN code files.

At the end of the implementation of MOGBHS-TNDFSP, there is a set of harmonies that represent the best configurations of routes and intervals for the BRTS. Hence, the analyst or decision maker could select, at his or her discretion, the solution to be implemented in the real world.

```
RoutesDescription = Array[Harmony.Variable[0]][ConstTotalStations]
for k = 1 to Harmony.Variable[0] do
    RoutesDescription[k-1] = ExtractRouteString(DBManager, Harmony.Variable[k])
end for
ReplaceInFile("Transport.exp", "$CantidadRutas", Harmony.Variable[0])
ReplaceInFile("Transport.exp", "$DetalleRutas", RoutesDescription.ToString())
ShortRoutes = DijkstraShortRoutes()
ReplaceInFile("Transport.exp", "$RutasCortas", ShortRoutes)
BestHarmony = Execute(MOGBHSforTNFSP)
Harmony.Intervals = BestHarmony.Intervals
Harmony.Evaluations = BestHarmony.Evaluations
```

Figure 7 Function *Evaluate* of MOGBHS for TNDFSP

3.3.1 Covering Arrays to Create the Database for Routes

In this study, a set with all possible feasible routes for the BRTS under consideration needed to be generated. This is a highly time-consuming problem. Covering Arrays [13] were used successfully to generate a representative set of feasible routes. It should be noted that this database of routes could be generated with many other mechanisms.

First, possible trips in the system were identified, considering topography. For each trip, a CA with Strength 3 and Alphabet 2 was generated. For example, 0 if a trip does not stop at a station, and 1 otherwise. There was the same number of columns as the quantity of stations, and the least possible quantity of lines. Given that there has been research to determine the optimal CAs with the characteristics mentioned, existing CAs were used [13, 61].

When analyzing BRTS passenger trips, it was found that a large majority of users of a specific route reaches three feasible stations (i.e. 'stop stations' on a route). This supports the decision to use a CA of Strength 3, given that it is guaranteed that by using all the rows of the CA, each row corresponding to a route, then all different combinations of three stations are covered. Each row of the CA represents one of the routes applicable to a specific trip. A feasibility validation needed to be performed for those routes that included fewer than two physical stations, whose bus stops were discarded. At the end of the generation and refinement process, a database was created with routes for the BRTS, formed by compilation of all lines in the CAs. Finally, to facilitate the previously described process of *SolutionComplete*, an 'easy' route was included in the routes database for each possible trip. On an 'easy' route, buses stop at all stations along the route.

3.3.2 Dijsktra for Short Routes

Each harmony generated by MOGBHS-TNDFSP represents a new configuration of routes. Hence, it is necessary to recalculate the map of shortest routes and movements in the system that users consider for their trip each time that a harmony is generated or modified. Due to the simplicity of implementation, the Dijkstra algorithm was selected as the base algorithm to implement the search process for the shortest routes.

Source files in SIMAN provide the simulation model created in ARENA with the configuration of routes given by a harmony. Information was taken regarding trip times and the availability of routes and trips from each station. For each station in the system, a search for shortest paths towards all other stations in the system was implemented using

Dijkstra. This consists of exploring all the paths from an origin – in this case, the station selected – and associating other stations in the system with the shortest routes found to reach them. In addition to the costs of trips between stations, the additional cost involved with making the transfer between routes was considered. After the search for the shortest routes from an origin, the matrix of temporary routes was updated.

Finally, when the matrix of shortest routes is completed, SIMAN is updated by changing the appropriate code in the simulation files. Thus, it was possible during the implementation of the simulation that the passenger-type entities could consider the map of shortest routes corresponding to the configuration of actual routes.

3.3.3 MOGBHS-TNFSP

Once the algorithm, MOGBHS-TNDFSP, generated a harmony with a configuration of routes, the SIMAN code files with the description of selected routes were updated, and the matrix of shortest routes was calculated. The modified MOGBHS responsible for finding the best configuration of frequencies of bus departures for the routes selected in the harmony being evaluated was implemented, and denoted as a MOGBHS-TNFSP adaptation.

For the MOGBHS-TNFSP internal algorithm, a harmony was defined as a set of *N* departure intervals, where *N* corresponds to the number of routes selected in the current harmony of the MOGBHS-TNFSP external algorithm. **Figure 8** illustrates a harmony of the MOGBHS-TNFSP algorithm.

0	1	2	…	$N-2$	N-1
I_0	I_1	I_2	…	I_{N-2}	I_{N-1}

Figure 8 A solution for TNFSP generated by MOGBHS.

In contrast to MOGBHS, in this proposed version, the random generation of values for the variables that form the harmonies was done – both for the generation of new harmonies in the main cycle of the algorithm and in the generation of the initial population – between two limits given by the domain of the problem. These limits were the minimum interval between buses on the same route and the maximum interval between buses that made the same trip.

Making a connection with the MOGBHS algorithm (defined in Section 4.1), the value for parameter NVariables of MOGBHS corresponded to the number of routes defined in the harmony. Similarly, a harmony of MOGBHS was equivalent to a configuration of bus departure frequencies; that is, an arrangement of size NVariables. **Figure 9** shows the pseudo-code of MOGBHS-TNFSP. The function *Evaluate* of MOGBHS-TNFSP is illustrated in **Figure 10**. In this figure, the function *ExecuteSimulation* was responsible for implementing the simulation by making use of the SIMAN engine. Finally, the function *ReadFromOut* reads the file that contained the results of the simulation to extract the values that correspond to the evaluation of the objectives selected. This was the average wasted capacity and the average total time that the passengers used the system.

The MOGBHS-TNFSP needs to provide only one best solution, which is the best configuration of frequencies. However, given the multi-objective optimization problem, the result is a set of best harmonies selected based on the performance with respect to two optimization objectives. Hence, it is necessary to include one automatic function (i.e., without human intervention) to select the best solution based on the Euclidean distance. This involves three steps:

1) Having a set of harmonies generated by MOGBHS-TNFSP that is evaluated before the objectives are minimized. The solutions that correspond to the first Pareto front are identified.
2) The Euclidean distance of each solution that form the first Pareto front up to the origin are calculated**.**
3) The solution with the least Euclidean distance to the axis is selected, that is, the solution that has the best performance in all the objectives to be minimized.

With the addition of this function for selecting the best solution (line 27 of **Figure 10**), the pseudo-code of MOGBHS-TNFSP needed the inclusion of the lines from **Figure 11**.

```
PopulationRandomInitialize(HM)
NonDominatedOrderCalculate(HM)
CrowdingDistanceCalculate(HM)
Sort(HM)
for I = 1 to NIter do
    NewHarmony = null // create a new Harmony
    PAR = PARMin + (((PARMax - PARMin) / NIter) * I)
    for J = 1 to NVariables do
        if Random(0,1) < HMCR then
            NewHarmony.Variable[J] = HM.Harmonies[Random(0,HMS)].Variable[J]
            if Random(0,1) < PAR then
                NewHarmony.Variable[J] = FromBestHarmony(HM, Random(0, NVariables))
            end if
        else
            NewHarmony.Variable[J] = Random(MinIntervalTime, MaxIntervalTime)
        end if
    end for
    if InPopulation(HM, NewHarmony) = false AND IsViable(NewHarmony) then
        Evaluate(NewHarmony)
        Add(HM, NewHarmony)
        NonDominatedOrderCalculate(HM)
        CrowdingDistanceCalculate(HM)
        Sort(HM)
        RemoveTheWorst(HM)
    end if
end for
return the best solution of the Pareto front in HM // best solution for MOGBHS-
TNFSP
```

Figure 9 MOGBHS for TNFSP.

```
IntervalString = “”
for k = 0 to NVariables do
    IntervalString = IntervalString + “,” + Harmony.Variable[k]
end for
ReplaceInFile(“Transport.mod”, “$intervalos”, IntervalString)
ExecuteSimulation()
ReadFromOut(Harmony.Evaluations)
```

Figure 10 Function *Evaluate* for MOGBHS-TNFSP.

```
PopulationRandomInitialize(HM)
…
end for
for J = 1 to HMS do
    if HM[J].Front == 0 then
        HM[J].euclidianDistance = CalculateEuclideanDistance(HM[J].Evaluations)
    else
        HM[J].euclidianDistance = Float.maxNumber()
    end if
end for
SortByEuclideanDistance(HM)
Return HM[1]
```

Figure 11 Selection of the best solution for MOGBHS-TNFSP.

4 Experiments

4.1 Comparisons for an Actual TNDFSP Problem

A simulation model was created for discrete events of the bus rapid transity system of the City of Pereira, Colombia, known as Megabus. A prototype software was implemented that included the proposed algorithm and the process of refining the algorithm's parameters. A set of evaluations with the best parameters found was implemented. Finally, the results of MOGBHS were compared with the results of two of the best algorithms found in the literature for multi-objective optimization, NSGA-II and MOEA/D.

For this study, Megabus had 37 stations. Of these, 16 enabled transfers between buses that were going in opposite directions, known as double stations; 18 stations did not have transfers; and three were exchangers, or stations at which users could make transfers to other types of transportation systems. The system of trunk lines was approximately 27 km long. There were three routes whose frequency of bus departures varied between 4 and 17 min, depending on the time of day and the day of the week. The service is provided every day from 5:30 a.m. until 8:30 p.m., and carries close to 200,000 users daily. **Figure 12** shows a map of the current Megabus routes and stations.

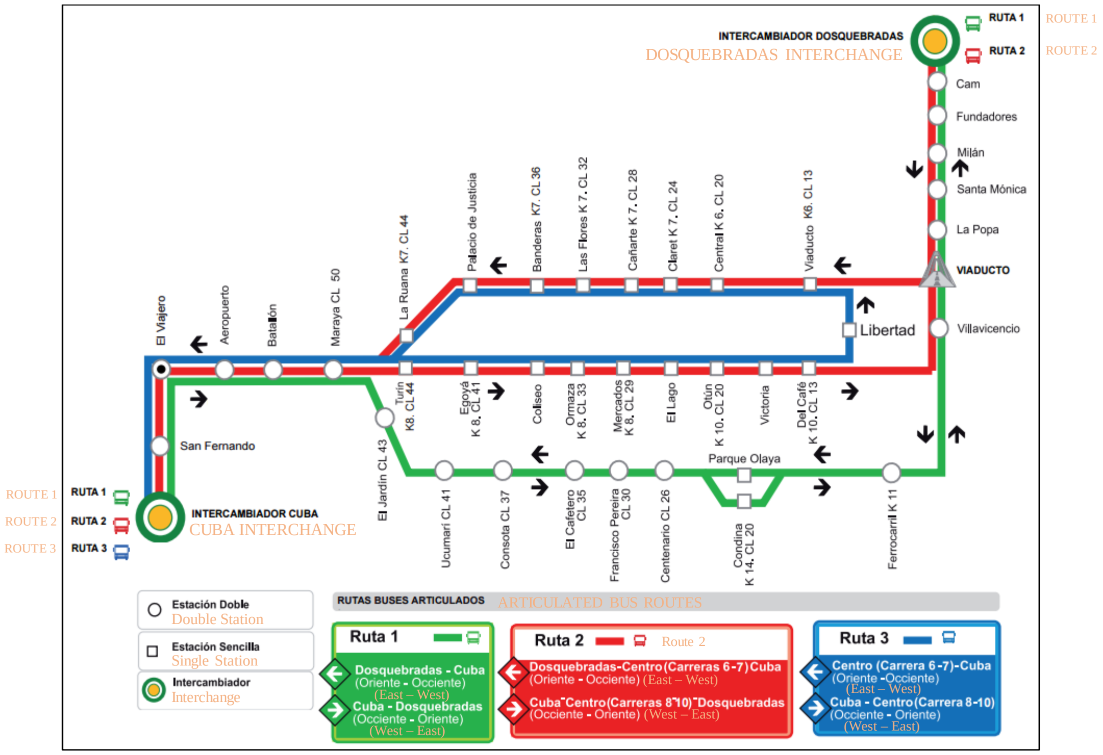

Figure 12 Map of the routes and stations of the simulation model for Megabus, a bus rapid transit system in Pereira, Colombia (taken from [64]).

To find the strengths and weaknesses of the proposed algorithm, the average results of 30 executions of MOGBHS, NSGA-II, and MOEA/D were performed on identical simulation models. These involved parameters resulting from refining each algorithm, and with the same criteria to evaluate their performances before the objectives were selected. To attain reproducibility of the experiments and independence of the implementations for all algorithms, various seeds (initial numbers) were used for the function to generate random numbers in each execution. To achieve the normalization of the values of 'usage time of the system' for the tests, the maximum and minimum values of 90 and 1 minutes, respectively, were established for usage time in the system. This was enough time to make two complete trips (one round trip) for the longest trip in the system in an easy route. The maximum capacity of users per bus was 140. This value corresponded to the maximum wasted capacity, while the minimum was set at zero.

4.2 BRTS Simulation Model and Database of Routes

As indicated earlier, the BRTS simulation model for discrete events was implemented using ARENA [57, 63]. For modelling and calibration of the simulation model, the configuration of the off-peak hours in the morning during midweek (i.e., workdays) was set as 8:30 a.m. to 11:30 a.m. For this period, the configuration of the real system currently operating is presented in **Table 2**.

Table 2 Configuration of Off-peak Hours in the Morning during Workdays for Megabus Routes [64]

Route	Longitude (km)	Cycle Time (min)	Fleet	Frequency (min)
R1	22.72	68	8	8.5
R2	22.32	75	10	7.5
R3	14.74	51	6	8.5

Once the configuration of the routes was obtained, the value of variable *Routes* was determined, and the shortest routes were calculated among all the stations in the system, using the Dijkstra algorithm adapted to facilitate the decision-making of user-type entities during the trip (variable *ShortRoutes* into the SIMAN code). To account for the lack of information about the travel patterns of users between the different stations, a uniform probability for selecting of destination stations was defined. It was established that all the buses were determined to have the same capacity. In addition, a waiting time of 30 sec in the stations was set for users to get on and off buses. These assumptions limited the validity of the solutions for real applications. Hence, future research should pursue the use of real data and the development of a formal process to calibrate the model.

The simulation model in this study made it possible to measure the average usage time of users in the system and the percentage of wasted capacity in light on the configuration of routes and departure frequencies. Based on the simulation model, which was configured with the operation plan used in the real BRTS for the off-peak hours in the morning and workdays, the following results were obtained:

- Average usage time in the system: 26.26 min
- Average wasted capacity: 91.7 seats

After finalizing the modelling process, SIMAN code files were generated, and some modifications to these files were performed so that they could serve as input for MOGBHS. In addition to the SIMAN files, the algorithm MOGBHS-TNDFSP requires a database of possible routes for the BRTS. Hence, the CAs were used in the following way:

(1) Five possible trips were identified based on the topography of the BRTS.
(2) Considering the number of stations in each trip, a list of CAs was selected to generate possible routes for these trips. These were:
  - CA (24; 30, 2, 3)
  - CA (26; 41, 2, 3)
  - CA (26; 42, 2, 3)
  - CA (26; 43, 2, 3)
(3) A CA with 42 columns was used in two trips, which were similar with regard to the number of stations involved.
(4) Routes that were not logical or infeasible were cleaned out, among others; this included those with less than two real stations, i.e., having a different start and finish station during the trip.
(5) Routes found for each trip were grouped together and inserted in the database. A total of 119 routes were obtained.
(6) In the case of invalid solutions, five easy routes (one per trip) were added to facilitate the process of completing the solution (*SolutionComplete*) for a total of 124 possible routes.
(7) Unique identifiers for all the routes in the database – as well as a marker for the easy routes – were established to facilitate their search in the process of completing the solution.

4.3 MOGBHS Prototype

To test the proposed algorithm, a software prototype was created that uses as input the SIMAN files resulting from the BRTS model and the database of feasible routes generated by CAs. This enables parameterization and implementation of MOGBHS-TNDFSP to determine the best configurations of bus frequencies. The prototype was developed in C#, using Visual Studio .NET 2012 as a development environment, and supported by Microsoft SQLServer 2008 for the database of routes as well as by SIMAN included in ARENA version 14.

4.4 NSGA-II and MOEA/D

Considering that state-of-the-art technology involves NSGA-II and MOEA/D as among the best algorithms for multi-objective optimization [12, 65, 66], these were implemented to solve the TNDFSP in a similar way as MOGBHS. That is, the problem of the configuration of routes and frequencies was addressed at two different levels. In addition, the simulation model of discrete events for a BRTS was used to evaluate the fitness of the solutions generated. This enabled comparing the performance of the algorithms.

4.5 Parameters Tuning

A tuning process was performed for some of the algorithmic parameters. Considering recommendations from previous studies regarding configurations for HS and GBHS, fixed values were defined for PAR Minimum (0.01) and the number of improvisations (100). To refine these parameters, 27 combinations were defined, with recommended values for HMS, HMCR, and PAR Maximum. After implementation of 27 tests and the analysis of the results, the configuration used for experimentation was HMS = 10, HMCR = 0.7, PAR Minimum = 0.01, PAR Maximum = 0.35, and Number of improvisations = 100.

In a parameter tuning process such as the one performed for MOGBHS, the parameters for NSGA-II and MOEA/D were determined. The configuration selected for the implementation of NSGA-II included: Population size = 10, Probability of crossing = 0.7, Probability of mutation = 0.333, and Number of generations = 11. The configuration selected for the implementation of MOEA/D included: Population size = 10, Neighbor size = 7, CR = 0.5, F = 0.5, and Iterations = 11.

4.6 Analysis of the Results

After implementing MOGBHS, NSGA-II, and MOEA/D with the parameters resulting from the tuning process, their performances were analyzed based on results for the optimization objectives. Further, an analysis was conducted to determine the best solutions found by MOGBHS before the actual configuration of the BRTS was selected for the study. Finally, the best solutions found by the algorithms were compared, and their execution times were analyzed.

4.6.1 Comparison of Performance Prior to the Optimization Objectives

The 30 harmony memories resulting from the executions of MOGBHS were joined, each one with 10 harmonies. Their performances were evaluated based on the selected objectives, which were transit capacity wasted and usage time of the system. The same process was performed for NSGA-II and MOEA/D. **Figure 13** illustrates that the configurations provided by the MOGBHS-TNDFSP algorithm proposed in this paper obtained the best results (minimum values) during performance evaluation of the selected objectives.

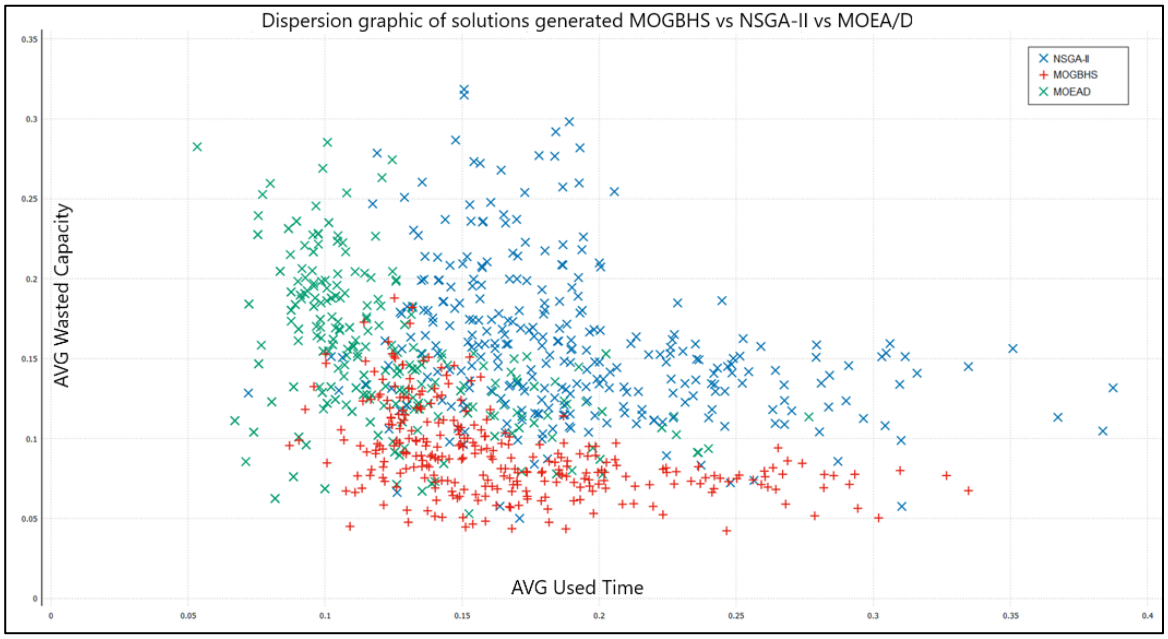

Figure 13 Dispersion of solutions provided by 30 implementations of MOGBHS, NSGA-II, and MOEA/D algorithms.

Table 3 compares the average and the standard deviation for evaluations of harmonies/solutions given by the algorithms, based on the selected objectives. An advantage can be seen in the minimization of both objectives for MOGBHS in comparison to NSGA-II. Compared to MOEA/D, it can be observed that MOGBHS won in minimization of average usage time and lost in minimize average wasted capacity. **Table 4** presents the standard deviation of the evaluations of harmonies given by MOGBHS and the solutions generated by NSGA-II and MOEA/D. It shows that the executions of MOGBHS were more compact or less dispersed than those by NSGA-II. Harmonies in MOGBHS were more compact in the wasted capacity than MOEA/D but less compact in usage time than MOEA/D.

Table 3 Averages of Evaluations by the Objective of the Solutions for MOGBHS, NSGA-II, and MOEA/D

Algorithm/Objective	Average usage time (hours)	Average usage time (minutes)	Average wasted capacity (percentage)	Average wasted capacity (bus seats)
MOGBHS	0.165404028	15.89	**0.0915333**	**12.81**
MOEA/D	**0.123442783**	**11.10**	0.1490842	20.87
NSGA-II	0.190890808	18.18	0.159106	22.27

Table 4 Standard Deviation of Evaluations by the Objective of the Harmonies/Solutions Given by MOGBHS, NSGA-II, and MOEA/D

Algorithm/Objective	Average usage time (hour)	Average usage time (minutes)	Average wasted capacity (percentage)	Average wasted capacity (bus seats)
MOGBHS	0.047403769	2.84	**0.027997591**	**3.92**
MOEA/D	**0.036081823**	**2.17**	0.046257313	6.48
NSGA-II	0.050960594	3.06	0.047636891	6.67

Figures 14 and **15** present the convergence of the solutions generated by the algorithms based on the optimization objectives. It is clear that MOGBHS generates harmonies with better values for both objectives than NSGA-II. Moreover, a greater advantage can be seen in the minimization of wasted capacity. In comparison, MOEA/D and MOGBHS generate harmonies with better wasted capacity values over all the generations. Howver, MOEA/D

generates solutions with better usage time values. In the last generations, the values of this objective converged to similar values.

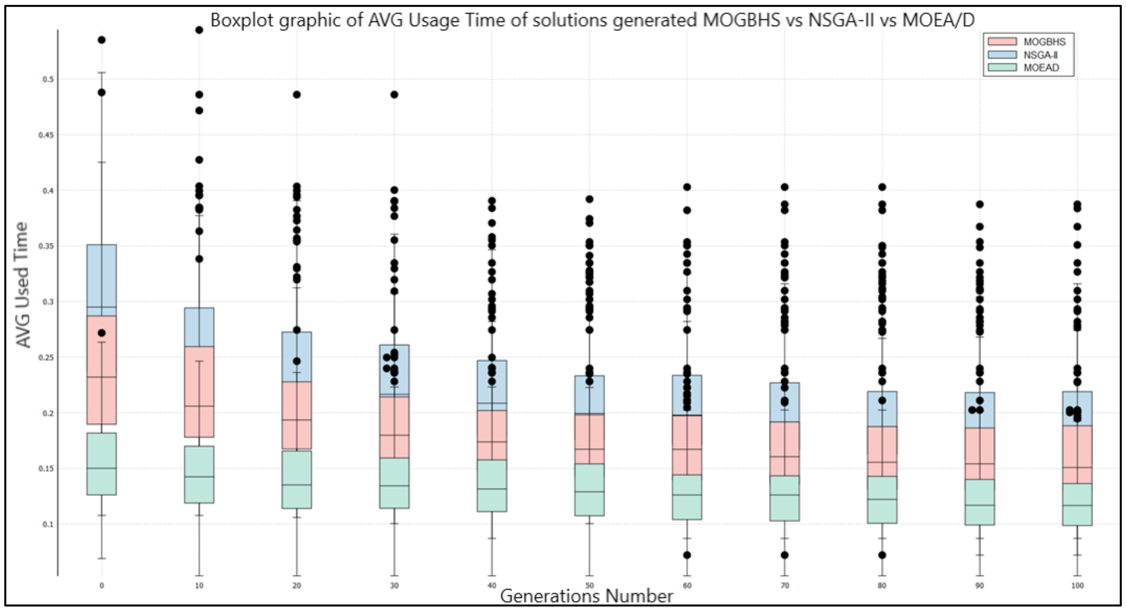

Figure 14 Convergence of MOGBHS, NSGA-II, and MOEA/D algorithms for the minimization of usage time.

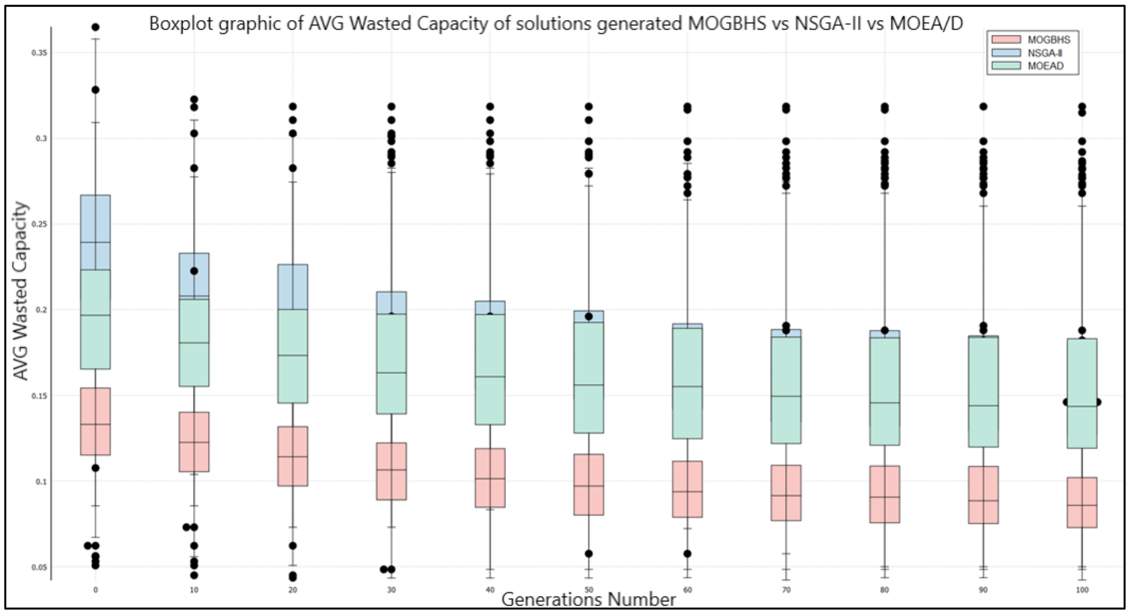

Figure 15 Convergence of MOGBHS, NSGA-II, and MOEA/D algorithms for the minimization of wasted capacity.

The average usage time and the average wasted capacity were compared to the harmonies generated by MOGBHS for the 30 executions (tests). The same measures were taken during the execution of the simulation model configured with the operational plan for the real BRTS during the off-peak hours of workday mornings, as described in Section 4.2. A considerable improvement was observed both in the minimization of the average usage time and the average wasted capacity. **Table 5** presents the comparison of these measures.

Table 5 Evaluation of Objectives with Solutions Given by MOGBHS vs. Evaluation of the Simulation Model

Source/Objective	Average usage time (min)	Average wasted capacity (bus seats)
MOGBHS	15.89	12.81
Simulation model (base line)	26.26	91.27

4.6.2 Determination of the Best Solutions

Afterwards Pareto fronts were organized both for the complete group of harmonies generated by MOBGHS and the group of solutions generated by NSGA-II and MOEA/D. The best solutions of the 30 executions of the three algorithms were obtained. and are presented in **Tables 6**, **7,** and **8**. In these tables, the lDs of the routes correspond to their unique identifiers in the route database. Easy routes are marked with an *f*, and the frequencies correspond to values (in minutes) for each of the routes.

Table 6 Best Harmonies Generated in 30 Executions of MOGBHS

Quantity of Routes	IDs of Routes	Frequencies (min)
5	64, 60, 95, 110, 23(f)	25, 16, 10, 28, 2
4	123(f), 11, 20, 23(f)	27, 18, 3, 2
4	100, 2, 103, 23(f)	25, 2, 29, 3
4	21, 38, 103, 23(f)	20, 11, 9, 2
4	92, 64, 108, 23(f)	14, 25, 14, 2
4	7, 103, 23(f), 106	26, 23, 3, 3
3	23(f), 123(f), 19	14, 4, 2
4	123(f), 7, 95, 23(f)	24, 16, 4, 3
Average: 4.0		Average: 12.78

Table 7 Best Solutions Generated in 30 Executions of NSGA-II

Quantity of Routes	IDs of Routes	Frequencies (min)
5	47, 8, 103, 52, 26	25, 14, 23, 19, 5
5	50, 0, 111, 113, 23(f)	25, 19, 16, 16, 6
6	114, 9, 80, 64, 81, 7	21, 18, 24, 16, 10, 8
6	101, 9, 80, 64, 81, 7	25, 15, 24, 27, 12, 5
4	53, 96, 109, 23 (f)	13, 27, 25, 4
Average: 5.2		Average: 17.0

Table 8 Best Solutions Generated in 30 Executions of MOEA/D

Quantity of Routes	IDs of Routes	Frequencies (min)
4	118, 109, 11, 23(f)	24, 23, 2, 29
5	118, 23(f), 11, 123(f), 0	9, 12, 20, 25, 5
4	62, 7, 103, 23(f)	23, 9, 15, 4
5	11, 16, 123, 11, 23(f)	26, 11, 8, 4, 28
4	53, 12, 53, 98(f)	24, 19, 18, 5
Average: 4.4		Average: 15.0

Regarding the best routes generated by the algorithms, the average in the quantity of routes selected by MOGBHS was 4.0, the MOEA/D algorithm selected an average of 4.4 routes, and NSGA-II selected an average of 5.2 routes. The average frequency by MOGBHS was 12.78 min, by MOEA/D was 15 min, and by NSGA-II was 17.0 min. This justifies the improvement in usage time by MOGBHS.

4.6.3 Comparison of Execution Times

Given that each simulation model can spend between 1 and 5 sec, to a large extent, the execution times of MOGBHS, NSGA-II, and MOEA/D depended on the quantity of executions of each simulation in order to evaluate solutions. The algorithms were implemented with the same size of population/memory, and performed the same quantity of generations both in the first level for routes and in the second level for frequencies. However, the implementation times varied because only configurations with frequencies that could be covered by the available fleet had been evaluated. Otherwise, very high evaluation values were established; these were unwanted solutions because they did not comply with the restriction of the fleet size. Hence, the logic of the algorithms discarded them.

As shown in both **Table 9** and **Figure 16**, the MOGBHS algorithm took much more time in executing itself than did NSGA-II. However, if before evaluating each solution, the algorithms check for feasibility – taking into consideration the available fleet, the total trip times of the selected routes and their frequencies – and that more than 90% of the algorithm execution time was used in the simulations, it can be concluded that MOGBHS evaluated more solutions than NSGA-II. Therefore, MOGBHS generated and evaluated more configurations that were feasible. Finally, MOEA/D spent more execution time (approximately 30% more) than MOGBHS, but obtained better results for just one objective.

Table 9 Consolidated Execution Times (in Minutes) of MOGBHS, NSGA-II, and MOEA/D

Execution	MOGBHS	NSGA-II	MOEA/D	Execution	MOGBHS	NSGA-II	MOEA/D
1	1381	799	2185	16	1545	762	2143
2	1405	797	2040	17	1366	763	2061
3	1464	718	2240	18	1310	757	2149
4	1235	761	1974	19	1283	711	2106
5	1492	707	2144	20	1499	743	2028
6	1366	988	1956	21	1226	947	2333
7	1727	959	1890	22	1763	987	2739
8	1689	966	2631	23	1669	966	2234
9	1612	973	2303	24	1774	970	2166
10	1504	923	1990	25	1382	983	2135
11	1482	773	2103	26	1281	780	2191
12	1446	801	2416	27	1574	783	2092
13	1330	687	2063	28	1371	788	2153
14	1382	843	1979	29	1576	778	2164
15	1439	812	2002	30	1081	810	2076
				Average	**1455±168**	**835±101**	**2156±185**

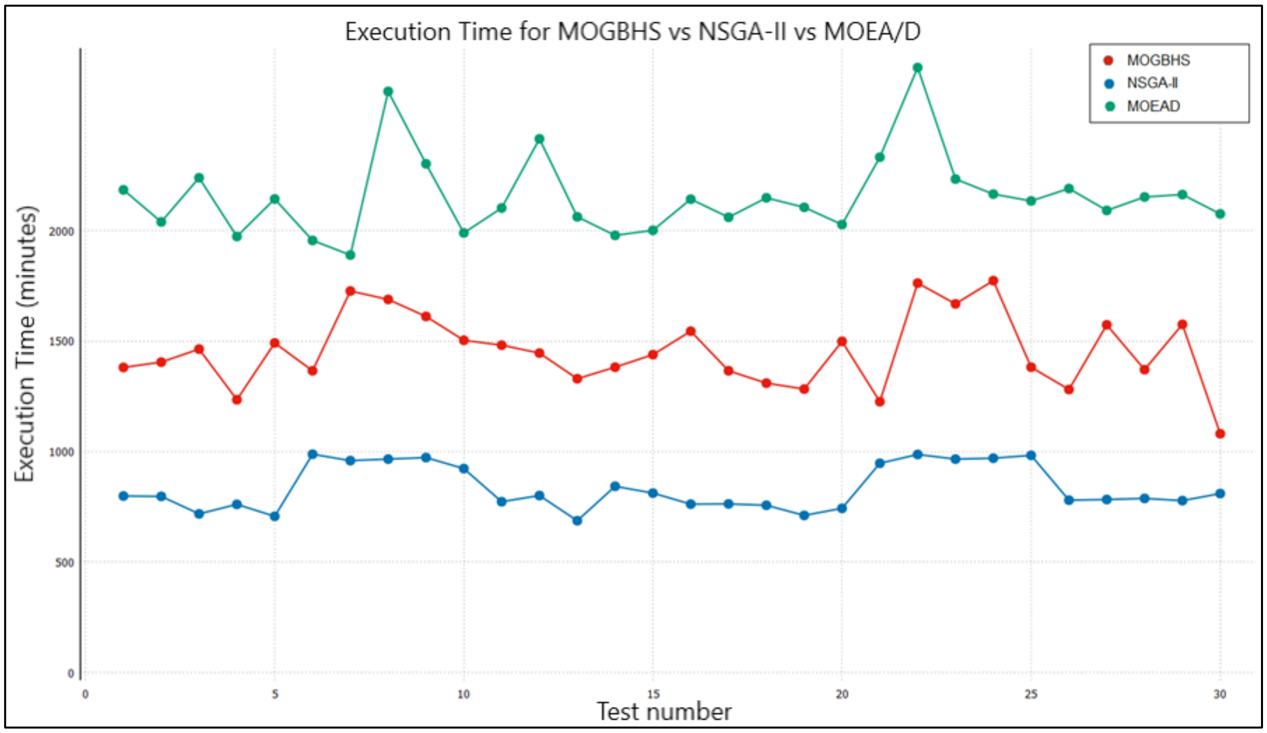

Figure 16 Execution times for MOGBHS, NSGA-II, and MOEA/D algorithms.

5 Conclusions and Future Work

The objective of this paper was to provide a mechanism that facilitates the configuration of routes and frequencies for BRTS in the context of TNDFSP. An adaptation of the global-best harmony search was proposed as part of a multi-objective optimization algorithm, using ordering by Pareto fronts and Crowding Distance. The proposed algorithm, Multi-objective Global Best Harmony Search, performed the following functions:

- It generated a random initial harmony memory,
- It evaluated the performance of each of the harmonies generated with respect to the objectives to be optimized, and
- It organized the harmony memory by Pareto fronts and Crowding Distance during a determined number of search iterations.

In each iteration, a new harmony was generated and evaluated, and was included in the actual harmony memory. The resulting harmony memory was organized by Pareto fronts and Crowding Distance, and the worst solution was eliminated based on that ordering. At the end of the iterative process, a set of harmonies with the best performances was obtained with respect to the optimization objectives.

With the intention of managing a large search space and the complexity of representing solutions for TNDFSP, a bilevel proposal of MOGBHS was performed. This involved an external level that handled the problem of selecting the best configurations of routes for the Transport Network Design Settings Problem and an internal level that selected the best frequencies for the set of routes generated by the first level (Transport Network Frequency Settings Problem).

To avoid the costs involved in evaluating the performance of the configurations of routes and frequencies generated by MOGBHS for an actual bus rapid transit system, this study used simulations of discrete events. SIMAN was selected as the simulation language, and ARENA was the tool used to create the BRTS simulation model. At the moment of evaluating a specific harmony, the SIMAN source code was modified to insert the configurations of routes and frequencies defined in the harmony and also the matrix of the shortest routes. The simulation model was executed using the SIMAN engine. Finally, information was extracted from the results of the execution to evaluate the performance of the harmony. In the prototype presented in this research project, an adaptation of the Dijsktra algorithm was used to generate the matrix of shortest routes, which are routes that each passenger would select to go from one station to another destination. However, given the independence of this component from the rest of the elements of MOGBHS, it could be replaced by any other algorithm that could fulfill the same objective.

To evaluate MOGBHS for finding optimal routes and frequencies, a simulation model based on discrete events was implemented for Megabus, the bus rapid transit system of the City of Pereira, Colombia. The result was a simulation model with 37 stations, including doubles, without transfers, and exchangers. This enabled measuring the average time that users spent on the system and the percentage of wasted capacity of the buses, depending on the configuration of routes and bus frequencies. After this process, given that MOGBHS-TNDFSP needs to have a set of valid routes applicable to the BRTS, and taking advantage of covering arrays for combinatorial problems, a set of these valid routes were used to define feasible solutions. This process considered all the possible trips on the BRTS, which in turn depended on the physical distribution of the stations and roads. In summary, the proposed approach provides a single comprehensive framework to solve the problem of defining routes and frequencies in an integrated transit system while considering multiple objectives by different key decision makers. The use of two levels allowed managing the complexity of the problem efficiently. The implementation of Covering Arrays enabled reducing the complexity in the route design by using an appropriate sampling of the needs of an actual BRT.

A set of tests of MOGBHS against NSGA-II and MOEA/D was completed to solve the problem of finding routes and frequencies for the actual configuration of stations and trips for Megabus. The objectives were to minimize both the total time of users in the system and the wasted capacity of the fleet. These tests were supported by the simulation model for discrete events implemented for the evaluation of the fitness of the solutions, and demonstrated that the solutions generated by MOGBHS were better for the two selected objectives compared to NSGA-II and MOEA/D. While the solutions generated by NSGA-II had an average total usage time of 18.18 minutes and an average wasted capacity per bus of 22.27 users, the harmonies generated by MOGBHS had an average usage time of 15.89 minutes and an average wasted capacity per bus of 12.81 users. The results for MOGBHS are promising, and very superior when compared to those for NSGA-II. In addition, the solutions generated by MOGBHS were competitive against MOEA/D, which had an average usage time of 11.1 minutes and an average wasted capacity of 20.87 bus seats.

The execution time of MOGBHS was superior to that of NSGA-II. However, given that the component that takes the most time during the implementation of these algorithms is the execution of the discrete event simulator – and considering that before the evaluation of each solution their feasibility was evaluated – the higher execution time means that MOGBHS executed a larger quantity of evaluations. Therefore, it generated a greater quantity of feasible solutions than NSGA-II.

Regarding future research, a version of MOGBHS will be developed to solve the Transport Network Design and Frequencies Settings Problem as well as other objectives, such as balancing the load of vehicles and minimizing the number of routes. In addition, a more exhaustive tuning parameter process is desirable that uses higher values of population size to search more diverse solutions in the Pareto front. Another important goal is to compare the proposed approach with other multi-objective algorithms, such as indicator-based MOEAs. MOGBHS can be used in other application domains, evaluated in multi-objective bi-level generic problems, and compared to other algorithms. Finally, experiments can be conducted regarding various configurations of covering arrays as well as several values of the strength parameter in order to define the best set of possible routes in a BRTS.

Acknowledgments

The work in this paper was supported by the University of Cauca, Colombia, under Project VRI-4043. We are especially grateful to Mrs. Julie Longo and Mr. Colin McLachlan for reviewing this document.

References

[1] R. Cervero, Bus Rapid Transit (BRT): An Efficient and Competitive Mode of Public Transport, IURD Working Paper 2013-01, (2013).

[2] H. Suzuki, R. Cervero, K. Iuchi, Transforming cities with transit: Transit and land-use integration for sustainable urban development, World Bank Publications, 2013.

[3] A. Mauttone, M.E. Urquhart, Diseño óptimo de recorridos y frecuencias para transporte público, in, Universidad de la República, Montevideo, 2007, pp. 1-13.

[4] V. Guihaire, J.-K. Hao, Transit network design and scheduling: A global review, Transportation Research Part A: Policy and Practice, 42 (2008) 1251-1273.
[5] C. Barnhart, G. Laporte, Handbooks in Operations Research & Management Science: Transportation: Transportation, Elsevier, 2006.
[6] R.Z. Farahani, E. Miandoabchi, W.Y. Szeto, H. Rashidi, A review of urban transportation network design problems, European Journal of Operational Research, 229 (2013) 281-302.
[7] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation, 1 (2011) 32-49.
[8] S. Luke, Essentials of Metahuristics, (2010).
[9] D.W.C. Abdullah Konak, Alice E. Smith, Multi-objective optimization using genetic algorithms: A tutorial, Reliability Engineering and System Safety 91 (2006).
[10] E. Mazloumi, M. Mesbah, A. Ceder, S. Moridpour, G. Currie, Efficient Transit Schedule Design of timing points: A comparison of Ant Colony and Genetic Algorithms, Transportation Research Part B: Methodological, 46 (2012) 217-234.
[11] M.G. Omran, M. Mahdavi, Global-best harmony search, Applied Mathematics and Computation, 198 (2008) 643-656.
[12] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, Evolutionary Computation, IEEE Transactions, 6 (2002) 17.
[13] J. Torres-Jimenez, I. Izquierdo-Marquez, Survey of covering arrays, in: Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on, IEEE, 2013, pp. 20-27.
[14] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobject optimization test Instances for the CEC 2009 special session and competition, in, University of Essex and Nanyang Technological University, 2009, pp. 1-30.
[15] B. Beltran, S. Carrese, E. Cipriani, M. Petrelli, Transit network design with allocation of green vehicles: A genetic algorithm approach, Transportation Research Part C: Emerging Technologies, 17 (2009) 475-483.
[16] A. Mauttone, M. Urquhart, A multi-objective metaheuristic approach for the Transit Network Design Problem, Public Transp, 1 (2009) 253-273.
[17] E. Cipriani, S. Gori, M. Petrelli, Transit network design: A procedure and an application to a large urban area, Transportation Research Part C: Emerging Technologies, 20 (2012) 3-14.
[18] J. Wang, G. Sun, X. Hu, Optimization of transit operation strategies: A Case Study of Guangzhou, China Annual Meeting of the Transportation Research Board, (2013).
[19] O.J. Ibarra-Rojas, R. Giesen, Y.A. Rios-Solis, An integrated approach for timetabling and vehicle scheduling problems to analyze the trade-off between level of service and operating costs of transit networks, Transportation Research Part B: Methodological, 70 (2014) 35-46.
[20] H. Zhao, R. Jiang, The Memetic algorithm for the optimization of urban transit network, Expert Systems with Applications, 42 (2015) 3760-3773.
[21] R. Giesen, H. Martínez, A. Mauttone, M.E. Urquhart, Multi-Objective Transit Frequency Optimization: Solution Method and Its Application to a Medium-Sized City, in: Transportation Research Board 94th Annual Meeting, 2015.
[22] F. Glover, Tabu search-part I, ORSA Journal on computing, 1 (1989) 190-206.
[23] F. Glover, Tabu search—part II, ORSA Journal on computing, 2 (1990) 4-32.
[24] A. Sinha, P. Malo, K. Deb, Approximated set-valued mapping approach for handling multiobjective bilevel problems, Computers & Operations Research, 77 (2017) 194-209.
[25] G. Eichfelder, Multiobjective bilevel optimization, Mathematical Programming, 123 (2010) 419-449.
[26] H. Li, Q. Zhang, Q. Chen, L. Zhang, Y.-C. Jiao, Multiobjective differential evolution algorithm based on decomposition for a type of multiobjective bilevel programming problems, Knowledge-Based Systems, 107 (2016) 271-288.
[27] M.J.o. Alves, S. Dempe, J.J. Jùdice, Computing the Pareto frontier of a bi-objective bi-level linear problem using a multiobjective mixed-integer programming algorithm, Optimization, 61 (2012) 335-358.
[28] K. Deb, A. Sinha, Solving Bilevel Multi-Objective Optimization Problems Using Evolutionary Algorithms, in: Evolutionary Multi-Criterion Optimization: 5th International Conference, EMO 2009, Nantes, France, April 7-10, 2009. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 110-124.

[29] A. Gupta, Y.S. Ong, An evolutionary algorithm with adaptive scalarization for multiobjective bilevel programs, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 1636-1642.
[30] M.J. Alves, J.P. Costa, An algorithm based on particle swarm optimization for multiobjective bilevel linear problems, Applied Mathematics and Computation, 247 (2014) 547-561.
[31] T. Zhang, T. Hu, X. Guo, Z. Chen, Y. Zheng, Solving high dimensional bilevel multiobjective programming problem using a hybrid particle swarm optimization algorithm with crossover operator, Knowledge-Based Systems, 53 (2013) 13-19.
[32] Y. Yin, Multiobjective bilevel optimization for transportation planning and management problems, Journal of Advanced Transportation, 36 (2002) 93-105.
[33] J. Xu, Y. Tu, Z. Zeng, A Nonlinear Multiobjective Bilevel Model for Minimum Cost Network Flow Problem in a Large-Scale Construction Project, Mathematical Problems in Engineering, 2012 40.
[34] K.S. Lee, Z.W. Geem, A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice, Computer Methods in Applied Mechanics and Engineering, 194 (2005) 3902-3933.
[35] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation, 188 (2007) 1567-1579.
[36] C.-M. Wang, Y.-F. Huang, Self-adaptive harmony search algorithm for optimization, Expert Systems with Applications, 37 (2010) 2826-2837.
[37] Q.-K. Pan, P.N. Suganthan, M.F. Tasgetiren, J.J. Liang, A self-adaptive global best harmony search algorithm for continuous optimization problems, Applied Mathematics and Computation, 216 (2010) 830-848.
[38] C. Cobos, D. Estupiñán, J. Pérez, GHS+ LEM: Global-best Harmony Search using learnable evolution models, Applied Mathematics and Computation, 218 (2011) 2558-2578.
[39] M. Khalili, R. Kharrat, K. Salahshoor, M.H. Sefat, Global Dynamic Harmony Search algorithm: GDHS, Applied Mathematics and Computation, 228 (2014) 195-219.
[40] V. Kumar, J.K. Chhabra, D. Kumar, Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems, Journal of Computational Science, (2013).
[41] K. Gao, Y. Zhang, A. Sadollah, R. Su, Optimizing urban traffic light scheduling problem using harmony search with ensemble of local search, Applied Soft Computing, 48 (2016) 359-372.
[42] M.A. Al-Betar, M.A. Awadallah, A.T. Khader, A.L.a. Bolaji, Tournament-based harmony search algorithm for non-convex economic load dispatch problem, Applied Soft Computing, 47 (2016) 449-459.
[43] B. Zeng, Y. Dong, An improved harmony search based energy-efficient routing algorithm for wireless sensor networks, Applied Soft Computing, 41 (2016) 135-147.
[44] N.S. Jaddi, S. Abdullah, A cooperative-competitive master-slave global-best harmony search for ANN optimization and water-quality prediction, Applied Soft Computing, 51 (2017) 209-224.
[45] S. Sivasubramani, K. Swarup, Multi-objective harmony search algorithm for optimal power flow problem, International Journal of Electrical Power & Energy Systems, 33 (2011) 745-752.
[46] Eckart Zitzler, Marco Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, TIK-Report, 103 (2001).
[47] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition, IEEE Transactions on Evolutionary Computation, 21 (2017) 440-462.
[48] C. Lücken, B. Barán, C. Brizuela, A survey on multi-objective evolutionary algorithms for many-objective problems, Comput. Optim. Appl., 58 (2014) 707-756.
[49] B. Yu, Z. Yang, C. Cheng, C. Liu, Optimizing bus transit network with parallel ant colony algorithm, in: Proceedings of the Eastern Asia Society for Transportation Studies, 2005, pp. 374-389.
[50] M. Bussieck, Optimal lines in public rail transport, Citeseer, 1998.
[51] R. Borndörfer, M. Grötschel, M.E. Pfetsch, A path-based model for line planning in public transport, Konrad-Zuse-Zentrum für Informationstechnik Berlin, 2005.
[52] W. Lampkin, P. Saalmans, The design of routes, service frequencies, and schedules for a municipal bus undertaking: A case study, OR, (1967) 375-397.
[53] S. Carrese, S. Gori, An urban bus network design procedure, Applied Optimization, 64 (2002) 177-196.
[54] Y.-J. Lee, V.R. Vuchic, Transit network design with variable demand, Journal of Transportation Engineering, 131 (2005) 1-10.

[55] S. Pattnaik, S. Mohan, V. Tom, Urban bus transit route network design using genetic algorithm, Journal of Transportation Engineering, 124 (1998) 368-375.

[56] V. Tom, S. Mohan, Transit route network design using frequency coded genetic algorithm, Journal of Transportation Engineering, 129 (2003) 186-195.

[57] R. Automation, Arena simulation software, Accessed November, 24 (2013).

[58] V. Bapat, D.T. Sturrock, The arena product family: enterprise modeling solutions: the arena product family: enterprise modeling solutions, in: Proceedings of the 35th conference on Winter simulation: driving innovation, Winter Simulation Conference, 2003, pp. 210-217.

[59] T.T. Allen, Introduction to Discrete Event Simulation and Agent-Based Modeling, Columbus: Springer, (2011).

[60] C.D. Pegden, R.P. Sadowski, R.E. Shannon, Introduction to simulation using SIMAN, McGraw-Hill, Inc., 1995.

[61] J. Torres-Jimenez, E. Rodriguez-Tello, New bounds for binary covering arrays using simulated annealing, Information Sciences, 185 (2012) 137-152.

[62] E.W. Dijkstra, A note on two problems in connexion with graphs, Numerische mathematik, 1 (1959) 269-271.

[63] T.T. Allen, Introduction to ARENA Software, in: Introduction to Discrete Event Simulation and Agent-based Modeling, Springer, 2011, pp. 145-160.

[64] A.P. Megabús S.A, Información operativa Megabús, in, 2016, pp. 4.

[65] J. Castro-Gutierrez, D. Landa-silva, J. Moreno Perez, Nature of real-world multi-objective vehicle routing with evolutionary algorithms, in: Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on, 2011, pp. 257-264.

[66] F.-A. Fortin, M. Parizeau, Revisiting the NSGA-II crowding-distance computation, in: Proceedings of the 15th annual conference on Genetic and evolutionary computation, ACM, 2013, pp. 623-630.