Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

A GRASP algorithm for the bus crew scheduling problem

David Pardo-Peña[a], David Álvarez-Martínez[a] and John Willmer Escobar[b]*

[a]*Department of Industrial Engineering, Universidad de los Andes, Colombia*
[b]*Department of Accounting and Finance, Universidad del Valle, Colombia*

C H R O N I C L E

Article history:
Received October 20 2023
Received in Revised Format December 18 2023
Accepted January 11 2024
Available online January 11 2024

Keywords:
Bus Crew Scheduling
Grasp Approach
Constructive Algorithm
Satisfaction Worker
Shifts

A B S T R A C T

This paper proposes a GRASP approach for solving the Bus Crew Scheduling Problem (BCSP) to find high-quality solutions within short computing times. The BCSP described the process related to the assignment of drivers and conductors to a bus company's regular daily operation of a mass transit system, seeking to minimize the cost of operation and, at the same time, the improvement of the working environment by considering the satisfaction of the drivers with the assigned shifts. The BCSP has drivers in charge of covering the demand for shifts, with an assignment that contains several constraints, such as minimum and maximum work blocks, minimum rest days, and shift sequences that must not be assigned. The former GRASP algorithm is proposed with a constructive procedure, a solution repair procedure, and two local search operators. Classical instances from the literature have been adapted for the shift assignment problem by adding a satisfaction variable. Besides, the proposed approach has been tested for a real company operating articulated and feeder vehicles. The results show that the satisfaction function adds value to the assignments, substantially improving the work environment and generating favorable results in terms of time and quality of the solution.

1. Introduction

Public passenger transportation is a determining factor in the city's way of life. A good transport system provides comfort in the mobility of people and becomes one of the main measures of quality of life. In this sense, public passenger transport plays a fundamental role in decision-making from a social, economic, and environmental point of view. Indeed, planning the transport network is extremely important for providing quality service and for the transport operator and the costs it assumes. Planning is complex and challenging, so this process is generally split into three phases covering strategic, tactical, and operational decisions. Strategic decisions are related to the design of the transportation network. Tactical decisions consider the set of frequencies and schedules of the transport network. Operational decisions are related to the problem of vehicle scheduling, shift scheduling, and shift assignment to drivers. However, each problem associated with each public passenger transport planning stage has been extensively studied because they are complex problems (mathematically and computationally). All these problems are classified as NP-hard problems. Therefore, each problem must be solved separately and sequentially (Byrne, 1973).

This paper is related to crew scheduling for transportation public belonging to the operational decisions. Whenever a bus is in operation, a driver must be assigned to it. Sometimes, the waiting time of a bus could be considered, for instance, the arrival of one journey and the departure of the next. However, when a crew leaves a bus, a relieving crew must be available to take it over (Smith, 1986). For the crew scheduling problem, it is necessary to consider that a crew leaves the bus to take a break or finish their duty, and another crew must take over the bus operation just starting their duty or having already worked on another bus (Smith, 1986). The Bus Crew Scheduling Problem (BSCP) is an integral part of the logistic management of bus

* Corresponding author
E-mail john.wilmer.escobar@correounivalle.edu.co (J. W. Escobar)
ISSN 1923-2934 (Online) - ISSN 1923-2926 (Print)
2024 Growing Science Ltd.
doi: 10.5267/j.ijiec.2024.1.003

planning and scheduling. The BSCP involves assigning staff (crew) to different scheduled bus services considering many resources that must be managed, including the complexity of allocating crew shifts, satisfaction level, unpredictability of traffic, crew compatibility, and availability—operational and practical constraints of the BSCP. Besides, the BSCP considers unpredictable events such as late crew sickness while on duty or absenteeism without prior notice (Cheng & Chang, 1999). When a specific unpredictable event occurs, changes must be made to the schedule or resource allocation. The times and crew schedule will remain the same, and only reallocation or reassignment is performed to cover the schedule (Shibghatullah et al., 2006a).

This paper proposes a GRASP approach for solving the operational planning stage, specifically the BSCP. The problem is solved because one of the main costs assumed by public passenger transportation companies is the operating costs related to the payroll of their drivers. This paper seeks to present an alternative solution within the context of articulated and feeder vehicle operators of massive transport systems (INTEGRA S.A.) and to test the effectiveness of the proposed methodology by adapting instances of Musliu (2006) addressing a similar shift assignment problem. The INTEGRA S.A. problem consists of a set of shifts with a particular demand that must be assigned to a group of drivers, and the drivers have a minimum and maximum number of working days per week and a minimum number of rest days. In addition, there are different types of shifts with different schedules, which must be reviewed before generating the shift assignment for the next day.

The rest of the paper is organized as follows. Section 2 shows the literature review related to the bus crew scheduling problem. Section 3 introduces the crew-scheduling problem for bus drivers (BSCP) in general terms and elaborates on constructing a mathematical model. Section 4 describes the proposed GRASP algorithm for solving the considered problem. Section 5 provides the computational experiments for the proposed approach. Finally, Section 6 gives conclusion remarks and suggestions for future research.

2. Literature Review

Several early works dealing with the crew scheduling problem, such as those proposed by Young & Wilkinson (1966), have proposed mathematical formulations and heuristic methods. The most natural formulation of the crew scheduling problem is the set covering or partitioning problem (Smith, 1986). Indeed, a large number of possible duties are performed, from which a crew schedule is selected covering all the shift works of the bus schedule at least once (set covering problem) or exactly once (set partitioning problem) (Smith, 1986). Smith (1986) studied a bus crew scheduling problem by three British bus companies. The problem is formulated as mixed integer linear programming using an extension of the set covering the problem and generating a large set of possible duties, minimizing the total cost. Some heuristic approaches are considered within the framework, reducing the set covering problem to a manageable size while still allowing good-quality schedules to be compiled. A branch and price algorithm for a real-life case of a BSCP has been proposed by Fournier (2009). This algorithm can solve a problem with 300 tasks in less than an hour, and instances with 1000 tasks can usually be solved in a few hours.

Shibghatullah et al. (2006a) model the bus crew scheduling problem with agents and study the feasibility of using the crew reassignment process to cope with unpredictable events. This work uses a Gaia methodology (Wooldridge et al., 2000; Zambonelli et al., 2003) for analyzing and designing agent-based models. Shibghatullah et al. (2006b) study the unpredictability problem of the BCSP and investigate how companies currently manage their schedules. This work proposes a set of requirements for a dynamic crew scheduling system that can reassign crew in real time given unpredictable events such as lateness for duty, sickness on duty, or crew absenteeism without prior notice. Kang et al. (2019) studied the BCSP considering mealtime windows for a single public transport bus route. Three Integer Linear Programming (ILP) models enable bus operators to solve their bus driver scheduling problems by directly invoking an available optimization solver such as CPLEX. A valid inequality approach that can generate valid cuts incorporating the CPLEX has been introduced. Cárdenas-Parra (2019) developed a mathematical model for the considered problem of this paper. However, this model does not consider the satisfaction of the drivers, showing in general feasible solutions to the problem, but with low satisfaction for the employees to whom the shifts are assigned, a situation that in the medium or long term may imply problems with the performance and permanence of the workers. Most proposed approaches are based on mathematical programming or hybrid approaches (combining heuristics and mathematical models) (Fores et al., 1999; Fores et al., 2002; Chen & Shen, 2013; Masbah et al., 2019). However, these approaches generally can solve small – to medium-sized instances with some limitations (Kwan et al., 2001; Li & Kwan, 2003).

Metaheuristic algorithms have found near-optimal solutions to the BSCP, considering large-scale problems and practical constraints. According to Song et al. (2015), metaheuristic approaches have three advantages: i) they are very efficient in searching large solution space, ii) they can find rapidly feasible solutions, and iii) they have their own methodical and strategic structure. Tabu search approaches for the BSCP have been proposed by Chen & Niu (2012a), Chen & Niu (2012b), and Shen & Kwan (2001). Chen & Niu (2012a) propose an approach for solving the BCSP considering early, day, and late duty modes with time shift and work intensity constraints. Besides, constraint with the least crew number of a specific duty has also been considered. An integer 0-1 model is proposed to minimize the expense of the total idle time of the crew for a circle bus line. Besides, a tabu search approach has been proposed to solve the problem. The same authors (Chen & Niu, 2012b) consider impartiality constraint on the BCSP, which is based on assurance to meet the time shift of trips and work intensity for the crew and establishes a crew scheduling model to minimize the total idle time. Also, a tabu search procedure is presented to solve the problem. Shen & Kwan (2001) propose a tabu search-based approach called HACS to handle the BSCP with windows of relief opportunities.

Genetic algorithms (GAs) are another prominent approach to solving major class crew scheduling problems. These algorithms have been proposed by Song et al. (2015). Song et al. (2015) propose a genetic algorithm (GA) with gene recombination for the BSCP. This work introduces a new method without using the potential shift set; instead, satisfied shifts generated from gene recombination in a genetic algorithm are employed. Experiments on real-life instances from Beijing Bus Group show the efficiency of the proposed approach. Ant colony algorithms for the BSCP have been proposed by Forsyth & Wren (1997), Ghoseiri & Morshedsolouk (2006), and Mazloumi et al. (2012). Finally, Simulated Annealing approaches for the BSCP have been introduced by Costa et al. (2013).

A Variable Neighborhood Search for a real-world case of BCSP has been proposed by Ma et al. (2016). This algorithm has been tested on a case study of two depots of the Beijing Public Transport Group. The results show that the former algorithm can reduce total driver costs by up to 18.1%, implying that the VNS algorithm may be a good optimization technique to solve the BCSP. Boyer et al. (2018) study the BCSP faced by urban bus transport agencies that must assign their resources (vehicles and drivers) to cover timetables generated at the tactical level. A mixed-integer linear programming model and a variable neighborhood search for this problem have been proposed.

An integrated framework considering the scheduling of buses and crew for local transportation public has been proposed by Ciancio et al. (2018). In the BSCP problem, each trip is assigned to a driver. The solution is performed with a classical sequential approach. This solution is then modified by changing the allocation of trips on vehicles in order to minimize the combined objective function. Perumal et al. (2021) introduce the integrated vehicle and crew scheduling problem for electric buses. An adaptive large neighborhood search that utilizes branch-and-price heuristics is proposed to tackle the considered problem. The proposed method is tested on real-life instances from public transport companies in Denmark and Sweden that contain up to 1109 timetabled trips. Moreno et al. (2019) propose a two-phase heuristic algorithm to solve the BCSP of a Megabus Bus Rapid Transit System. In the first stage, a division of the original schedules is performed using a recursive algorithm based on dynamic scheduling. In the second stage, work shift construction based on Fig. theory is performed using a pairing algorithm (i.e., matching). Öztop et al. (2017) consider a real-life BCSP determining the optimal number of different types of crew members with a minimum cost covering a given set of tasks regarding working and spread time limitations. Each driver has a spread time limit from the start time to the end time of the shift, including the idle times. Additionally, a driver cannot exceed the maximum total working time limit.

Recently, Xue et al. (2023) and Esquivel-González et al. (2023) have proposed methodologies for solving crew scheduling problems in metro and public bus transportation, respectively. Xue et al. (2023) consider the nature of the work of metro crews to study the equitability of crew planning. The train running diagram is split into segments by the duty points, including shifting stations, depots, and parking lots. Esquivel-González et al. (2023) consider the BSCP seeking to serve the most significant number of passengers possible instead of minimizing the schedule cost. A model and strategies for solving the problem are introduced, and clustering and Reoptimization procedures are proposed.

According to the literature review, we found different methodologies already developed for the BSCP. However, the satisfaction of the drivers has yet to be considered for the scheduling tasks. Therefore, our paper considers a GRASP method for the BSCP, considering the satisfaction of the drivers. This variable generates a stabilization of the staff and avoids the rotation of the crew. Indeed, this situation could be reached by generating a cyclical schedule that repeats at a specific time for a group of workers, complying with a matrix of requirements and constraints. Among the conditions to be met are the availability of employees to work the scheduled hours, the number of hours worked, the number of days off, the number of consecutive shifts, and specific constraints, such as the prohibition of certain consecutive shift assignments. This fact allows the companies to plan workers' schedules efficiently and effectively.

3. Problem description

Due to the combinatorial nature of the BSCP, an exact method for large instances (real-size instances) may have a high computational cost. Therefore, the main contribution of this paper is to find shift assignment strategies that allow obtaining feasible solutions within a reasonable time while maximizing driver satisfaction using an approximate optimization algorithm so that the problem can be applied or adapted to any real situation.

3.1 Related work

Most of the models used for the rostering solution are based on the Set Covering model proposed by Dantzig (1954) and its variations. Additionally, there is a strong relationship with the Generalized Assignment Problem (GAP) model proposed by Martello and Toth (1992), where n items are assigned to m units so that the total available resources are not exceeded, and the sum of the penalties related to the assignment are minimal. The formulation goes very well in parallel, but for a real situation, it would have to be adapted to the requirements of the company, and additional constraints would be added to the model:

$$(GAP) = Min \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}\, x_{ij} \tag{1}$$

subject to:

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad \forall i = 1, \dots, m \tag{2}$$

$$\sum_{i=1}^{n} a_{ij} x_{ij} \le b_j \qquad \forall j = 1, \dots, n \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \forall i = 1, \dots, m;\ \ \forall j = 1, \dots, n \tag{4}$$

The objective function (1) considers the minimization of assignment costs considering the binary decision variable x_{ij} (which takes the value of 1 when element i is assigned to a unit j) to a cost matrix c_{ij}. Eq. (2) refer to the fact that only one element i can be assigned to a unit j, and all elements are assigned. Eq. (3) is associated with a matrix a_{ij}, where the number of resources used is represented, and the logic is that the resources spent are not greater than the available resources of each unit.

$$min\ Z = \sum_{j \in S} c_j x_j \tag{5}$$

subject to:

$$\sum_{j \in S} a_{ij} x_j \le 1 \qquad \forall i = 1, \dots, m \tag{6}$$

We can observe the general set covering model of Dantzig (1954), which aims to cover all the rows of the matrix a_{ij} using the same cost of subsets of the columns. Eq. (5) considers the objective function where the cost associated with the selected columns is minimized and Eq. (6) restricts the coverage of each of the rows to at least one column.

3.2 Mathematical formulation

We proposed a mathematical model based on the formulation proposed by Cárdenas-Parra (2019) for the BCSP but considering the satisfaction level of the drivers. We have tested the proposed model on instances of Musliu (2006) with constraints similar to the considered problem. The rostering model is mathematically formulated as an integer linear model with a three-subindex decision variable and a two-subindex auxiliary variable:

Sets and Variables

i	Set of available workers, $i = 1,2, \dots, I$.
j	Set of roster horizon days, $j = 1,2, \dots, J$.
k	Set of shift types, $k = 1,2, \dots, K$
x_{ijk}	Decision variable that takes the value of 1 if worker i is assigned to day j in shift k, and 0 otherwise
y_{ik}	Auxiliary variable that takes the value of the number of shift types k that worker i performs during the roster

Parameters

P_S	Number of shifts that can be assigned to a worker
D_k	Shift k duration
JO	Maximum time allowed that can be assigned to a worker in a roster.
L	Maximum number of days that can be assigned to a worker in a time horizon
Req_{jk}	Number of workers required in a day j in a shift k
Min_i	Minimum number of days that must be assigned to a worker i for a given horizon
c_k	Cost associated with shift k type

Proposed General Mathematical Model

$$max\ Satisf.(\%) = \frac{Satisfied\ shifts - Unsatisfied\ shifts}{Shifts\ to\ be\ assigned} \tag{7}$$

subject to:

$$\sum_{k=1}^{K} x_{ijk} \leq P_s \qquad \forall i,j \tag{8}$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk} * D_k \leq JO \qquad \forall i \tag{9}$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk} \leq L \qquad \forall i \tag{10}$$

$$\sum_{i=1}^{I} x_{ijk} \geq Req_{jk} \qquad \forall j,k \tag{11}$$

$$\sum_{j=1}^{J}\sum_{k=1}^{K} x_{ijk} \geq Min_i \qquad \forall i \tag{12}$$

$$\sum_{i=1}^{I} x_{ijk} \leq y_{ik} \qquad \forall i,k \tag{13}$$

$$x_{ijk} \in \{0,1\}, y_{ik} \geq 0 \qquad \forall i,j,k \tag{14}$$

The objective function (7) considers the satisfaction level of the drivers. This satisfaction level can be represented by quantifying the number of favorable shifts for the driver since many of them have other constraints such as study, the time they would like to dedicate to their family or other activities. The objective function is defined in terms of the shifts the drivers dislike and the total shifts. A weight of 1 in the objective function was assigned to all shifts assigned to a driver comfortable with the shift and a weight of -1 if the driver was uncomfortable. In this case, a score is calculated from the satisfied and dissatisfied, as explained above. It is divided by the maximum number of satisfied drivers that we could hypothetically have, where all shifts are assigned to drivers who like the shift. In addition, the assumption was made that the shifts to which the driver is indifferent do not add or subtract from the objective function. However, it could be an interesting topic for further research: How much impact does indifference have on satisfaction? We can find cases where there is much indifference with little dissatisfaction and cases with much satisfaction but also much dissatisfaction. Note that we could find such solutions with the same level of satisfaction but need to know which one is better. Constraints (8) limit the total number of shifts that a driver could make. Constraints (9) and (10) limit the maximum time per day and the maximum number of days to be assigned a driver, respectively. Equations (11) determine the number of workers required per day. Constraints (12) ensure the minimum number of days to be assigned to a driver. Eq. (13) generate the relationship between the variables of the assignment and the number of shifts. Finally, constraints (14) are related to the nature of the variables.

4. Proposed methodology

A GRASP-based methodology is proposed with a constructive procedure, a solution repair mechanism, and two local search operators (INSERT and SWAP operators) for the BSCP. The constructive procedure relaxes the constraint of the maximum working days that can be assigned to the driver. Then, it applies a repair and balancing strategy, thus allowing the exploration of promising solutions. The constructed solutions (repaired and balanced) are improved through an exhaustive local search using two neighborhoods. The balancing mechanism reallocates the shifts to level the workload and recover the feasibility of the solution in terms of the relaxed constraint.

4.1 GRASP algorithm

The proposed GRASP algorithm consists of two phases: the constructive and the local search phases, for this it uses the $itera$ parameter that allows us to calibrate the number of times that both phases will be repeated (line 2). First, it checks if the $itera$ parameter has already exceeded the limit of solutions without repair and if no feasible solution has been found, it turns on the construction repairer (lines 3 and 4). Then the shifts are assigned using the constructive algorithm (line 5). If the constructive assignment is promising, it enters the improvement phase (line 7) where the result of the assignment is validated to see if it can improve the incumbent (lines 8,9,10,11). To do so, the solution must be checked for feasibility considering the number of assignments made and the number of shifts (line 6).

Algorithm 1 GRASP

Parameters: $itera$: Total number of iterations, $alpha$: size of the restricted list of candidates, $limit$: Proportion of iterations during which no repair is performed.

Input: Dict $aday$: dictionary with list of available drivers per day, **Dict** $proh$: dictionary with list of prohibited drivers according to the day and shift type

Output: Dict *t_incumbent*: Shift assignment dictionary, **List** *worked_incumbent*: List of days worked by driver, **Integer** *obj_incumbent*:

```
1.  $repair = False$
2.  $\boldsymbol{For}\ i = 1\ \boldsymbol{to}\ itera\ \boldsymbol{do}$
3.          $\boldsymbol{if}\ i \geq itera * limit\ \ and\ \ \boldsymbol{then}$
4.                  $repair = True$
5.          $\boldsymbol{Solution}\ \ t, worked, obj \leftarrow constructivo(alpha, repair)$
6.          $\boldsymbol{if}\ \boldsymbol{sum}(worked) = \boldsymbol{len}(t)\ \boldsymbol{then}$
7.                  $\boldsymbol{Solution}\ \ t, worked, obj \leftarrow busquedaLocal(\ t, worked, obj)$
8.                  $\boldsymbol{if}\ obj > obj\_incumbent\ \ \boldsymbol{then}$
9.                          $obj\_incumbent = obj$
10.                         $t\_incumbent = t$
11.                         $worked\_incumbent = worked$
12.   $\boldsymbol{Return}\ t\_incumbent$
```

4.2 Constructive Algorithm

The constructive algorithm is divided into three phases: a pseudo-random procedure (line 1), a repair process (line 3), and a balancing procedure (line 5). As the construction relaxes the constraint of maximum working days, the idea is to make a restricted list of candidates based on the number of days the drivers have worked as a strategy to narrow down the constraint and balance the shift allocation in a certain way. With these lists, drivers are randomly selected to be assigned to shifts, checking that the other restrictions are met. Then there is the repair phase, which is only performed if it is activated; there are no shifts to be assigned, and at least a percentage of the shifts have been assigned (line 2). In this phase, we review all the drivers that can be assigned to the missing shifts and assign the first one found, all this in order not to lose the process of some solutions that are close to being feasible. Finally, it is verified that the solution is feasible to move on to balancing (line 4), where the workload is reviewed and shifts already assigned to other drivers are reassigned to balance the workdays until a solution is found that not only meets the maximum workload constraint but is also feasible. It should be noted that some solutions go through the repairer and balancing and are still not feasible because they are not suitable for balancing.

Algorithm 2 General Constructive

Parameters: $alpha$: size of the restricted list of candidates, $repair$: Parameter to activate the repairer

Input: Dict $aday$: dictionary with list of available drivers per day, **Dict** $proh$: dictionary with list of prohibited drivers according to the day and shift type

Output: Solution $t, worked, obj$

```
1.  $\boldsymbol{Solution}\ t, worked, ddia, proh \leftarrow pseudorandomConstruction(alpha)$
2.  $\boldsymbol{if}\ repair = True\ \boldsymbol{and}\ alpha * t \leq \boldsymbol{sum}(worked) < \boldsymbol{len}(t)\ \ \boldsymbol{then}$
3.          $\boldsymbol{Solution}\ t, worked, aday, proh \leftarrow repairFunction()$
4.  $\boldsymbol{if}\ \boldsymbol{sum}(worked) = \boldsymbol{len}(t)\ \ \boldsymbol{then}$
5.          $\boldsymbol{Solution}\ t, worked, aday, proh \leftarrow balancing()$
```

The pseudo-random algorithm checks if any assignment has been made each iteration as a stop condition (line 1). The idea is that it runs through the shifts several times because sometimes, the random choice of the driver is not feasible. Each iteration starts running through the shifts (line 3) and checks if the shift has already been assigned if it has a value of -1 (line 4). It then generates the restricted list of candidates from the available drivers on the day of the shift. Having the initial restricted list of candidates, it generates three other lists categorizing which drivers like the shift, which are indifferent, and which do not like the shift, then one of the three lists is chosen as the restricted list of candidates (line 5). The criteria for choosing the list changes as it finds no drivers to assign to shifts until it reaches the list of those who do not like it (lines 11-13). The list is checked to see if it contains any drivers to choose one randomly (lines 6-7). Before assigning the chosen driver to the shift, it should be checked that it is not prohibited for the type of shift to be assigned that day (line 8) to subsequently assign it (line 9) and update the lists of prohibited, available by day and shifts assigned by the driver (line 10). When it no longer finds drivers to assign to shifts, the algorithm stops. In summary, it determines which of the three lists to use, randomly selects drivers, and then assigns them to shifts.

Algorithm 3 pseudo-random construction

Parameters: $alpha$: size of the restricted list of candidates

Input: Dict $aday$: dictionary with list of available drivers per day, **Dict** $proh$: dictionary with list of prohibited drivers according to the day and shift type

Output: Solution $t, worked, obj$

```
1.  while $assigned = True$
2.      $assigned = False$
3.      For $type, day, requirement$ in $t$
4.          if $t[type, day, requirement\ ] = -1$ then
5.              Solution $list \leftarrow RCL(type, aday[day], worked, alpha)[v]$
6.              if len$(list) > 0$ Then
7.                  $chosen \leftarrow$ random.choice$(list)$
8.                  if $chosen$ not in $proh[day, type]$ then
9.                      $t[tipo, dia, requirement\ ] = chosen$
10.                     Update $aday, proh, worked, obj$
11.     if $assigned = False$ and $v < 2$:
12.         $added = True$
13.         $v = v + 1$
```

The detailed pseudocode of the algorithm that constructs the restricted list of candidates for the pseudorandom construct is presented below. First, the maximum and minimum days worked to generate the general restricted list of candidates are found (lines 1-5). Then, the general candidate-constrained list is generated (lines 6-8). Finally, splitting into three lists according to the drivers' preferences is performed (lines 9-15).

Algorithm 4 Restricted list of candidates (RCL)

Parameters: $alpha$: size of the restricted list of candidates, $type$: Type of shift to be assigned, $aday$: Dictionary with list of available drivers per day, $worked$: List of shifts assigned per driver,

Input: Dict $proh$: dictionary with list of prohibited drivers according to the day and shift type **Dict** $preference$: dictionary with tuple list of values of the shift type liked and disliked by each driver. Position 0 contains the type of shift liked and position 1 the type of shift disliked.

Output: Solution $list$: List containing 3 lists with drivers suitable for shift assignment, the first one containing drivers who like the shift, the second one containing drivers who are indifferent to the shift and the last one containing drivers who do not like the shift.

```
1.  For $i$ in $aday$
2.      if $worked[i] < min$ then
3.          $min\ = worked[i]$
4.      if $worked[i] > max$ then
5.          $max = worked[i]$
6.  For $i$ in $aday$
7.      if $worked[i] \leq$ math.ceil$(min\ + alpha * (max - min))$ then
8.          $RCL.$append$(i)$
9.  For $p$ in $RCL$
10.     if $preferences[p][0] = type$ then
11.         $list[0].$append$(p)$
12.     elif $preferences[p][1] = type$ then
13.         $list[2].$append$(p)$
14.     else
15.         $list[1].$append$(p)$
```

After the initial construction, it is verified that the solution is suitable and that there are no unassigned shifts. To do so, it enters the repair phase, where there are two essential criteria to start with the repair. First, the repair parameter must be valid, and second, the number of assigned shifts must be less than the total number of shifts and greater than the percentage of unassigned shifts (line 1). The percentage of minimum assignments to repair the solution was determined as the same parameter alpha of the RCL size. If this decreases, it is expected that more solutions should be repaired and, therefore, this percentage as well. The logic of the repair is to go through the available drivers for the day the shift that has yet to be assigned is needed and choose the first one that meets the conditions in terms of no constraint being breached (lines 5-7). In this case, it will not be considered whether the driver likes or dislikes the shift. After being assigned, the lists of banned, available per day, and assigned shifts per driver will be updated (line 8). If no suitable driver is found, the solution will not be repaired to avoid wasting computational time on that solution (lines 9-10).

Algorithm 5 Repair procedure

Parameters: $alpha$: size of the restricted list of candidates, $repair$: Parameter to activate the repairer

Input: Dict $aday$: dictionary with list of available drivers per day, **Dict** $proh$: dictionary with list of prohibited drivers according to the day and shift type

Output: Solution $t, worked, obj$

```
1.  $\boldsymbol{if}\ alpha * \boldsymbol{len}(t) \leq \boldsymbol{sum}(worked) < \boldsymbol{len}(t)\ \boldsymbol{and}\ repair = True\ \boldsymbol{Then}$
2.      $\boldsymbol{For}\ type, day, requirement\ \boldsymbol{in}\ t$
3.          $\boldsymbol{if}\ t[type, day, requirement] = -1\ \boldsymbol{then}$
4.              $assigned = False$
5.              $\boldsymbol{For}\ chosen\ \boldsymbol{in}\ aday[day]$
6.                  $\boldsymbol{if}\ chosen\ \boldsymbol{not\ in}\ proh[day, type]\ \boldsymbol{then}$
7.                      $t[type, day, requirement] = chosen$
8.                      $\boldsymbol{Update}\ aday, proh, worked, obj$
9.              $\boldsymbol{if}\ assigned = False\ \boldsymbol{then}$
10.                 $\boldsymbol{break}$
```

The last step of the construction corresponds to the balancing stage, where the workload is leveled to meet the constraint of maximum working days that can be assigned to a driver. The algorithm works similarly to the pseudo-random construction in the sense that it runs through the shifts several times until no more changes have been made and has an additional criterion that all shifts must be assigned (line 1). The logic of the algorithm starts from three important lists, $missing$ contains all the drivers that have not reached the maximum workload, $complete$ contains the drivers that already have the maximum workload, and $exceed$ contains the drivers that exceed the maximum workload. It scrolls through the shifts until it finds a shift that is on the $exceed$ list (llines3-5), then randomly chooses a driver from the $missing$ list (line 6) and verifies that it is suitable for assignment to the shift and meets all the constraints (line 7). After the driver is assigned (line 8), the $missing$, $complete$, and $exceed$ lists are updated, as well as the prohibited, available per day, and assigned shifts per driver lists (lines9-10).

Algorithm 6 Balancing

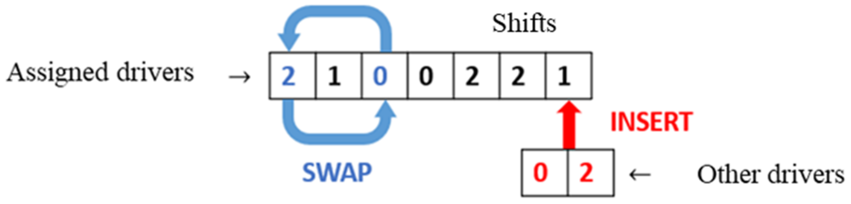
Input: Dict $aday$: dictionary with list of available drivers per day, **Dict** $proh$: dictionary with list of prohibited drivers according to the day and shift type, **List** $missing$: List containing all drivers who have not reached the maximum workload, **List** $complete$: List containing the drivers who already have the maximum workload, **List** $exceeded$: List containing the drivers who exceed the maximum workload

Output: Solution $t, worked, obj$

```
1.  $\boldsymbol{While}\ changed = True\ \boldsymbol{and}\ \boldsymbol{sum}(worked) = \boldsymbol{len}(t)$
2.      $changed = False$
3.      $\boldsymbol{For}\ type, day, requirement\ \boldsymbol{in}\ t$
4.          $actual = t[type, day, requirement]$
5.          $\boldsymbol{if}\ actual\ \boldsymbol{in}\ exceed\ \boldsymbol{then}$
6.              $chosen = \boldsymbol{random.choice}(missing)$
7.              $\boldsymbol{if}\ chosen\ \boldsymbol{in}\ aday[day]\boldsymbol{and}\ chosen\ \boldsymbol{not\ in}\ proh[day, type]\ \boldsymbol{then}$
8.                  $t[type, day, requirement] = chosen$
9.                  $\boldsymbol{Update}\ aday, proh, worked, obj$
10.                 $\boldsymbol{Update}\ missing, complete, exceed$
```

4.3 Local Search Algorithm

Once we have an initial feasible construction, we can improve it through two operators (INSERT and SWAP). The first one consists of going through the shifts and assigning different drivers to those who are currently assigned to improve driver satisfaction. The second operator consists of swapping between two shifts the drivers that are assigned in order to see if both drivers or one of them can be more satisfied. An explanation diagram of both is shown below:



5. Computational results

This section is divided into two parts. The first part presents the calibration of the parameters along with the performance of the GRASP algorithm in terms of the percentage of successful, unsuccessful, repaired, and unrepaired solutions. The second part compares the obtained results with the work proposed by Cárdenas-Parra (2019). The solution approach was coded in Python 3.8.3 and was executed on a computer with an Intel(R) Core (TM) i7-8750H CPU 2.20 GHz using a Windows 11 operating system and 16 GB of RAM.

5.1 Calibration of parameters and performance of the proposed methodology

Recall that the three calibration parameters of the GRASP algorithm are $itera$ which is the number of iterations our algorithm will perform, $alpha$ which is the size of our restricted list of candidates and $limit$ which is the percentage of iterations from where it will start repairing solutions in case it has not found any feasible one. $limit$ was calibrated under the assumption that, although we do not want it to repair all the time due to computational time issues in case it is required it should be 50% or a little more. For the number of iterations, we took more into account the execution time, increasing the instance size increases the computational effort. The largest instance has a size of 595 shifts and takes on average approximately ten seconds, considering that the company's instance has a size of 323 shifts, we can see in the Fig. that it would take approximately two seconds to execute. Although we could consider increasing the number of iterations for the smaller instances, it does not generate a great added value since the performance was quite good and the variation between 10-1000 iterations (increasing every 100) was minimal for the small instances.

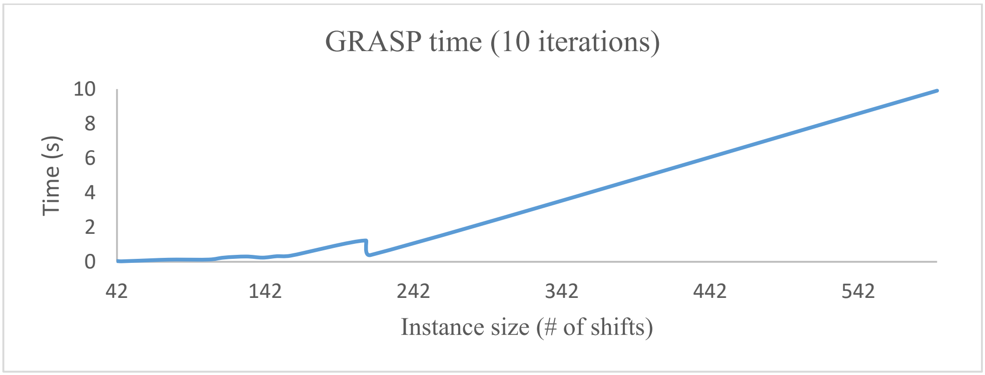


Fig. 1. GRASP run time with 10 iterations vs instance size

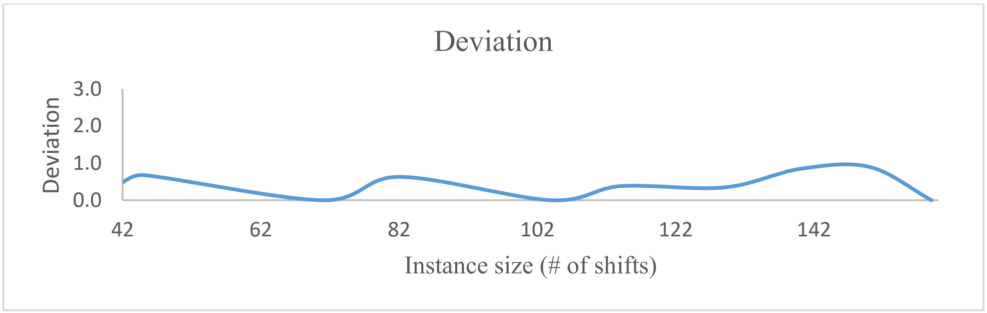


Fig. 2. Deviation by changing iterations

The alpha that defines the size of the restricted list of candidates was calibrated. In line with the logic of the algorithm, which consists of relaxing the workload constraint, the alpha should not be decreased too much since the constraint would be narrowed down again and the exploration of solutions would be too short. This parameter should be above 0.5 but at the same time we want to give a certain degree of restriction so that the solutions are not extremely unbalanced. For this reason, the calibration was focused mainly on the values of 0.6 and 0.9. For small instances, we found very good results with a value of 0.7 but the result with large instances was not favorable because they were not given enough freedom to explore solutions and required to be repaired all the time, this is computationally expensive and decreases the quality of the solutions. In this case, we sought to find an alpha where the largest instances were given enough freedom to explore. It was found that the optimal alpha is 0.84 since all instances have successful solutions and the repair percentage is minimal. It can also be observed that the rate of successful solutions of the proposed methodology is quite high with the parameters used. The repair strategy was only activated for the instance with 213 shifts. This indicates that for all of them, feasible solutions are being found approximately 80% of the time.

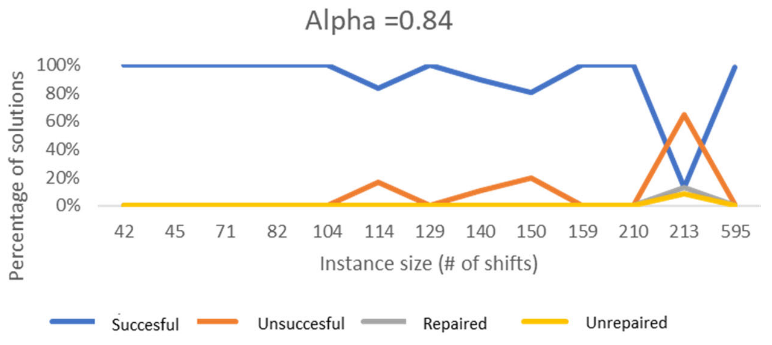


Fig. 3. Percentage of effective solutions with Alpha=0.84

Likewise, we can observe the computational effort and the percentage of improvement of the objective function of the constructive algorithm and the local search, where we observe that the local search has a computational cost almost more than three times that of the constructive one within the whole algorithm despite the complexity of the constructive one. The improvement of the objective function behaves oppositely; we observe that most of the impact comes from the constructive algorithm. However, as the instance grows, we see that it starts to be more significant in the local search for the improvement of the solution.

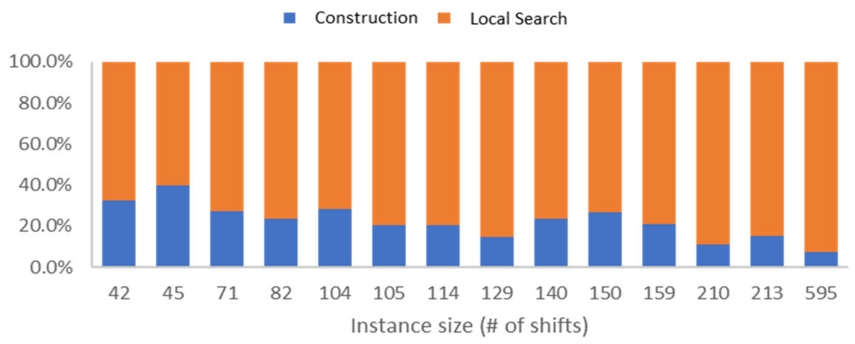


Fig. 4. Comparison of computational

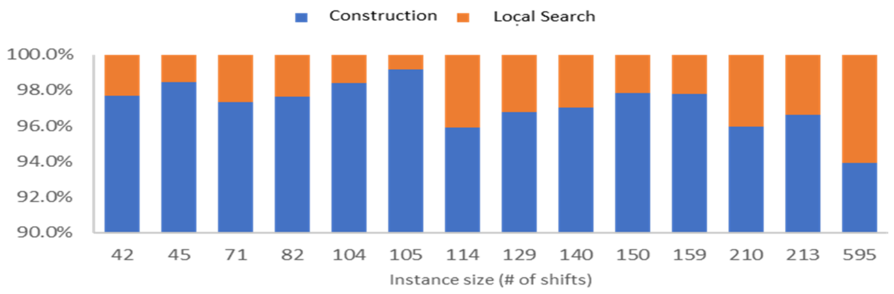


Fig. 5. Comparison of Objective function improvement

5.2 Comparison of the proposed methodology

A comparison has been performed with the work proposed by Cárdenas-Parra (2019) to validate the performance of the GRASP methodology. The algorithm was run 30 times for each instance to make the mean of the indicators balance and thus carry out a better analysis of their behavior. Regarding feasible and complete solutions delivered, the result of the proposed methodology outperformed the linear model by 93% versus 57%, for neither proposal was a feasible solution found for the ninth instance. The difference is that the linear model does not find a feasible solution for two other instances and delivers three incomplete feasible solutions. Another point of comparison is driver satisfaction. It is important to mention that the linear model did not have the objective of maximizing satisfaction in any way. However, it shows us that it is possible to give added value to the assignment of driver shifts without violating the company's restrictions and requirements. In the linear model, the number of satisfied, indifferent, and unsatisfied employees is balanced. However, the difference is evident when we see how, on average, it is possible to obtain a percentage of satisfied employees of 85% and, in many cases, a percentage of satisfied employees of over 90%.

Table 1
GRASP approach

Instance	Global Satisfaction	Satisfied	Indifferent	Unsatisfied	Feasibility
1	0.8985	0.9111	0.0763	0.0126	Yes (Complete)
2	0.9000	0.9032	0.0936	0.0032	Yes (Complete)
3	0.9089	0.9142	0.0805	0.0053	Yes (Complete)
4	0.9991	0.9991	0.0010	0.0000	Yes (Complete)
5	0.9437	0.9437	0.0563	0.0000	Yes (Complete)
6	0.8352	0.8539	0.1276	0.0186	Yes (Complete)
7	0.9757	0.9762	0.0233	0.0050	Yes (Complete)
8	0.4636	0.6436	0.1765	0.1800	Yes (Complete)
9	-	-	-	-	No
10	0.6038	0.6824	0.2389	0.0786	Yes (Complete)
11	0.4691	0.6433	0.1824	0.1743	Yes (Complete)
12	0.9119	0.9560	0.0000	0.0440	Yes (Complete)
13	0.9518	0.9519	0.0479	0.0002	Yes (Complete)
14	0.5506	0.6470	0.2567	0.0964	Yes (Complete)
Average	**0.8009**	**0.8481**	**0.1047**	**0.0476**	

Table 1
Work proposed by Cárdenas-Parra (2019)

Instance	Global Satisfaction	Satisfied	Indifferent	Unsatisfied	Feasibility
1	0.0667	0.3111	0.4444	0.2444	Yes (Complete)
2	0.0476	0.3571	0.3333	0.3095	Yes (Complete)
3	-0.1707	0.2195	0.3902	0.3902	Yes (Complete)
4	-	-	-	-	No
5	0.0845	0.3662	0.3521	0.2817	Yes (Complete)
6	-0.0991	0.283	0.3349	0.3821	Yes (1 Missing)
7	-0.0930	0.2636	0.3798	0.3566	Yes (Complete)
8	0.0400	0.3667	0.3067	0.3267	Yes (Complete)
9	-	-	-	-	No
10	-0.0439	0.307	0.3421	0.3509	Yes (Complete)
11	0.0429	0.3571	0.3286	0.3143	Yes (Complete)
12	0.1154	0.5577	0.0000	0.4423	Yes (3 Missing)
13	-	-	-	-	No
14	-0.0017	0.3418	0.3148	0.3434	Yes (1 Missing)
Average	**-0.0010**	**0.3392**	**0.3206**	**0.3402**	

The computing time of the proposed methodology is comparable with the work proposed by Cárdenas-Parra (2019) and, in the case of small instances, does not exceed 1 second. A clear difference in satisfaction was found between the two methods for the INTEGRA S.A. solution, as shown below:

Table 2
Solutions INTEGRA S.A.

Instance	Global Satisfaction	Satisfied	Indifferent	Unsatisfied	Time (s)	Feasibility
1	1	1	0	0	1.91	Yes (Complete)
2	0.0495	0.3746	0.3003	0.3251	1.07	Yes (Complete)
Average	**0.5248**	**0.6873**	**0.1502**	**0.1626**	**1.49**	

Finally, we show the behavior of the objective function of the best solution for the company INTEGRA S.A. (see Fig. 5) and for the large instance of Musliu (2006), which has 595 shifts to be assigned (see Fig. 4). The solution obtained for the company is quite efficient and involves little work for the methodology compared to the other. It can be seen in Fig. 4 that part of the solution is infeasible and corresponds to the constructive algorithm, where the aim is to explore solutions and then recover feasibility thanks to the relaxation of the constraint and the balancing strategy. However, in Fig. 5, we do not observe this behavior because, with the current constraints of the company, workers have enough free time, and a shift configuration where everyone is satisfied is possible. It would be possible to meet all requirements with fewer employees than currently.

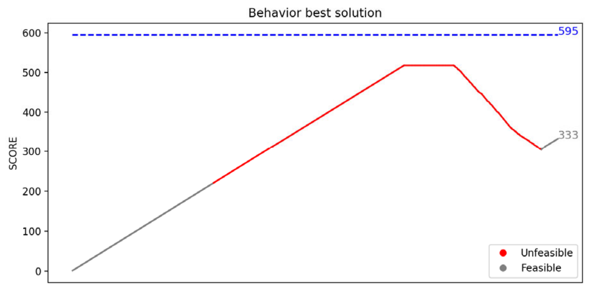


Fig. 6. Instance 14 Musliu

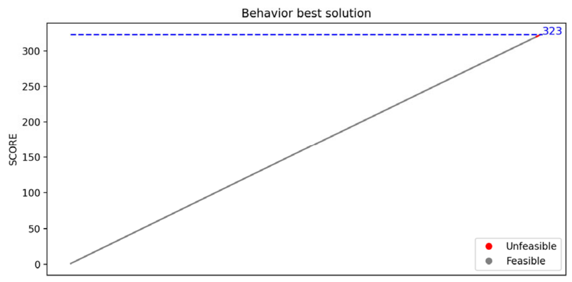


Fig. 7. Instance INTEGRA S.A.

The alternative presented is similar to the model presented by Cárdenas-Parra (2019) in speed and usefulness in terms of possible solutions found. However, the methodology proposed here gives added value to the solution of the problem of shift assignment to drivers in terms of staff satisfaction, generating solutions of better quality and welfare to the company's workers, an algorithm that will also indirectly help the company to minimize staff turnover.

6. Concluding Remarks

This paper proposes a GRASP approach for solving the Bus Crew Scheduling Problem (BCSP) to find high-quality solutions within short computing times. The BCSP described the process related to the assignment of drivers and conductors to a bus company's regular daily operation of a mass transit system, seeking to minimize the cost of operation and, at the same time, the improvement of the working environment by considering the satisfaction of the drivers with the assigned shifts. The proposed methodology resulted in an improvement in solution generation compared to previous work dealing with the same problems, finding a more significant number of feasible and complete solutions for the driver shift assignment problem. The GRASP algorithm proved to be 36% more efficient with a minimal difference in computational effort. As for the satisfaction maximization approach, it can be concluded that it implies a relatively significant improvement in terms of work environment, so it would be interesting to implement it in the company to observe the benefits that this can bring in the long term and to study other factors that can be added and taken into account in the calculation for the achievement of satisfaction. This paper opens the door to a new work proposal in which we will seek to improve this satisfaction function to make it more robust and delve into the information about the impact that brings a solution of this type compared to one that only seeks to comply with the restrictions and minimum parameters of a shift assignment in companies that have established their operation in this way. Finally, the algorithm developed is a flexible methodology that can be implemented not only in the company INTEGRA S.A. but also in other sectors of the industry where companies use shift assignment for their workers, such as health personnel in hospitals, workers in companies that provide public cleaning services or security service workers. It is interesting to observe how the calculation of satisfaction differs and changes depending on the sector of interest and how feasible it is to work it as proposed in this paper.

References

Boyer, V., Ibarra-Rojas, O. J., & Ríos-Solís, Y. Á. (2018). Vehicle and crew scheduling for flexible bus transportation systems. *Transportation Research Part B: Methodological, 112*, 216-229.

Byrne, J. L. (1973). A Note on the Bus Crew Scheduling Methods of Bennett and Potts. *Transportation Science, 7*(2), 203-206.

Cárdenas-Parra, K. (2019). Modelo matemático lineal flexible para la rotación de turnos de trabajos de los conductores del sistema de transporte masivo del área metropolitana centro occidente. Bachelor Thesis, Universidad Tecnológica de Pereira. Access: https://repositorio.utp.edu.co/items/6f1684ad-6104-4324-8acb-a818656fc682

Cheng, J. H., & Chang, Y. H. (1999). Application of a fuzzy knowledge base on bus operations under uncertainty. In FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No. 99CH36315) (Vol. 3, pp. 1355-1360). IEEE.

Chen, M., & Niu, H. (2012a). A model for bus crew scheduling problem with multiple duty types. *Discrete Dynamics in Nature and Society, 2012.*

Chen, M., & Niu, H. (2012b). Research on the scheduling problem of urban bus crew based on impartiality. *Procedia-Social and Behavioral Sciences, 43*, 503-511.

Chen, S., & Shen, Y. (2013). An improved column generation algorithm for crew scheduling problems. *Journal of Information and Computational Science, 10*(1), 175-183.

Ciancio, C., Laganà, D., Musmanno, R., & Santoro, F. (2018). An integrated algorithm for shift scheduling problems for local public transport companies. *Omega, 75*, 139-153.

Costa, M. F., Fiedler, N. C., & Mauri, G. R. (2013). Clustering Search and Simulated Annealing to solve the driver scheduling problem for timber transport. *Scientia Forestalis, 41*(99), 299-305.

Dantzig, G. B. (1954). A comment on Edie's "Traffic delays at toll booths". *Journal of the Operations Research Society of America, 2*(3), 339-341

Esquivel-González, G., Sedeño-Noda, A., & León, G. (2023). The problem of assigning bus drivers to trips in a Spanish public transport company. *Engineering Optimization, 55*(9), 1597-1615.

Fores, S., Proll, L., & Wren, A. (1999). *An improved ILP system for driver scheduling* (pp. 43-61). Springer Berlin Heidelberg.

Fores, S., Proll, L., & Wren, A. (2002). TRACS II: a hybrid IP/heuristic driver scheduling system for public transport. *Journal of the Operational Research Society, 53*(10), 1093-1100.

Forsyth P., & Wren A., (1997). An ant system for bus driver scheduling, *in Proceedings of the 7th International Workshop on Computer-Aided Scheduling of Public Transport, Boston, Mass*, USA, 1997.

Fournier, S. (2009). Branch-and-price algorithm for a real-life bus crew scheduling problem. ERPOSul.

Ghoseiri, K., & Morshedsolouk, F. (2006). ACS-TS: Train scheduling using ant colony system. *Journal of Applied Mathematics and Decision Sciences*, 2006.

Kang, L., Chen, S., & Meng, Q. (2019). Bus and driver scheduling with mealtime windows for a single public bus route. *Transportation Research Part C: Emerging Technologies, 101*, 145-160.

Kwan, R. S., Kwan, A. S., & Wren, A. (2001). Evolutionary driver scheduling with relief chains. *Evolutionary Computation, 9*(4), 445-460.

Li, J., & Kwan, R. S. (2003). A fuzzy genetic algorithm for driver scheduling. *European Journal of Operational Research, 147*(2), 334-344.

Ma, J., Ceder, A., Yang, Y., Liu, T., & Guan, W. (2016). A case study of Beijing bus crew scheduling: a variable neighborhood-based approach. *Journal of Advanced Transportation, 50*(4), 434-445.

Martello, S., & Toth, P. (1992). Generalized assignment problems. In Algorithms and Computation: Third International Symposium, ISAAC'92 Nagoya, Japan, December 16–18, 1992 Proceedings 3 (pp. 351-369). Springer Berlin Heidelberg.

Masbah, N. A., Nordin, S. Z., & Ahmad, R. (2019). Binary linear programming model in solving bus crew problem as tactical fixed task scheduling. *In Journal of Physics: Conference Series* (Vol. 1212, No. 1, p. 012030). IOP Publishing.

Mazloumi, E., Mesbah, M., Ceder, A., Moridpour, S., & Currie, G. (2012). Efficient transit schedule design of timing points: a comparison of ant colony and genetic algorithms. *Transportation Research Part B: Methodological, 46*(1), 217-234.

Moreno, C., Falcón, L., Escobar, J., Zuluaga, A., & Echeverri, E. (2019). Heuristic constructive algorithm for work-shift scheduling in bus rapid transit systems. *Decision Science Letters, 8*(4), 519-530.

Musliu, N. (2006). Heuristic methods for automatic rotating workforce scheduling. *International Journal of Computational Intelligence Research, 2*(4), 309-326.

Öztop, H., Eliiyi, U., Eliiyi, D. T., & Kandiller, L. (2017). A bus crew scheduling problem with eligibility constraints and time limitations. *Transportation Research Procedia, 22*, 222-231.

Perumal, S. S., Dollevoet, T., Huisman, D., Lusby, R. M., Larsen, J., & Riis, M. (2021). Solution approaches for integrated vehicle and crew scheduling with electric buses. *Computers & Operations Research, 132*, 105268.

Shen, Y., & Kwan, R. S. (2001). Tabu search for driver scheduling. *In Computer-Aided Scheduling of Public Transport (pp. 121-135). Berlin, Heidelberg: Springer Berlin Heidelberg.*

Shibghatullah, A. S., Eldabi, T., & Kuljis, J. (2006a). A proposed multiagent model for bus crew scheduling. *In Proceedings of the 2006 Winter Simulation Conference (pp. 1554-1561)*. IEEE.

Shibghatullah, A. S., Eldabi, T., & Rzevski, G. (2006b). The requirements for a dynamic bus crew scheduling system. *In Proceedings of the 10th International Conference on Computer-Aided Scheduling of Public Transport.*

Smith, B. M. (1986). Bus crew scheduling using mathematical programming (Doctoral dissertation, University of Leeds).

Song, C., Guan, W., Ma, J., & Liu, T. (2015). Improved genetic algorithm with gene recombination for bus crew-scheduling problem. *Mathematical Problems in Engineering, 2015*, 1 – 14.

Wooldridge, M., Jennings, N. R., & Kinny, D. (2000). The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and multi-agent systems, 3*, 285-312.

Xue, F., Zhang, X., Hu, P., Ma, X., & Chen, C. (2023). Metro crew planning with heterogeneous duty paths and period-cycle pattern considerations. *Computers & Industrial Engineering, 109354*.

Young, A., & Wilkinson, J. C. (1966). The scheduling of buses and crews by computer. In Public Transport Association Annual Conference, Scarborough.

Zambonelli, F., Jennings, N. R., & Wooldridge, M. (2003). Developing multiagent systems: The Gaia methodology. *ACM Transactions on Software Engineering and Methodology (TOSEM), 12*(3), 317-370.