IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received October 15, 2021, accepted November 7, 2021, date of publication November 16, 2021,
date of current version November 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3128871

A Matheuristic Algorithm for the Multiple-Depot Vehicle and Crew Scheduling Problem

EMILIANA MARA LOPES SIMÕES[1,2], LUCAS DE SOUZA BATISTA[3], AND MARCONE JAMILSON FREITAS SOUZA[4]

[1]Institute of Science and Technology, Federal University of Vales do Jequitinhonha e Mucuri, Diamantina 39100-000, Brazil
[2]Graduate Program in Electrical Engineering, Federal University of Minas Gerais, Belo Horizonte 31270-901, Brazil
[3]Department of Electrical Engineering, Federal University of Minas Gerais, Belo Horizonte 31270-901, Brazil
[4]Department of Computing, Federal University of Ouro Preto, Ouro Preto 35400-000, Brazil

Corresponding author: Emiliana Mara Lopes Simões (emiliana.simoes@ufvjm.edu.br)

This work was supported in part by the Brazilian agencies the Coordination for the Improvement of Higher Education Personnel (CAPES) under Finance Code 001, the National Council for Scientific and Technological Development (CNPq) under Grant 303266/2019-8, and the Minas Gerais State Foundation for Research Support (FAPEMIG) under Grant PPM/CEX/676-17; in part by the Federal University of Vales do Jequitinhonha e Mucuri (UFVJM); in part by the Federal University of Minas Gerais (UFMG); and in part by the Federal University of Ouro Preto (UFOP).

ABSTRACT This work addresses the multiple-depot vehicle and crew scheduling problem (MDVCSP). In MDVCSP, we deal with two NP-hard problems in an integrated way: the multiple-depot vehicle scheduling problem (MDVSP) and the crew scheduling problem (CSP). For solving the MDVCSP, we define the vehicles' operational routine and the workdays of the crews of a public bus transport system with multiple depots. Given the difficulty of solving real-world instances of the MDVCSP using exact mathematical methods, we propose a matheuristic algorithm for solving it. This matheuristic algorithm combines two strategies into an iterated local search (ILS) based framework: a branch-and-bound algorithm for solving the MDVSP and a variable neighborhood descent (VND) based algorithm for treating the associated CSPs. We compared the proposed ILS-MDVCSP with five approaches in the literature that use the same benchmark test instances. We also solved a real-world problem of one of Brazil's largest cities. For this problem, we proposed a formulation based on a time-space network to address the MDVSP subproblem. The results obtained showed the effectiveness of ILS-MDVCSP, mainly to deal with real-world and large-scale problems. The algorithm was able to solve the largest instances from the literature, for which there was no reported solution. Regarding the run time, as the instances' size increases, our approach becomes substantially less costly than the others from the literature. For the Brazilian instances, the ILS-MDVCSP saved, on average, the use of 12 vehicles per day and reduced by up to 15% the daily operational time of the vehicles.

INDEX TERMS Iterated local search, matheuristic, multiple-depot vehicle and crew scheduling, public transportation, time-space network, variable neighborhood descent.

I. INTRODUCTION

The planning process of the public bus transport system is highly complex, and therefore it is normally decomposed into several subproblems. We usually approach these subproblems in three stages: strategic, tactical, and operational.

Strategic planning addresses problems that involve long-term decisions. At this stage, we aim to meet users' demands and, at the same time, respect established budget limitations. Moreover, we specify the bus transit routes through the city.

The associate editor coordinating the review of this manuscript and approving it for publication was Juan Wang.

In tactical planning, we define the frequency with which the bus routes must be traveled and the timetable. The timetable is composed of the daily trips to be performed by the public transport company. Each trip has times and locations of the start and end.

Thus, given the timetable, we address the operational planning problems. As illustrated in Fig. 1, we have a vehicle scheduling problem (VSP), a crew scheduling problem (CSP), and a crew rostering problem (CRP).

The vehicle scheduling problem (VSP) consists of determining a daily operating routine for a vehicle fleet. Its objective is to make it possible to execute all timetable trips,

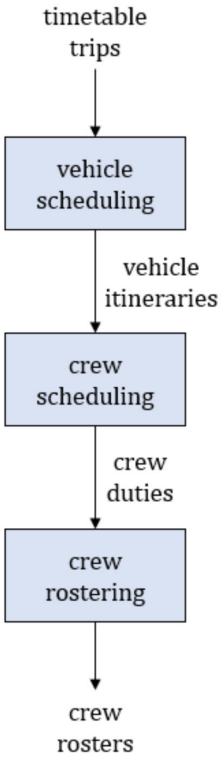


FIGURE 1. Operational planning of public urban bus transport.

reduce costs and, at the same time, respect all operational restrictions. As a result, vehicle itineraries are obtained.

There must be a crew (possibly a driver and a collector) responsible for each vehicle's activity in the fleet. So, the crew scheduling problem (CSP) deals with defining the workdays (duties) for the crews. When resolving the CSP, we sought to minimize labor costs; however, obligatorily in compliance with the operational and labor rules in force.

At last, there is the crew rostering problem (CRP). This problem consists of assigning duties to each crew to define their monthly work routine. Thus, in the CRP, specific rules regarding long periods are considered and therefore were not addressed in the daily schedule.

The division into stages presented above follows [1], [2]. These works provide a comprehensive review of models and approaches to solve the subproblems associated with each stage.

For a long time in the literature, proposals for sequential and independent resolution of the subproblems of the planning process of the public bus transport system predominated. However, this type of approach leads to suboptimal solutions for global planning. Thus, with the development of efficient computers and optimization techniques, more and more successful research has emerged to integrate some of these subproblems [3]–[9].

In the scope of operational planning, some authors deal with the integrated vehicle and crew scheduling problem (VCSP) [10]–[15]. In the VCSP, vehicle itineraries and the workdays of the crews are defined simultaneously. Solving VSP and CSP in an integrated way is useful, as there is a dependency relationship between these problems. Operating costs can even be an opposite relationship; that is, a characteristic favorable to a solution for the VSP does not always reflect satisfactorily in the CSP and vice versa.

In this work, we approach the VCSP with multiple depots. In the so-called multiple-depot vehicle and crew scheduling problem (MDVCSP), each timetable trip may be assigned to a subset of existing depots. In addition, each depot has its vehicles and crews. Therefore, solving more than one depot simultaneously (rather than distributing trips between depots and then solving each depot separately) makes the problem more flexible and more likely to result in better solutions.

The benefits of considering multiple depots were initially identified in solving the multiple-depot vehicle scheduling problem (MDVSP), and there is a vast literature that addresses this problem [16]–[20].

MDVCSP is an even more complex problem than MDVSP and has also received the attention of researchers [3], [21]–[26]. The works that address the MDVCSP differ significantly in terms of the assumptions considered. Gaffi and Nonato [21], for example, impose that a crew is assigned to the same vehicle during the whole duty. On the other hand, Mesquita and Paias [24] consider that a crew can change vehicles at any time at an appropriate local. There are still works that allow the exchange of vehicles, but only between work shifts of the crew [3], [22], [25], [26]. Thus, the particularities of the works often make it impossible to compare them.

This paper approaches the MDVCSP following the same assumptions considered in [3], [22], [23], and [26]. In addition, we also deal with a Brazilian real-world problem.

We propose a formulation for the MDVSP based on the time-space network elaborated by Steinzen *et al.* [3]. In the Brazilian problem that we address, trips are distributed throughout the day on the timetable. Therefore, we must guarantee a minimum daily stay in the depot for each vehicle in our formulation. This time is for the maintenance and cleaning of the vehicles. Steinzen *et al.* [3] and other authors [3], [22], [23], [26], [27] do not consider this situation. In the problem they address, there is a long interval of the day, in which no trip is scheduled on the timetable.

Furthermore, it is possible to combine our formulation for the MDVSP with a set partitioning formulation for the CSP and thus constitute a model for the MDVCSP.

MDVCSP is an NP-hard problem [3], [22]. As shown in [28] for the MDVSP and in [29] for the CSP, each of these problems is NP-hard. Therefore, we approach the MDVCSP through a matheuristic algorithm to address real-world instances, which generally involve a high number of variables.

Matheuristics are hybrid algorithms that combine mathematical programming and metaheuristic techniques. According to Raidl [30], this class of algorithms exploits the individual advantages of these methods and benefits from the synergy between them. Given its success

in solving several combinatorial optimization problems, this approach has been applied in some studies in the literature [31], [32].

Our matheuristic algorithm, named ILS-MDVCSP, uses the iterated local search (ILS) [33] framework for exploring the solution space of the MDVCSP. In turn, its solution process combines two methods: a branch-and-bound method to solve the MDVSP in optimality and a heuristic procedure based on the variable neighborhood descent (VND) [34] to treat associated CSPs.

To the best of our knowledge, only Steinzen *et al.* [27] addressed the same problem using a matheuristic. They deal with the MDVCSP using a hybrid evolutionary algorithm (EA) to assign trips to depots. In this context, an individual is a trip-depot vector where each trip is assigned to a single depot. In addition, these authors used mathematical programming techniques to define vehicle and crew schedules and thus evaluate an individual's fitness. The results showed that the hybrid EA overcame a sequential approach to solving the MDVSP and CSP. However, the hybrid EA achieved lower performance than the integrated resolution algorithms proposed in [35] and [22]. As well as Steinzen *et al.* [27], we compared the ILS-MDVCSP against the approaches proposed in [22] and [23] (the improved work of [35]). Unlike Steinzen *et al.* [27], our algorithm performed better for all instances.

To validate our algorithm, we perform experiments with benchmark test instances and compare the results against those of other well-known approaches from the literature. We also present and discuss the results obtained for a real-world problem of a city in Brazil, where the bus is the most popular form of public transport in large urban centers. We compare the companies' solutions (VSP and CSP sequential planning) with the solutions obtained from two integrated approaches: VCSP resolution and MDVCSP resolution. The results obtained showed:

1) The effectiveness of the matheuristic algorithm ILS-MDVCSP, mainly to deal with real-world and large-scale problems.
2) The integrated resolution of the VSP and CSP is more efficient than the traditional sequential resolution of these problems. Furthermore, the integrated approach with multiple depots is more effective than the one with a single depot.

Finally, it is worth noting that, together with population growth and the new challenges of urban mobility, the demand for an efficient, safe, quality, and cost-effective locomotion service has been increasing. Concurrently, alternatives to move in the cities have emerged. Therefore, the urban bus sector needs to adapt to this new competitive and demanding scenario. In this respect, the optimized definition of vehicle and crew schedules, as we propose in this work, is essential. It is directly related to the working conditions of the employed labor and involves most of the sector's operating costs.

The remaining of this work is as follows. In Section II we review some related works and in Section III define the MDVCSP. Section IV describes the literature formulations for the MDVSP and MDVCSP, which support our formulations. Section IV also presents the formulation we propose to deal with a Brazilian real-world problem. The problem-solving approach is detailed in Section V. Section VI presents and discusses the results obtained. Finally, concluding remarks are pointed out in Section VII.

II. RELATED WORK

VSP and CSP have long been studied extensively in the Operations Research (OR) literature. Among the precursors are the works [36]–[41]. In the 1980s, Ball *et al.* [10] already pointed out the advantages of addressing these problems in an integrated manner and proposed a heuristic resolution method for the VCSP.

Patrikalakis and Xerocostas [11] proposed the first mathematical formulation for the VCSP. However, this model was for illustrative purposes only, being computationally intractable [42]. Thus, the authors in [43] are considered the pioneers in proposing an integer programming formulation for the problem. This work gave rise to a series of studies developed by Freling *et al.* on the topic, such as [13], [42], [44]. The formulations presented in these works are similar and consist basically of two parts:

1) A quasi-assignment formulation based on a network to ensure the viability of the VSP;
2) A set partitioning formulation to ensure each vehicle activity is assigned to a duty.

With the transformations in urban public transportation resulting from population growth, social changes, and policies to encourage sustainability, recent research has addressed more complex variants of the VCSP. Perumal *et al.* [7] solved the VCSP with electric buses (E-VCSP). So, new restrictions are imposed, such as i) limited driving range of electric vehicles, ii) longer recharging times of vehicles, and iii) fixed charging locations. Boyer *et al.* [45] and Andrade-Michel *et al.* [46] addressed the VCSP by treating each vehicle and crew individually. They considered the compatibility between each pair of vehicle-driver (the driver has/hasn't the ability to drive the vehicle model), driver-line (driver knows or not the line route), and vehicle-line (the bus must have appropriate characteristics to meet the line). Thus, the VCSP solution is a feasible trip-vehicle-driver assignment for each line. Furthermore, to resolve the VCSP, Andrade-Michel *et al.* [46] considered the reliability of each driver (probability that he will not be absent from work) and the importance of each trip (measured by the number of passengers it serves). Amberg *et al.* [47] observed that interruption (delay) can occur in urban public transport due to heavy traffic or passenger behavior. In addition, this initial delay of a scheduled task can cause delays in other activities that use the same resources (vehicle and crew). Thus, Amberg *et al.* [47] solved the VCSP aiming to minimize operational costs and, at the same time, define robust vehicle and crew schedules. The robustness in this context concerns the ability of an integrated schedule to

prevent the spread of delays. These previous works and many others solved the VCSP considering that there was only one depot for managing vehicles and crews (see e.g., [14], [15], [48]). This problem is so-called single-depot VCSP (SDVCSP).

Medium-sized and large public transport companies usually have multiple depots to manage their vehicles and crews. In this scenario, we have the definition of the multiple-depot VCSP (MDVCSP).

Based on Freling *et al.* [13], Huisman *et al.* [22] proposed a model and algorithm to deal with the MDVCSP. The developed algorithm combined column generation with Lagrangian relaxation.

Huisman *et al.* [22] developed another model for the MDVCSP from the formulation proposed by Haase *et al.* [12] for the VCSP. However, their algorithm generated better results using the model based on Freling *et al.* [13].

Before Huisman *et al.* [22], only Gaffi and Nonato [21] addressed the MDVCSP. Gaffi and Nonato [21] used a model and algorithm similar to those presented in [22], but they consider particular constraints that facilitate the resolution of the problem [22].

Based on Freling *et al.* [13], [42], [44] and using one of the models proposed in [22], Borndörfer *et al.* [23] proposed a Lagrangian relaxation and column generation approach to the MDVCSP. For the solution of the Lagrangian relaxations, they used a proximal bundle method. Besides, they used the primal and dual information generated by this bundle method to guide a branch-and-bound algorithm to produce integer solutions.

Mesquita and Paias [24] proposed two formulations for the MDVCSP. The first (SP-VCSP) was similar to the one presented in [22] but with a smaller number of constraints and decision variables. This formulation combined two models: the multicommodity network flow model for vehicle scheduling and the set partitioning/covering model for crew scheduling. The second formulation (SPC-VCSP) was an extension of the previous one, and it replaced some set partitioning constraints by set covering constraints. This modification made the model more flexible and made possible situations in which the crew was allowed to change vehicles during their duty without increasing the complexity of the problem. Unlike Huisman *et al.* [22] and other authors [3], [23], [26], Mesquita and Paias [24] allowed a given crew to drive vehicles from any depot and not just the depot to which they belong. Furthermore, a crew could change from a vehicle to another at any time at an appropriate local. Therefore, the results in [24] cannot be directly compared with those of other studies that addressed the MDVCSP.

Steinzen *et al.* [3] presented a new formulation for the MDVCSP, based on the so-called time-space network. Until then, in the literature, all works used models similar to the one proposed in [22], based on the so-called connection-based network to address the MDVCSP. The network proposed in [3] was much smaller considering the number of nodes and arcs, which resulted in a mathematical formulation with a much smaller number of constraints and variables. The proposed solution methodology was similar to that presented in [22] and combined the generation of columns with Lagrangian relaxation.

For the tests, Steinzen *et al.* [3] used the instances available at [49] and generated larger ones using the same algorithm of Huisman *et al.* [22], which are available in [50]. According to Steinzen *et al.* [3], when considering other works in the literature, their results were the best in terms of processing time (when it was possible to compare) and quality of the solution generated (number of vehicles and crews employed).

As we mentioned earlier, Steinzen *et al.* [27] addressed the MDVCSP using a hybrid evolutionary algorithm (EA). In the proposed strategy, the hybrid EA was used to assign trips to depots and, in this way, transform the MDVCSP into several VCSPs. Both problems, MDVCSP and VCSP, are NP-hard. However, in VCSP, the associated vehicle scheduling problem can be solved in polynomial time, as it is the minimum cost flow problem. Thus, the hybrid EA used Lagrangian heuristics based on column generation to solve the VCSP and compute the individuals' fitness.

The solution approaches described in [27] and [3] provided the basis for Kliewer *et al.* [25] to solve the MDVCSP with time windows for scheduled trips (MDVCSP-TW). At MDVCSP-TW, the scheduled time for timetable trips is not fixed. That is, at MDVCSP-TW, the trip departure and arrival times are variables. This flexibility increases the number of trips compatible with each other and can reduce required vehicles and crews in the schedule. Kliewer *et al.* [25] compared the approaches a) traditional sequential planning of vehicles and crews, b) sequential planning extended with consideration of time windows in the vehicle scheduling phase, c) integrated planning, and d) integrated planning with time windows. The results indicated that the integrated planning approach with time windows was better than the others.

Ciancio *et al.* [51] solved the MDVCSP considering several real-world restrictions according to the European Union legal framework. These restrictions are so specific that their proposed problem is quite different from those found in the literature [51]. The authors proposed an ILS-based heuristic to address the MDVSP and a greedy heuristic combined with local searches to build an initial solution for the CSP. In integration, these solutions are modified by changing the trips' allocation on vehicles to minimize the combined objective function. The authors used instances proposed by themselves in the tests. The largest instance considered had 712 trips and 4 depots.

Horváth and Kis [26] proposed a mathematical formulation for the MDVCSP based on the model of Steinzen *et al.* [3]. Furthermore, they presented a branch-and-price procedure to approach the problem exactly. This approach used an efficient pricing method, some branching strategies, and a simple primal heuristic. According to the authors, this was the first work to propose an exact solution to the MDVCSP

defined in [22]. The proposed approach was able to efficiently solve only small instances, with 4 depots and 80 or 100 trips. Optimal solutions were found for 4 of the 20 instances considered and, for the others, the GAP of the solution found was defined concerning the lower bound obtained. The GAP was less than 0.5% for 7 instances.

Table 1 summarizes the approaches proposed to the MDVCSP from the literature review and present work. This table also describes the instances considered in the works' computational experiments.

Table 1 shows that the literature applies different optimization techniques to tackle the MDVCSP (mathematical programming, evolutionary algorithms, metaheuristics, and matheuristics). Our work proposes a matheuristic that combines branch-and-bound and variable neighborhood descent into an iterated local search-based framework to approach the MDVCSP. To the best of our knowledge, we are the first to address the largest instances from the benchmark instances widely used in the literature. Regarding the run time, as the instances' size increases, our approach becomes substantially less costly than the others from the literature. Furthermore, we propose a model based on a time-space network for the MDVSP that allows us to deal with the specifics of a Brazilian real-world problem. We show that the proposed algorithm can efficiently handle literature and real-world instances.

III. PROBLEM DEFINITION

In the *multiple-depot vehicle and crew scheduling problem* (MDVCSP), vehicle and crew schedules are defined simultaneously and must be mutually compatible. That is, the workday of some crew must cover each operational activity of a vehicle. Besides, we consider more than one depot for fleet and labor management. Therefore, we must also assign each vehicle and crew on the schedule to a single depot.

The rest of this section presents the concepts involved in defining the MDVCSP.

Timetable trips is the set of trips to be made daily by the public bus transit company. In this table are the features of the trips considered relevant for the operational planning, such as start time of the trip, start point (which corresponds to the place of the start of the trip), end time of the trip, and end point (which refers to the point the trip ends).

Deadhead is each travel of a vehicle, which is not a timetable trip. According to its operational itinerary, this journey can be necessary to place a vehicle in an appropriate local. For example, we have the vehicle's travel to the starting point of a trip or return to the depot at the end of the day. Thus we must provide the deadhead times between the various stopping points of the vehicles. We organized this information in a structure called *deadhead matrix*.

Therefore, given a timetable and a deadhead matrix, the *vehicle scheduling problem* (VSP) consists of determining the operational routine for a vehicle fleet. The aim here is to make the execution of all trips feasible and, at the same time, to minimize the costs involved.

Vehicles itineraries are specified when addressing the VSP. Each itinerary corresponds to the trips assigned to a vehicle that can be carried out successively, without violating the operational rules and time and space limitations. It is noteworthy that, as long as it is feasible, a vehicle can return to the depot more than once during its operating day. Thus, the trips comprised between an exit/return sequence to the depot constitute the so-called *vehicle block*. The cost of the vehicle schedule includes fixed costs for each vehicle's purchase and variable costs for deadhead and waiting time outside the depot.

In the context of public bus transport, a *depot* corresponds to an installation for the management, maintenance, and parking of vehicles when they are not in use (operation). A depot can have an associated maximum capacity and type of fleet. Furthermore, a depot is located at a known distance from each trip's start and end point. Regarding the number of available depots, the vehicle scheduling problem (VSP) can be classified into *single-depot vehicle scheduling problem* (SDVSP) and *multiple-depot vehicle scheduling problem* (MDVSP).

Whenever a vehicle is out of the depot, there must be a *crew* responsible for it. Usually, a crew is composed of a driver and a collector. Therefore, the *crew scheduling problem* (CSP) consists of creating the workdays, so-called *duties*, for the crews. These duties must ensure the feasibly of the vehicle schedule. Besides, we must carry out this distribution of work to minimize labor costs and, at the same time, comply with labor legislation, collective labor agreements, and the operational rules under which the company operates.

Therefore, to address the CSP, the vehicle itineraries from a solution for the VSP are considered. We often observe that a vehicle's operational routine is not adequate to be executed entirely by a single crew (for example, because of its long duration). However, changing the crew responsible for a vehicle cannot be done at any time. It will only occur at so-called *relief points*, that is, at appropriate locations and time intervals.

Whereas in the VSP, the manipulated unit is the trip, in the CSP, it is the *task*. A task represents the smallest amount of work that can be assigned to a crew. It includes consecutive trips of the same vehicle and between which there is no relief point.

In addition to the tasks, many studies propose the formation of the so-called *piece of work* [3], [12], [13], [22]. Each piece of work includes consecutive tasks for the same vehicle, is limited by a minimum and a maximum duration, must be fully assigned to the same crew, and does not include time for rest/feeding of the crew. In this case, a duty consists of combining pieces of work.

To meet legal and operational requirements, companies usually define the types of duties that are valid. Some characteristics that can be considered to specify the types of duties are minimum/maximum working time, minimum

TABLE 1. Summary of the literature review for the MDVCSP.

Reference	Solution approach		Literature instances		Real-world instances	
	Type	Approach	Generated by	Largest instance treated	Companies from	Largest instance treated
Gaffi and Nonato (1999) [21]	Heuristic	CG-LR[1]	-	-	Italy[9]	257 trips, 28 depots
Huisman *et al.* (2005) [22]	Heuristic	CG-LR	themselves	200 trips, 4 depots	Netherlands[10]	653 trips, 1.74 depot/trip[12]
Steinzen *et al.* (2007) [27]	Heuristic	EA-CG-LR[2]	Huisman *et al.* [22]	200 trips, 4 depots	-	-
Borndörfer *et al.* (2008) [23]	Heuristic	CG-LR	Huisman *et al.* [22]	400 trips, 4 depots	Germany[10]	1414 trips, 3 depots
Mesquita and Paias (2008) [24]	Heuristic	CG-LPR-BB[3]	Huisman *et al.* [22][7]	400 trips, 4 depots	-	-
Steinzen *et al.* (2010) [3]	Heuristic	CG-LR	Huisman *et al.* [22][8]	640 trips, 4 depots	-	-
Kliewer *et al.* (2012) [25]	Heuristic	CG-LR	Huisman *et al.* [22][8]	640 trips, 4 depots	-	-
Ciancio *et al.* (2018) [51]	Heuristic	ILS-GH-LS[4]	themselves	712 trips, 4 depots	-	-
Horváth and Kis (2019) [26]	Exact	BP[5]	Huisman *et al.* [22]	100 trips, 4 depots	-	-
Present work	Heuristic	ILS-BB-VND[6]	Huisman *et al.* [22][8]	800 trips, 4 depots	Brazil[11]	934 trips, 3 depots

[1] Column generation in combination with Lagrangian relaxation.
[2] Hybrid evolutionary algorithm and column generation in combination with Lagrangian relaxation.
[3] Column generation in combination with linear programming relaxation; branch-and-bound.
[4] ILS-based heuristic and a greedy heuristic combined with local searches.
[5] Branch-and-price.
[6] Combines branch-and-bound and variable neighborhood descent into an iterated local search-based framework.
[7] Unlike other authors, Mesquita and Paias [24] allowed a given crew to drive vehicles from any depot.
[8] Includes the instances generated by Huisman *et al.* [22] and Steinzen *et al.* [3]. Both used the instances generation algorithm of Huisman *et al.* [22].
[9] A driver must be assigned a single vehicle during the whole duty in this problem. Furthermore, there are different types of vehicles, and for each trip the suggested type is known, corresponding to the most suitable type for that service.
[10] Some trips have to be assigned to vehicles and drivers from a certain subset of depots in this problem.
[11] Every trip can be assigned to any depot in this problem.
[12] The average number of depots to which a trip may be assigned.

break length, maximum time of overtime, and time allowed for the beginning and end of a duty.

Fig. 2 shows the itinerary of a vehicle and its decomposition in a duty. The vehicle makes five trips and returns to the depot once during its itinerary. Thus, it consists of two vehicle blocks. The vehicle's itinerary is decomposed into four tasks as there are only two relief points, station C and DEPOT. From these tasks, we created three feasible pieces of work. We also define a crew's duty, considering that a feasible duty is formed by, at most, two pieces of work and that there must be a break between them. Note that while the vehicle is parked at the depot, the crew takes a break for rest/feeding.

From the above, the MDVCSP approaches the MDVSP and CSP problems simultaneously.

This work proposes a matheuristic capable of satisfactorily addressing real-world and large-scale instances for the MDVCSP. These instances come from the public bus transport system in the city of Belo Horizonte, MG. It is a Brazilian city with the fourth largest fleet of buses in the country.

The constraints considered in solving the problem are:

C1 - Each timetable trip can be serviced by any depot;
C2 - Each trip has exactly two relief points: one at the beginning and one at the end of the trip. That is, each trip of the vehicle is considered a task;
C3 - Each vehicle is assigned to a single depot and must start and end your itinerary there;
C4 - Each crew is associated with a depot and can only drive vehicles from it. Every duty does not need to start and end in the depot. However, when they take place outside the depot, these activities require additional time for the crew to prepare;
C5 - The viability of a piece of work depends only on its duration, which is limited by a minimum and a maximum time;
C6 - During its itinerary, the vehicle will return to the depot whenever the idle time between two consecutive trips is sufficient to perform a round trip to the depot;
C7 - A crew can change vehicles (*changeover*), but only during a rest/feeding break (that is, between pieces of work);
C8 - The so-called *continuous attendance* is respected. That is, always there is a crew responsible for a vehicle that is outside the depot, regardless of whether it is stationary or moving;
C9 - Each depot is unlimited concerning the number of associated crews;
C10 - Each depot has a limited number of assigned vehicles;
C11 - If the rest/feeding interval of the crew occurs between work shifts, the crew must end the first shift at the start point of the second shift;
C12 - Each vehicle must remain in the depot for a minimum of time at the end of its daily operating itinerary. This time is for cleaning and maintaining the vehicle.

IV. MODELING APPROACH

A. A LITERATURE MODELING APPROACH FOR THE MDVCSP

This section presents the mathematical formulation proposed by Steinzen *et al.* [3] for the MDVCSP. It combines a multi-commodity network flow formulation for vehicle scheduling with a set partitioning formulation for crew scheduling.

To model the problem, Steinzen *et al.* [3] consider the same constraints presented in Section III, except constraints C11 and C12. Regarding constraint C11, they consider that there may be a change of point in a rest/feeding interval, but there must be enough time for the crew to comply with the rest time and walk between these locations. About constraint C12, there is no limitation on a vehicle's itinerary duration.

1) TIME-SPACE NETWORK FOR THE MDVSP

In a time-space network for the MDVSP each node represents a *location* in a given *instant*; each arc corresponds to a transition in time and, possibly, in space. For each vehicle stopping point (station or depot), we define an imaginary timeline with nodes representing the departure and arrival events at this point. In the timeline, the nodes are increasingly ordered by the instant they represent.

We associate a network layer with each depot of the problem. Fig. 3 illustrates a network layer with three stations (A, B, and C), five trips, and the respective depot. In this example, the planning horizon is from 6:00 am to 12:00 pm. The types of arcs present in the network are:

1) *trip arc*: arc associated with a trip. It connects the departure node (station and start time of the trip) to the arrival node (station and end time of the trip);
2) *pull-out/pull-in arcs*: a *pull-out arc* connects a node in the depot to the starting node of a trip and represents the deadhead from the depot to the start station of the trip. A *pull-in arc* connects the arrival node of a trip to a node in the depot and represents the deadhead from the end station of the trip to the depot;
3) *waiting arc*: connects consecutive nodes on the point's timeline and corresponds to waiting at the point. We eliminate *waiting arcs* from the network that represents long durations in the stations. Note that in Fig. 3, for example, there is no *waiting arc* in station C connecting the trips t_4 and t_5. This simplification is made possible by the constraint C6 of Section III, initially proposed by Huisman *et al.* [22] to address a problem in the literature;
4) *deadhead arc*: connects the arrival node of a trip to a departure node in another station where there are trips compatible with that trip. Therefore, this arc represents a deadhead between compatible trips. In Fig. 3, to execute trips t_2 and t_3 consecutively, a vehicle needs to make a deadhead from station A to station B. On the other hand, to execute trip t_4 just after t_2, the vehicle needs to make the same previous deadhead and must remain to wait in station B;

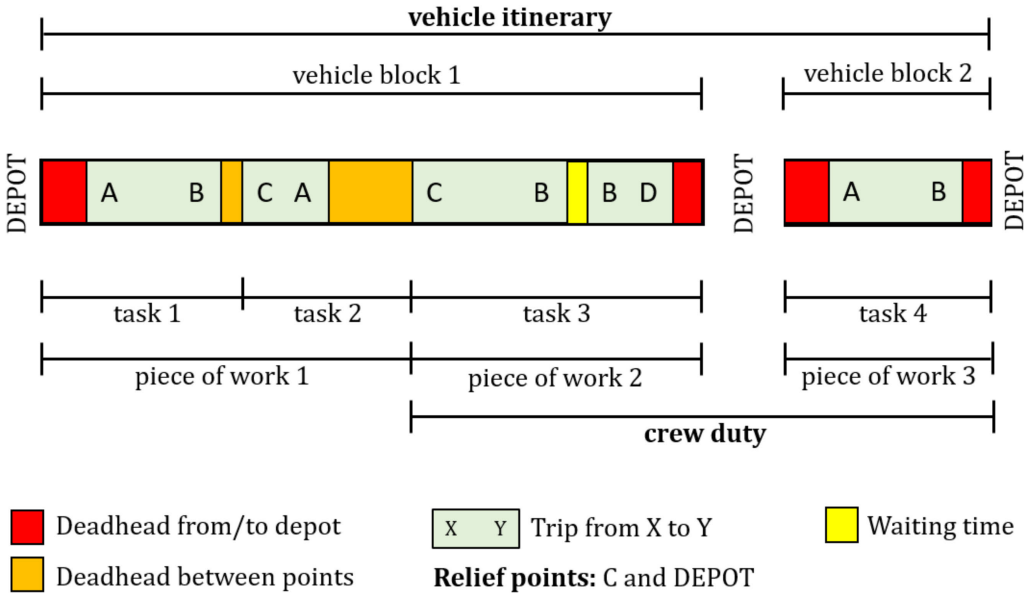


FIGURE 2. Example of vehicle itinerary and its crew duty.

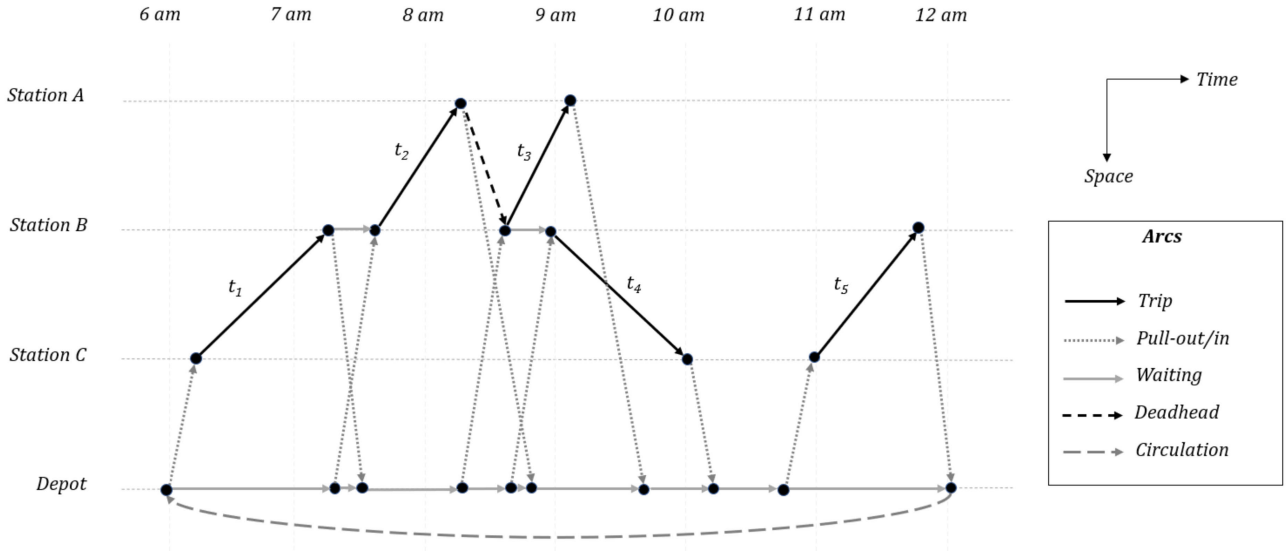


FIGURE 3. Time-space network layer for the MDVSP.

5) *circulation arc*: connects the last node (*sink*) to the first node (*source*) in the depot timeline. This arc allows flow circulation in the network. Each flow unit in the *circulation arc* corresponds to a vehicle used in the schedule.

To the best of our knowledge, the formulations proposed in the literature for the MDVCSP represent the associated MDVSP in two ways: 1– *connection-based network*, initially proposed by Huisman *et al.* [22]; or 2– *time-space network* [3], as presented above.

In a connection-based network, we have an arc between each pair of compatible trips. On the other hand, many of these connections are considered only implicitly in the time-space network. In Fig. 3, for example, the trips t_2 and t_4 are compatible since it is possible to traverse different types of arcs from t_2 and arrive at t_4.

As shown in [16] and [3], in the time-space network the amount of *deadhead arcs* is much less than in the connection-based network. This fact has a substantial impact on the number of variables and constraints of the model for the MDVSP. So, we choose to use the time-space network to formulate the vehicle scheduling in this work.

For a detailed description about building time-space networks, see Kliewer *et al.* [16] and Steinzen [52].

2) MATHEMATICAL FORMULATIONS FOR THE MDVSP AND MDVCSP

Let $T = \{1, 2, \ldots, m\}$, the set of all m timetable trips and $D = \{1, 2, \ldots, n\}$, the set of n depots. For each depot $d \in D$ there is an acyclic time-space network layer $G^d = (N^d, A^d)$, where N^d is the set of nodes and A^d is the set

of arcs. We denote $\tilde{A}^d \subset A^d$, the set of arcs that represent activities that involve the joint participation of vehicle and crew. So, $\tilde{A}^d$ does not include the *waiting arcs* in the depots, according to the *continuous attendance* requirement (Section III, constraint C8). Let $A^d(t) : T \rightarrow A^d$ the function that returns a set with precisely one *trip arc*, $(i, j) \in A^d$, associated with the trip $t \in T$, if t can be assigned to the depot $d \in D$. Otherwise, we have an empty set. The maximum capacity u_{ij}^d of each arc $(i, j) \in A^d, \forall d \in D$, is defined as follows: *pull-out/in* and *trip arcs* have a maximum capacity of 1; all other arcs have a maximum capacity equal to the number of vehicles available in the depot $d \in D$. Each arc $(i, j) \in A^d$ has an associated cost c_{ij}^d. In *circulation arcs*, c_{ij}^d represents the cost of purchasing a vehicle. In the other arcs, c_{ij}^d is the operating cost. Let K^d be the set of duties associated with the depot $d \in D$. $K^d(i, j) \subset K^d$ is the set of duties associated with the depot d that covers the $(i, j) \in \tilde{A}^d$ arc. f_k^d refers to the cost of the duty $k \in K^d$ and involves fixed and working time crew costs.

Be the following types of decision variables:

- y_{ij}^d: indicates the flow associated with the arc $(i, j) \in A^d$, $d \in D$ (flow variable);
- x_k^d: determines whether the duty $k \in K^d$, $d \in D$, is part of the schedule (binary variable).

The model proposed by Steinzen *et al.* [3] for the MDVCSP is presented below.

$$\min \sum_{d \in D} \sum_{(i,j) \in A^d} y_{ij}^d c_{ij}^d + \sum_{d \in D} \sum_{k \in K^d} x_k^d f_k^d, \tag{1}$$

$$\text{s. t.} \sum_{d \in D} \sum_{(i,j) \in A^d(t)} y_{ij}^d = 1 \quad \forall t \in T \tag{2}$$

$$\sum_{\{j:(j,i) \in A^d\}} y_{ji}^d - \sum_{\{j:(i,j) \in A^d\}} y_{ij}^d = 0 \quad \forall d \in D, \ \forall i \in N^d \tag{3}$$

$$\sum_{k \in K^d(i,j)} x_k^d - y_{ij}^d = 0 \quad \forall d \in D, \ \forall (i, j) \in \tilde{A}^d \tag{4}$$

$$0 \leq y_{ij}^d \leq u_{ij}^d, \ y_{ij}^d \in \mathbb{N} \quad \forall d \in D, \ \forall (i, j) \in A^d \tag{5}$$

$$x_k^d \in \{0, 1\} \quad \forall d \in D, \ \forall k \in K^d. \tag{6}$$

The objective is to minimize the cost of vehicle and crew schedules (1). Constraints (2) ensure that each timetable trip will be assigned to precisely one vehicle from one of the depots that can serve it. In constraints (3), flow conservation at the nodes is ensured for each network layer (depot). Constraints (4) guarantee that, for each depot, there will be the same number of vehicles and crews covering each activity (arc) that requires the association of a vehicle and a crew. Constraints (5) ensure that the maximum capacities of the flow variables are respected.

In the formulation for MDVCSP (1)–(6), considering only the portion of the objective function associated with vehicle schedule ($\sum_{d \in D} \sum_{(i,j) \in A^d} y_{ij}^d c_{ij}^d$) and the constraints (2), (3), and (5), we get a formulation for the MDVSP. This formulation was proposed in [16].

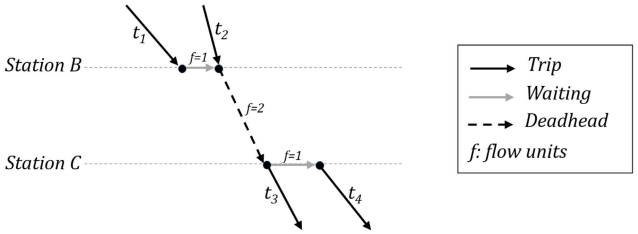


FIGURE 4. Flow decomposition possibilities.

In this formulation, an optimal solution (flow) represents a set of optimum vehicle schedules due to the aggregation of connections in the time-space network specification. Fig. 4 illustrates this situation in a small portion of the network and its optimal flow. There are two different ways to sequence trips t_1, t_2, t_3, and t_4 on the itineraries of two vehicles, they are: $t_1 - t_3$ (vehicle 1) and $t_2 - t_4$ (vehicle 2) or $t_1 - t_4$ (vehicle 1) and $t_2 - t_3$ (vehicle 2). Thus, we use a flow decomposition procedure to obtain a specific vehicle schedule. There are different strategies for defining paths in the time-space network with the optimal flow. Each path from the source node to the sink node in the depot timeline represents a vehicle itinerary.

B. PROPOSED MODELING APPROACH FOR THE MDVSP AND MDVCSP

In this section, we discuss our network structure and mathematical formulation for the MDVSP that considers the particularities of the Brazilian real-world problem addressed in the present work. That is, this modeling approach meets the assumptions and constraints stated in Section III. As we will describe later, our MDVCSP resolution heuristic uses this model to generate an initial vehicle schedule.

1) PROPOSED TIME-SPACE NETWORK FOR MDVSP

The formulation for the MDVSP presented in Section IV-A2 is consistent and efficient for solving instances widely used in the literature, as the ones proposed by Huisman *et al.* [22] and Steinzen *et al.* [3]. However, it is not adequate to represent the Brazilian real-world problem that we have addressed in this work.

In the instances from literature, trips always start from 6 am and end before 1 am. Thus, in the 24 hours period, there is an interval of more than 5 hours in which no trip is performed. However, in the Brazilian real-world instances that we consider, there are timetable trips distributed throughout the day. There is even an extrapolation of the 24-hour operating period. In this sense, some trips start at the end of a day and end in the early hours of the next day; other trips start in the day's first moments. Besides, the public transport management company requires that every vehicle remains in the depot for a minimum time at the end of its daily itinerary. This time is used for cleaning and overhauling the vehicle.

Fig. 5 illustrates the timeline of a depot for a Brazilian real-world instance. Shaded nodes are associated with

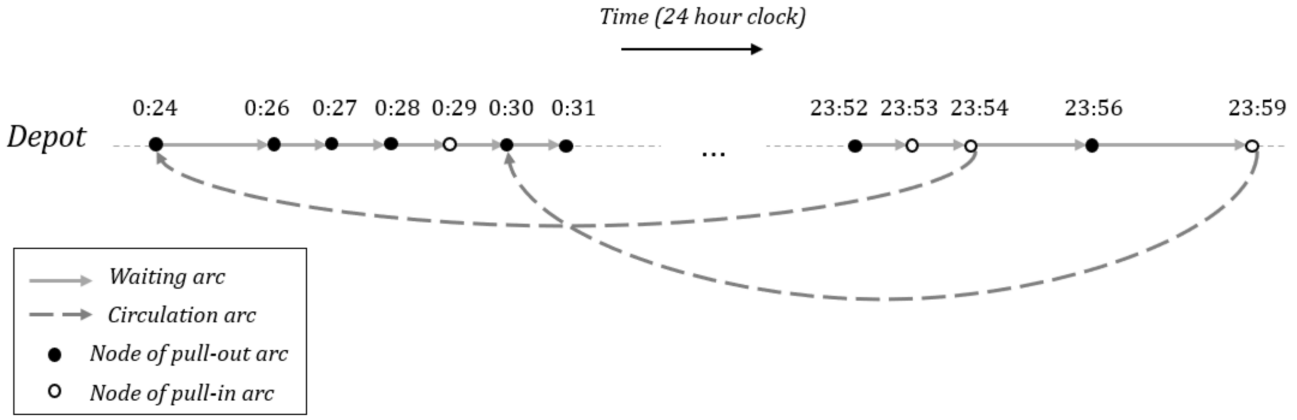


FIGURE 5. Circulation arcs of the timeline of a depot in a Brazilian real-world problem.

pull-out arcs (depot exit) and non-filled nodes with *pull-in* arcs (depot arrival). This example requires that the itinerary of any vehicle lasts a maximum of 23 hours and 30 minutes.

Suppose a single circulation arc in Fig. 5 connects the last node in the depot's timeline to the first one. In that case, we could define itineraries that are longer than acceptable. So we created more than one circulation arc. Each arc will represent a valid operating period (shift) for the vehicles in this depot.

By definition and without loss of generality, we can consider that a circulation arc must always connect a node of an arc *pull-in* (arrival at the depot) to a node of an arc *pull-out* (exit of the depot). Thus, below we detail how the circulation arcs are created. We go through the depot timeline iteratively (node by node) and from right to left, as follows: at each node of a *pull-in* arc reached (origin of the circulation arc), we search for the node of a *pull-out* arc (destination of the circulation arc). We chose this second node in such a way as to define the longest feasible operating period possible. When selecting the destination node of the circulation arc, two situations can occur:

1) There is already a circulation arc arriving at this node. So, we did not create a new circulation arc;
2) There is no circulation arc with a destination at this node. So, we created a new circulation arc. If the target node of this arc is the first node in the depot timeline, we finish the arc definition process.

At the end of this procedure, all nodes in the depot will be covered, i.e., they will belong to one or more operating periods.

In our representation of the MDVSP, we must maintain a circulation arc per network layer. Therefore, each layer will be associated with the depot-operating period feasible combination; and it will only represent trips that can be carried out in the depot and period considered. In Fig. 6, we break down the network layer of a depot with two circulation arcs in Fig. 6a (inconsistent representation) into two network layers in Figs. 6b and 6c. Therefore, in general, the timeline of a depot with w circulation arcs will give rise to w layers of the time-space network.

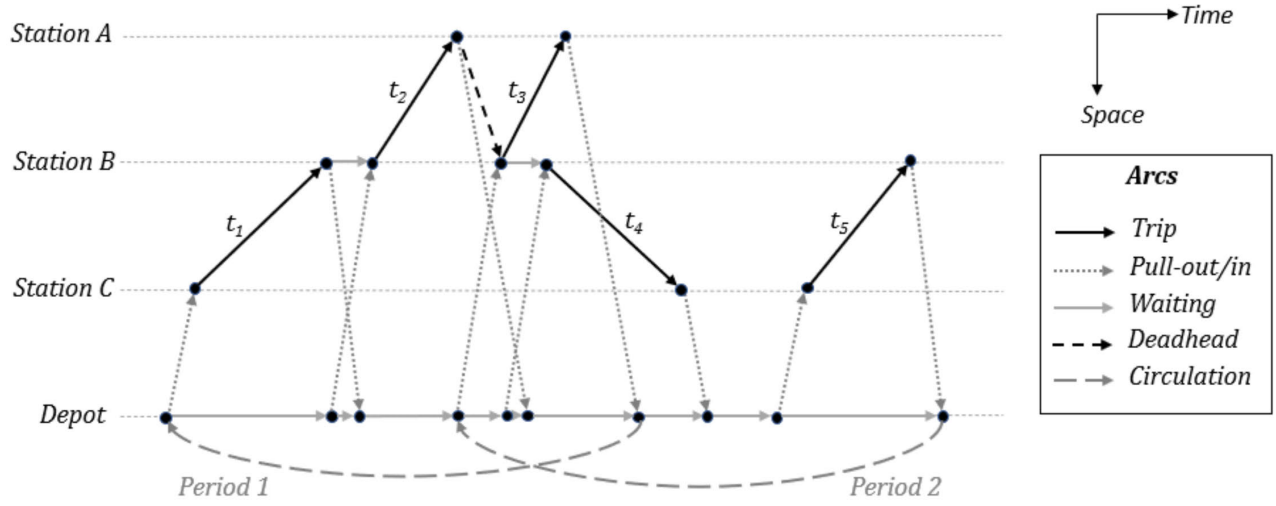
In the following, we discuss the complexity of the proposed network structure. Let m be the number of timetable trips, p the number of different stations, and ℓ the number of network layers (depot-operating period combinations). Thus, the number of *deadhead arcs* in a network layer is $O(mp)$ because one *deadhead arc* connects a trip with all subsequent trips at a different station. On the other hand, the number of *waiting arcs*, *pull-out/pull-in arcs*, or *trip arcs* grows linearly with the number of trips in a network layer. Finally, there is one circulation arc in a network layer. Therefore, the number of arcs in the time-space network is $O(\ell mp)$. Note that this number of arcs determines the number of variables of the proposed mathematical formulation for the MDVSP, as shown in the next section.

2) PROPOSED MATHEMATICAL FORMULATIONS FOR THE MDVSP AND MDVCSP

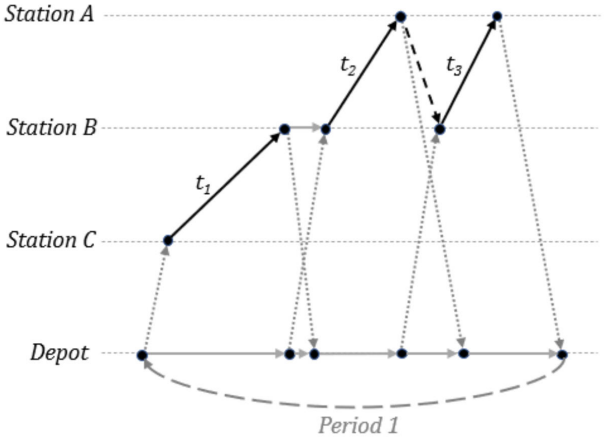
The formulation we propose for the MDVSP is presented below (7)–(11).

Let $T = \{1, 2, \ldots, m\}$ be the set of all m timetable trips and $D = \{1, 2, \ldots, n\}$ the set of n depots. From the depots in D we define $\Delta = \{1, 2, \ldots, \ell\}$, the set of all possible depot-operating period combinations. Each feasible operating period is defined by a circulation arc. For each depot-period $\delta \in \Delta$ we have an acyclic time-space network layer $G^{\delta} = (N^{\delta}, A^{\delta})$, where: N^{δ} is the set of nodes and A^{δ} the set of arcs. Let $A^{\delta}(t) : T \rightarrow A^{\delta}$ be the function that returns a set with exactly one *trip arc*, $(i, j) \in A^{\delta}$, associated with the trip $t \in T$, if t can be covered by the depot and operating period given by $\delta \in \Delta$. Otherwise, we have an empty set.

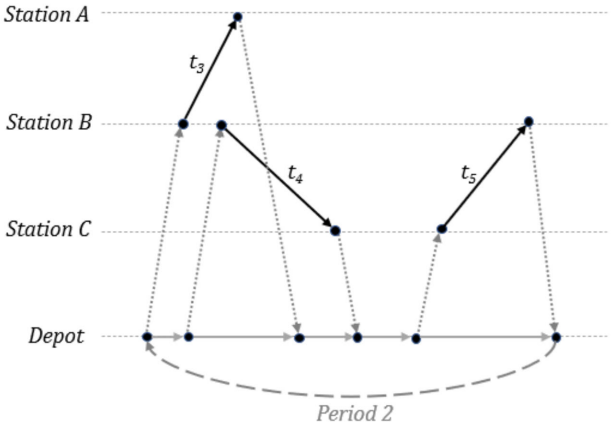
Let Δ^{d} be the set of depot-period combinations in which the depot involved is $d \in D$. F^{δ} is the set formed by the circulation arc associated with the depot-period $\delta \in \Delta$. cap^{d} refers to the capacity of the depot $d \in D$ concerning the number of vehicles.



(a) Network layer of a depot with two circulation arcs.



(b) Network layer with only the circulation arc of period 1.



(c) Network layer with only the circulation arc of period 2.

FIGURE 6. Definition of one network layer for each depot-period.

Each arc $(i, j) \in A^\delta$ has an associated cost c_{ij}^δ. In *circulation arcs*, c_{ij}^δ represents the cost of purchasing a vehicle. In the other arcs, c_{ij}^δ is the operating cost. Finally, we have the decision variable y_{ij}^δ which indicates the flow associated with the arc $(i, j) \in A^\delta$, $\delta \in \Delta$.

$$\min \sum_{\delta \in \Delta} \sum_{(i,j) \in A^\delta} y_{ij}^\delta c_{ij}^\delta, \tag{7}$$

$$\text{s. t.} \sum_{\delta \in \Delta} \sum_{(i,j) \in A^\delta(t)} y_{ij}^\delta = 1 \quad \forall t \in T \tag{8}$$

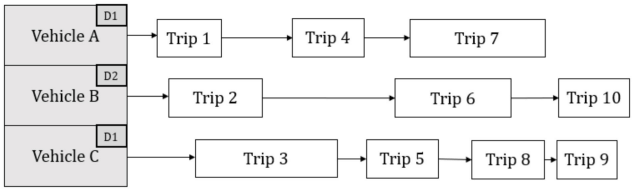
$$\sum_{\{j:(j,i) \in A^\delta\}} y_{ji}^\delta - \sum_{\{j:(i,j) \in A^\delta\}} y_{ij}^\delta = 0 \quad \forall \delta \in \Delta,\ \forall i \in N^\delta \tag{9}$$

$$\sum_{\delta \in \Delta^d} \sum_{(i,j) \in F^\delta} y_{ij}^\delta \leq cap^d \quad \forall d \in D \tag{10}$$

$$y_{ij}^\delta \in \mathbb{Z}^+ \quad \forall \delta \in \Delta,\ \forall (i, j) \in A^\delta. \tag{11}$$

The objective is to minimize the cost of vehicle schedule (7). The constraints (8) ensure that each timetable trip will be assigned to precisely one vehicle. This vehicle must be linked to a depot-operating period combination that is compatible with the trip. In (9) we have the flow conservation constraints at the nodes of each network layer (depot-period). The constraints (10) ensure that the capacity of each depot, in relation to the number of vehicles, is respected. The constraints (8)–(11) implicitly establish an upper limit for the value of each flow variable y_{ij}^δ. Let u_{ij}^δ be the upper limit for $(i, j) \in A^\delta$ and $\delta \in \Delta$, we have: $u_{ij}^\delta = 1$, for *trip* and *pull-in/out arcs*. In addition, u_{ij}^δ depends on the depot capacity associated with δ. That is, the sum of the flows of the *circulation arcs* of a depot cannot exceed the capacity of the depot; and the flow in any arc in a network layer cannot be greater than the flow of your *circulation arc*.

This formulation can be combined with the set partitioning formulation for the CSP of Section IV-A2. Thus, we obtain a formulation for the MDVCSP.



Two depots: D1 and D2

FIGURE 7. Example of a solution s^v for the MDVSP.

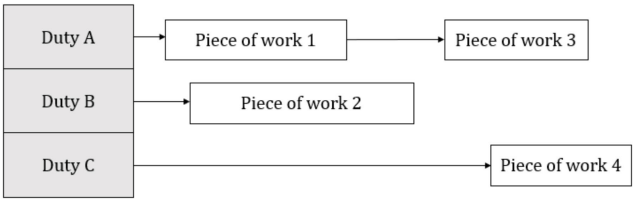


FIGURE 8. Example of a solution s^c for the CSP.

V. SOLUTION APPROACH

Given the difficulty of solving real-world instances of the MDVCSP using exact mathematical methods, we propose a matheuristic algorithm for solving it. This matheuristic algorithm combines two strategies into an ILS-based framework: a branch-and-bound algorithm for solving the MDVSP and a VND-based algorithm for treating the associated CSPs.

We generate an initial solution for the MDVCSP addressing its subproblems, MDVSP and CSP, sequentially. Initially, we solve the MDVSP in optimality. Then, from its solution with n depots ($n \geq 2$), we generate n independent CSPs, that is, one CSP per depot. Each CSP is solved separately by a VND-based heuristic method. Note that it does not make sense to solve a single CSP considering all depots simultaneously because, according to constraints C4 of Section III, each crew can only drive vehicles from the same depot.

Next, we describe how to represent the MDVSP, CSP, and MDVCSP in Section V-A. In Section V-B, we present the neighborhood structures used to explore the solution space of these problems. In Section V-C, we define the evaluation functions to vehicle and crew schedules. Section V-D presents the proposed matheuristic algorithm for solving the MDVCSP. Section V-E describes heuristic algorithms to treat the CSP exclusively, which we use as auxiliary methods to solve the MDVCSP in Section V-D.

A. SOLUTION REPRESENTATION

1) SOLUTION REPRESENTATION FOR THE MDVSP

A solution s^v for the MDVSP consists of a list of vehicles. In turn, for each vehicle, we associate its depot and the list of trips that it will perform during a working day. Fig. 7 illustrates a solution s^v for the MDVSP in which a fleet of three vehicles, distributed over two depots, must perform ten trips.

In this representation, we organize the trips made by each vehicle according to their start times. Thus, it is possible to evaluate the vehicle's operational itinerary in several aspects, such as the locomotion time out of operation (deadhead), the waiting time at the station between two consecutive trips, the number of returns to the depot, and the viability of the vehicle block.

2) SOLUTION REPRESENTATION FOR THE CSP

A solution s^c for the CSP consists of a list of duties. Each duty associates the pieces of work to be performed by the same crew during a working day. Fig. 8 illustrates a solution s^c for the CSP. In this example, there is a depot with three duties and four pieces of work assigned.

We keep the pieces of work for each day on a list and sort them in ascending order by their start times. In this way, it is possible to determine the day's relevant characteristics, such as time reserved for rest/food, the occurrence of vehicle change, and the existence of an overlap between tasks.

3) SOLUTION REPRESENTATION FOR THE MDVCSP

We represent a solution S^{vc} for the MDVCSP by a pair $S^{vc} = [s^v, S^c]$, where s^v represents the solution to the MDVSP and S^c represents the solution set of the n CSPs associated to each solution s^v. Both representations were previously described and illustrated in Sections V-A1 and V-A2, respectively.

Let s^v be a solution for the MDVSP, s_i^c a solution for the CSP associated with the i-th depot, and n the number of depots. Then, we define S^c as a set of n solutions, one for each of the n CSPs obtained from s^v, that is:

$$S^c = \left\{s_1^c, s_2^c, \ldots, s_n^c\right\}. \tag{12}$$

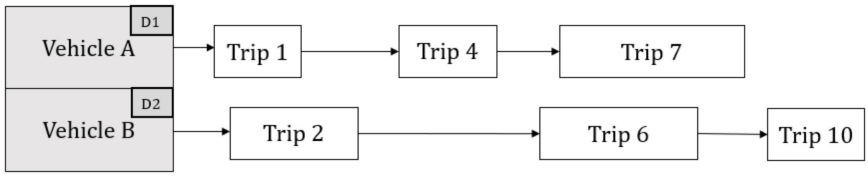
As some depots may be inactive (i.e., without linked vehicles) in a solution for the MDVSP, some solutions s_i^c into the set S^c may be empty (i.e., without duties).

B. NEIGHBORHOOD STRUCTURES

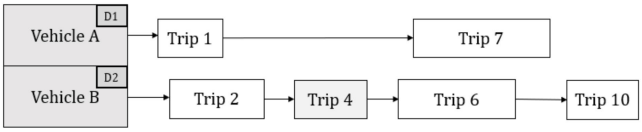
1) MDVSP NEIGHBORHOOD STRUCTURES

We apply the six types of moves described below to explore the MDVSP solution space. Each move defines a neighborhood structure. Fig. 9 illustrates all of them.

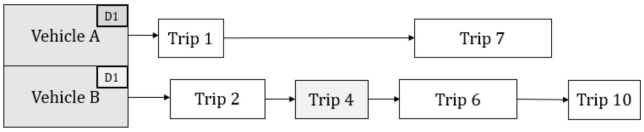
1) *Trip relocate without depot adjustment* (Neighborhood structure N_r^v): It consists of reallocating a trip without allowing changing depots. Fig. 9b illustrates this move. A trip is transferred from one vehicle to another one, and the depots of the vehicles involved are maintained;
2) *Trip relocate with depot adjustment* (Neighborhood structure N_{rd}^v): It consists of reallocating a trip allowing changing depots. In Fig. 9c, a trip is transferred from one vehicle to another one. This transfer seeks to assign each modified vehicle to the depot that implies the lowest operating cost considering the new itinerary configuration;
3) *Trip exchange without depot adjustment* (Neighborhood structure N_e^v): This move consists of exchanging trips between two vehicles, not allowing changing their depots. Fig. 9d shows how it works. A vehicle exchanges



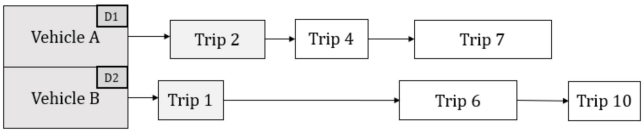
(a) Initial MDVSP solution.



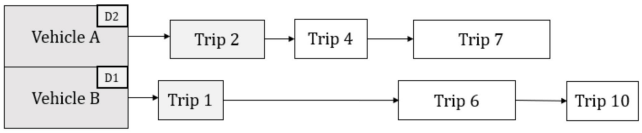
(b) Move 1 - Relocates the Trip 4 of Vehicle A to Vehicle B.



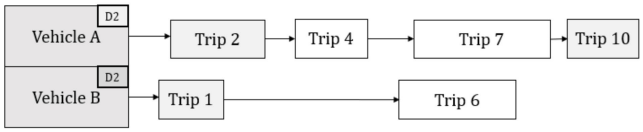
(c) Move 2 - Equal Fig. 9b and makes depot adjustments.



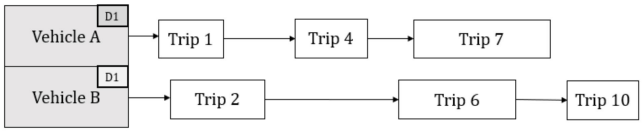
(d) Move 3 - Exchange of Trip 1 of Vehicle A with Trip 2 of Vehicle B.



(e) Move 4 - Equal Fig. 9d and makes depot adjustments.



(f) Move 5 - Redistributes the trips of the A and B vehicles and makes depot adjustments.



(g) Move 6 - Vehicle B depot change from D2 to D1.

FIGURE 9. **MDVSP moves.**

one of its trips with a trip of another vehicle, and the depots of the vehicles involved are maintained;

4) *Trip exchange with depot adjustment* (Neighborhood structure N_{ed}^{v}): This move consists of exchanging trips between two vehicles allowing them to change their depots. Fig. 9e illustrates its operation. A vehicle exchanges one of its trips with a trip from another vehicle. In this exchange, we seek to assign each modified vehicle to the depot that implies the lowest operating cost considering the new itinerary configuration;
5) *Trip redistribution* (Neighborhood structure N_{dd}^{v}): This move consists of redistributing the trips of two vehicles allowing them to change their depots. Fig. 9f) shows the application of this move. Two vehicles have all their trips removed. These trips are then randomly redistributed to each other. Finally, we seek to assign each vehicle to the depot that implies the lowest operating cost considering the new itinerary configuration;
6) *Depot change* (Neighborhood structure N_{dc}^{v}): It consists of changing the depot linked to a vehicle. Fig. 9g shows a neighbor in which vehicle B changes from depot D2 to D1.

We emphasize that in the relocation, exchange, and redistribution moves, the vehicles involved should not necessarily belong to the same depot. Besides, we apply only those moves that maintain the feasibility of the solution.

2) CSP NEIGHBORHOOD STRUCTURES

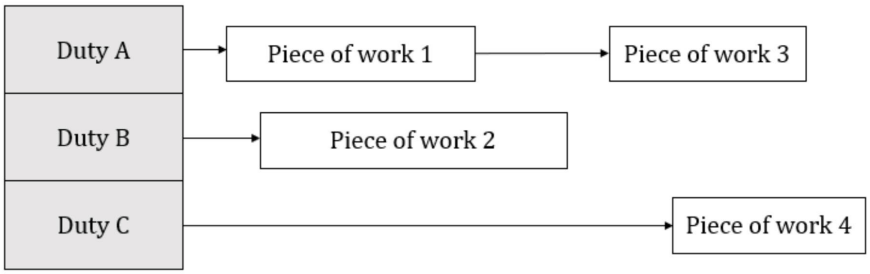
We use two types of moves to explore the CSP solution space: relocate a piece of work and exchange pieces of work. These moves are illustrated in Fig. 10 and are described below, together with the associated neighborhood structures.

1) *Relocate piece of work* (Neighborhood Structure N_{r}^{c}): It consists of reallocating one piece of work from one duty to another duty. Fig. 10b illustrates this move. A piece of work is transferred from one duty to another;
2) *Exchange pieces of work* (Neighborhood Structure N_{e}^{c}): This move consists of exchanging pieces of work between two duties. Fig. 10c shows how it works. One piece of work belonging to a duty is exchanged with a piece of work from another duty.

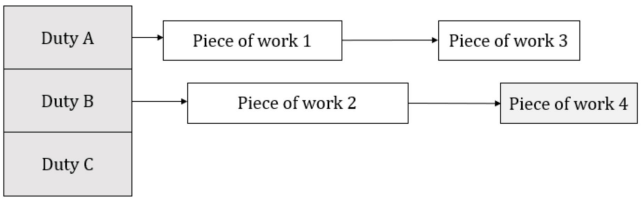
In these moves, the manipulated duties belong to the same depot since there is an independent CSP per depot. Besides, we only carry out movements that maintain the viability of the s^c solution.

C. EVALUATING FUNCTION

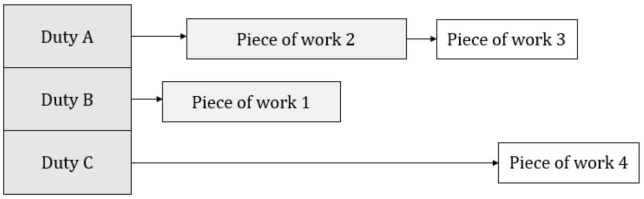
To determine the quality of a solution $S^{vc} = [s^v, S^c]$ for the MDVCSP, we associate a cost with vehicle and crew schedules.



(a) Initial CSP solution.



(b) Move 1 - Relocates the Piece of work 4 from Duty C to Duty B.



(c) Move 2 - Exchange of Piece of work 1 of Duty A with Piece of work 2 of Duty B.

FIGURE 10. CSP moves.

We evaluate a solution s^v for the MDVSP based on the following function f^v (13), which should be minimized:

$$f^v(s^v) = totalVehicles \times vehicleCost + operationTime \times operationalCost, \quad (13)$$

where:

(a) *totalVehicles* is the number of vehicles used on the vehicle schedule;
(b) *vehicleCost* is the vehicle cost;
(c) *operationTime* is the total time, in minutes, that the vehicles in the fleet were out of the depot (regardless of the activities carried out during this period, which may include: trip, deadhead, or waiting at the station);
(d) *operationalCost* is the cost per minute due to a vehicle staying outside its depot.

In turn, we calculate the cost of a solution s^c for the CSP using the function f^c, as shown below in (14):

$$f^c(s^c) = totalCrews \times crewCost + totalWorkingTime \times workingTimeCost, \quad (14)$$

where:

(a) *totalCrews* is the number of crews on the schedule;
(b) *crewCost* is the cost of each crew;
(c) *totalWorkingTime* is the total working time, in minutes, of all crews on the schedule. That is, we consider the total duration of each duty, including the mandatory rest time;
(d) *workingTimeCost* is the cost of each minute of work for one crew.

It is worth noting that $S^{vc} = [s^v, S^c]$ and $S^c = \{s_1^c, s_2^c, \ldots, s_n^c\}$ for n depots. In this sense, the cost of a solution S^{vc} for MDVCSP is evaluated according to (15):

$$f^{vc}(S^{vc}) = f^v(s^v) + \sum_{i=1}^{n} f^c(s_i^c), \quad (15)$$

where $f^v(s^v)$ and $f^c(s_i^c)$ are the functions that evaluate the solutions s^v and S^c for the MDVSP and CSP, respectively. There is no cost for infeasibility since the solution S^{vc} is kept feasible throughout the solution methods.

D. MATHEURISTIC ALGORITHM FOR THE MDVCSP

The matheuristic algorithm ILS-MDVCSP proposed for solving the MDVCSP combines an exact method with a heuristic method into an ILS-based framework [33]. It solves the MDVSP optimally using a branch-and-bound algorithm. A constructive procedure and a VND-based local search procedure [34] generate the solutions of the associated CSPs. Finally, the ILS-MDVCSP iteratively modifies the integrated solution through perturbations and performs a VND-based local search on the CSPs that have been changed.

The ILS-MDVCSP is described in Algorithm 1. Initially, we generate a solution s^v for the MDVSP (line 8). For that, an integer linear programming (ILP) optimization solver solves the problem from one of the formulations presented in the Sections IV-A2 and IV-B2. The choice of the formulation depends on the MDVSP specifically addressed, that is, the MDVSP as defined in the literature or the Brazilian real-world problem addressed in this work.

As each depot considered in the MDVSP generates a crew scheduling problem, in line 9 we define the set of solutions S^c. For a problem with n depots, S^c contains n crew schedules. This procedure is detailed in Algorithm 2.

In Algorithm 2, the itinerary of each vehicle of the solution s^v is partitioned into pieces of work to form the set P

(line 4). Subsection V-E1 describes the two partitioning methods used: direct and inverse. For each vehicle in the schedule, we randomly choose one of these procedures. Then, we separate the pieces of work obtained by the depot (line 5). Thus, we generate n sets of pieces of work. When no vehicles are assigned to some available depots, some of them may be empty. In this way, there will be n CSPs to solve at most. At each iteration (lines from 8 to 12), a solution s_i^c for the CSP associated with the i-th depot is obtained and added to the solution set S^c. For that, we consecutively invoke the constructive procedure described in Algorithm 7 (Subsection V-E3) and the local search procedure based on the basic variable neighborhood descent (B-VND) [34] described in Algorithm 8 (Subsection V-E4).

We form a complete solution for the MDVCSP (S_*^{vc}) in line 10 of Algorithm 1. Then, we execute the procedures described in lines 12 to 14 as long as the maximum processing time ($time_{max}$) has not been reached.

Algorithm 3 describes the *perturbation* method. In this method, a perturbation of a certain level begins with applying random movements in the solution s^v for the MDVSP (line 4).

We applied six levels of perturbations based on the moves defined in Subsection V-B1, namely:

1) *level 1*: one depot change move,
2) *level 2*: one trip relocation move,
3) *level 3*: one trip exchange move,
4) *level 4*: one trip redistribution move,
5) *level 5*: two trips exchange moves,
6) *level 6*: two trips redistribution moves.

As the perturbation level increases, the search procedure gradually moves away from the current solution towards other regions that have not yet been explored in the problem's solution space.

When carrying out the relocate and exchange moves of trips, we randomly chose between two possibilities: 1) maintaining the depots of the vehicles involved (neighborhood structures N_r^v and N_e^v) and 2) inserting the modified vehicles in the depots that generate lower cost (neighborhood structures N_{rd}^v and N_{ed}^v).

Any change in the MDVSP solution (s^v) causes changes to the CSP solutions (S^c). So, for vehicle and crew schedules to be kept individually feasible and mutually compatible, the pieces of work of the vehicles that have been modified must be rebuilt. For this, we randomly choose one of the partitioning methods defined in Subsection V-E1 (direct or inverse) for each vehicle. In this process, we must remove the pieces of work that no longer exist from the crews to which we assign them. We also need to assign each new piece of work to a crew of the schedule. Thus, the pieces of work removed and created are separated by depot (lines 5 and 6, respectively). Pieces of work are excluded from the CSP solution directly. To do it, we have only to find the crews responsible for these pieces (line 10). In lines 13 to 15, each new piece of work is inserted into the schedule using the heuristic procedure described in Algorithm 6 (Subsection V-E2). Due to the construction process, in set C_i, each vehicle's pieces of work are arranged sequentially. However, we can also sort this set by the start time of the pieces of work. Then, the parameter *sortPieces_pert* defines the order of the pieces of work in the set C_i (line 11).

From this perturbed solution for the MDVCSP, we apply the *LocalSearch* method (line 13 from ILS-MDVCSP algorithm). According to Algorithm 4, we apply a VND-based local search (Algorithm 8, Subsection V-E4) for solving each CSP that has been modified. In this step, the vehicle schedule is not changed.

Finally, we evaluate the new solution obtained at line 14 of the ILS-MDVCSP algorithm. According to Algorithm 5, we accept the new solution S^{vc} (vehicle and crew schedules) only if it has a cost less than or equal to the cost of the current solution S_*^{vc}. In this case, we restart the perturbation level to intensify the search in this current region. Otherwise, we increase the perturbation level by one unit to move away from the current region or restart it if it is already at the highest perturbation level.

Algorithm 1: ILS-MDVCSP

1 **Data:** T, the set of the timetable trips.
2 **Data:** n, the number of depots of the problem.
3 **Data:** l_{max}, the maximum level of perturbation.
4 **Data:** $time_{max}$, the maximum processing time.
5 **Result:** S_*^{vc}, the best solution found for the MDVCSP.
6 $time \leftarrow 0$ // the current processing time of the algorithm
7 $l \leftarrow 1$ // the current level of perturbation
8 $s^v \leftarrow solver(T, n)$
9 $S^c \leftarrow initialSolutions\_CSPs(s^v, n)$ // See Algorithm 2
10 $S_*^{vc} \leftarrow [s^v, S^c]$ // The complete solution S_*^{vc} for the MDVCSP consists of the solution s^v for the MDVSP and the set S^c of solutions for the n CSPs
11 **while** $time < time_{max}$ **do**
12 $S_1^{vc} \leftarrow perturbation(S_*^{vc}, l)$ // See Algorithm 3
13 $S_2^{vc} \leftarrow localSearch(S_1^{vc})$ // See Algorithm 4
14 $[S_*^{vc}, l] \leftarrow acceptanceCriterion(S_*^{vc}, S_2^{vc}, l, l_{max})$ // See Algorithm 5

E. HEURISTIC METHODS FOR THE CSP

In this section, we present the heuristic procedures developed to address the CSP. These procedures consider that the vehicle schedule of an MDVSP depot is known. We apply them to solve the MDVCSP described in Section V-D.

Algorithm 2: *initialSolutions_CSPs*

Data: s^v, a solution for the MDVSP.
Data: n, the number of depots of the problem.
Result: S^c, the set of solutions for the n CSPs.
Let P be the set of pieces of work obtained from s^v
Let $\{P_1, P_2, \ldots, P_n\}$, where P_i is the set of pieces of work associated with the i-th depot and $P_i \subseteq P$
$S^c \leftarrow \{\}$
for $i \leftarrow 1$ **to** n **do**
  Let s_i^c be an initially empty solution for the CSP associated with the i-th depot
  if $P_i \neq \emptyset$ **then**
    $s_i^c \leftarrow P_i$ // as described in Subsection V-E3, Algorithm 7
    $s_i^c \leftarrow VND\_CSP(s_i^c)$ // as described in Subsection V-E4, Algorithm 8
  $S^c \leftarrow S^c \cup \{s_i^c\}$

Algorithm 3: *perturbation*

Data: $S^{vc} = [s^v, S^c]$, a solution for the MDVCSP. s^v is the solution for the MDVSP and $S^c = \{s_1^c, s_2^c, \ldots, s_n^c\}$ is the set of solutions for the n CSPs (for a problem with n depots).
Data: l, the level of perturbation.
Result: $S_2^{vc} = [s_2^v, S_2^c]$, the solution perturbed.
$s_2^v \leftarrow perturb(s^v, l)$
Let $\{R_1, R_2, \ldots, R_n\}$, where R_i is the set of pieces of work removed from the i-th depot, that is, the pieces associated with s^v and not associated with s_2^v
Let $\{C_1, C_2, \ldots, C_n\}$, where C_i is the set of pieces of work created in the i-th depot, that is, the pieces associated with s_2^v and not associated with s^v
$S_2^c \leftarrow \emptyset$ // the new set of solutions for the n CSPs
for $i \leftarrow 1$ **to** n **do**
  Let $s_i^c \in S^c$
  $s_i^c \leftarrow s_i^c \setminus R_i$
  $piecesOrder(C_i, sortPieces\_pert)$ // Defines the pieces' order in C_i
  while $C_i \neq \emptyset$ **do**
    Let p be the first piece of work from C_i
    $C_i \leftarrow C_i \setminus \{p\}$
    $s_i^c \leftarrow s_i^c \cup \{p\}$ // as described in Subsection V-E2, Algorithm 6
  $S_2^c \leftarrow S_2^c \cup \{s_i^c\}$
$S_2^{vc} \leftarrow [s_2^v, S_2^c]$ // perturbed solution for the MDVCSP

In Subsection V-E1, we show how to split the vehicle schedule into pieces of work. Then, in Subsection V-E2, we present the heuristic procedure for adding a new piece of work to a partial solution for the CSP. Finally, in Subsections V-E3 and V-E4, we detail the heuristic procedures for generating an initial solution and local search for the CSP.

1) HEURISTIC METHODS FOR GENERATING THE PIECES OF WORK

To solve the CSP, we first split the vehicle itineraries originated from an MDVSP solution into pieces of work. To exemplify this operation, assume that:

- All start and end trip points are also relief points. That is, each vehicle trip is a task;
- The minimum and maximum durations of a piece of work are, respectively, 30 minutes and 5 hours.

Table 2 shows a vehicle's itinerary split into three pieces of work. On the left is the vehicle itinerary and, on the right, the pieces of work formed (*Piece 1*, *Piece 2*, *Piece 3*).

We note that a crew is responsible for a vehicle throughout its working day. This responsibility includes:

1) the trips themselves attributed to the vehicle,
2) the moves for positioning the vehicle in the appropriate places (i.e., the so-called deadheads),
3) the waiting time at the stations to wait for the next trip to start.

Thus, the following attributes are associated with each piece of work:

- Expanded start time: Piece of work start time;
- Expanded start point: Location where the piece of work starts;
- Expanded end time: Piece of work end time;
- Expanded end point: Location where the piece of work ends;
- Duration: Duration of the piece of work.

In Table 2, *Piece 1* starts at 7:53 am in depot G1, ends at 11:34 am at point B, and has a total duration of three hours and forty-one minutes. *Piece 2* starts at 11:34 am at point B, ends at 4:14 pm also at point B, and has a total duration of four hours and forty minutes. Therefore, we observe that the *Piece 1* ends precisely at the place and time when the *Piece 2* starts. The same situation occurs between the pieces of work *Piece 2* and *Piece 3*. In this way, the vehicle is never without a crew outside the depot.

We propose two heuristic procedures to define the pieces of work associated with the vehicle itinerary. They are: direct partitioning and reverse partitioning. The objective in both procedures is to build pieces of work that have as many trips as possible and do not include waiting times in the depot.

TABLE 2. Example of direct partitioning of the vehicle itinerary into pieces of work.

Vehicle Itinerary					Piece of Work					
Trip Number	Start Time	Start Point	End Time	End Point	Exp. Start Time	Exp. Start Point	Exp. End Time	Exp. End Point	Duration	
-	7:53	G1	8:35	A						
1	8:35	A	9:18	D						
-	9:18	D	9:42	A	7:53	G1	11:34	B	3:41	*Piece 1*
2	9:55	A	10:33	D						
-	10:33	D	10:55	B						
3	11:34	B	12:24	C						
4	13:31	C	13:56	A	11:34	B	16:14	B	4:40	*Piece 2*
5	14:01	A	14:26	C						
6	14:30	C	15:23	B						
7	16:14	B	17:07	C						
8	17:31	C	17:56	A						
9	18:01	A	18:26	C	16:14	B	20:04	G1	3:50	*Piece 3*
10	18:30	C	19:23	B						
-	19:23	B	20:04	G1						

Algorithm 4: *localSearch*

Data: $S^{vc} = [s^v, S^c]$, a perturbed solution for the MDVCSP. s^v is the solution for the MDVSP and $S^c = \{s_1^c, s_2^c, \ldots, s_n^c\}$ is the set of solutions for the n CSPs (for a problem with n depots).
Result: $S_2^{vc} = [s_2^v, S_2^c]$, the solution for the MDVCSP after local search.

$S_2^c \leftarrow \emptyset$ // the new set of solutions for the n CSPs
for $i \leftarrow 1$ **to** n **do**
  Let $s_i^c \in S^c$
  if s_i^c *has been modified* **then**
    $s_i^c \leftarrow VND\_CSP(s_i^c)$ // as described in Subsection V-E4, Algorithm 8
  $S_2^c \leftarrow S_2^c \cup \{s_i^c\}$
$s_2^v \leftarrow s^v$ // solution for the MDVSP does not change
$S_2^{vc} \leftarrow [s_2^v, S_2^c]$ // the solution for the MDVCSP after local search

Algorithm 5: *acceptanceCriterion*

Data: S_*^{vc}, the current solution for the MDVCSP.
Data: S^{vc}, the solution for the MDVCSP after perturbation and local search.
Data: l, the current level of perturbation of the ILS-MDVCSP.
Data: l_{max}, the maximum level of perturbation.
Result: A pair $[S_*^{vc}, l]$, where S_*^{vc} is the updated current solution for the MDVCSP and l is the updated level of perturbation.

if $f^{vc}(S^{vc}) < f^{vc}(S_*^{vc})$ **then**
  $S_*^{vc} \leftarrow S^{vc}$
  $l \leftarrow 1$
if $f^{vc}(S^{vc}) = f^{vc}(S_*^{vc})$ **then**
  $S_*^{vc} \leftarrow S^{vc}$
  $l \leftarrow l + 1$
if $f^{vc}(S^{vc}) > f^{vc}(S_*^{vc})$ **then**
  $l \leftarrow l + 1$
if $l > l_{max}$ **then**
  $l \leftarrow 1$

In direct partitioning, the procedure starts in the depot, before the vehicle's first trip on the day. The procedure systematically covers the entire vehicle itinerary, trip by trip, and forms pieces of work. In this approach, a piece of work always:

- starts in the depot or at the associated first trip beginning;
- ends in the depot or at the first trip beginning of the next piece of work.

Table 2 illustrates an example of the generation of pieces of work by direct partitioning.

In the reverse partitioning procedure, the pieces of work' construction start in the depot after the vehicle's last trip. The method systematically goes through the entire vehicle itinerary, from back to front, building the pieces of work. In this case, a piece of work always:

- starts in the depot or at the end of the last trip of the previous piece of work;
- ends in the depot or at the associated last trip ending.

See an example of this type of partitioning in Table 3.

TABLE 3. Example of inverse partitioning of the vehicle itinerary into pieces of work.

Vehicle Itinerary					Piece of Work					
Trip Number	Start Time	Start Point	End Time	End Point	Exp. Start Time	Exp. Start Point	Exp. End Time	Exp. End Point	Duration	
-	7:53	G1	8:35	A	7:53	G1	10:33	D	2:40	*Piece 1*
1	8:35	A	9:18	D						
-	9:18	D	9:42	A						
2	9:55	A	10:33	D						
-	10:33	D	10:55	B	10:33	D	15:23	B	4:50	*Piece 2*
3	11:34	B	12:24	C						
4	13:31	C	13:56	A						
5	14:01	A	14:26	C						
6	14:30	C	15:23	B						
7	16:14	B	17:07	C	15:23	B	20:04	G1	4:41	*Piece 3*
8	17:31	C	17:56	A						
9	18:01	A	18:26	C						
10	18:30	C	19:23	B						
-	19:23	B	20:04	G1						

There are many ways to partition a vehicle itinerary to build pieces of work. However, we chose to use only these two methods.

2) HEURISTIC METHOD FOR INSERTING A PIECE OF WORK INTO CREW SCHEDULES

Let be the following data:

- p: piece of work to be included in the schedule;
- s_0^c: solution partially built for the CSP and that always keeps an empty duty in the last position on its list of duties (see representation in Section V-A2).

We defined three basic methods for inserting p into s_0^c maintaining the feasibility of the solution, they are:

1) Random insertion - *randomInsertion*(p, s_0^c): We randomly select a crew from the schedule and, if the viability of s_0^c is maintained, we insert the piece of work p into its duty. As there is a possibility of failure, this procedure returns *true* if the insertion is effective and *false*, otherwise;
2) Sequential insertion - *sequentialInsertion*(p, s_0^c): The list of duties for the s_0^c solution is inspected sequentially, starting from the first position. The piece of work p is assigned to the first crew that can perform it;
3) Greedy insertion - *greedyInsertion*(p, s_0^c): We analyze the cost of inserting the piece of work p in each of the possible duties of s_0^c. In this way, p is allocated to the lowest cost duty according to the evaluation function (14).

Since we keep an empty duty in s_0^c, we will always find a duty to insert p in the sequential and greedy insertion procedures. That is, if there is no other possibility, the empty duty of s_0^c will receive p.

Algorithm 6 shows the heuristic procedure for inserting a piece of work p into the partially built crew schedule s_0^c. This procedure combines the three primary methods presented above (random, sequential, and greedy insertions).

According to Algorithm 6, we initially tried, at most it_{max} times, to randomly insert p in a duty (lines 7 to 9). If we can't, we randomly choose one of the two remaining types of insertions, sequential or greedy, to insert the piece of work p (lines 10 and 11). To define $it_{\max}$ (line 4), we consider the percentage of duties in s_0^c that we can test. This percentage is given by *percDutiesTested* and consists of a parameter whose value we need to specify. Finally, in lines 12 and 13, we guarantee that s_0^c will have an empty duty in the last position in its duty list.

3) HEURISTIC METHOD FOR GENERATING AN INITIAL SOLUTION FOR THE CSP

Algorithm 7 describes the heuristic method that generates an initial solution for the CSP. For this procedure, we must supply the set of pieces of work of the associated vehicle schedule. Due to the construction process, in set P, each vehicle's pieces of work are arranged sequentially. However, we can also sort this set by the start time of the pieces of work. The parameter *sortPieces_const* defines the order of the pieces of work in the set P (line 3). Then, we allocate each piece of work to a crew, exactly as presented in Algorithm 6.

4) HEURISTIC METHOD OF LOCAL SEARCH FOR THE CSP

To improve a CSP solution, we propose a local search method based on the basic variable neighborhood descent (B-VND) [34] heuristic. This method is based on the principle that a local optimal concerning a given neighborhood structure does not necessarily correspond to a local optimum concerning another neighborhood structure. Thus, the objective is to explore the solution space through systematic changes of neighborhood structures.

Algorithm 8 presents the pseudo-code of the developed method, the VND-CSP. We do not apply local searches with the best improvement strategy to the current solution,

Algorithm 6: *insertPiece_Solution*

Data: p, a piece of work to insert in the solution.
Data: s_0^c, a solution partially built for the CSP. It has an empty duty at the last position in its duties list.
Result: s_1^c, a partially built solution for the CSP that includes the piece of work p. It has an empty duty at the last position in its duties list.
$it_{max} \leftarrow percDutiesTested \times$ number of duties in s_0^c
`// the maximum number of attempts to insert p randomly into` s_0^c
$it \leftarrow 0$ `// the number of attempts made to insert p randomly into` s_0^c
$inserted \leftarrow false$ `// boolean variable that will be true if p is inserted randomly into` s_0^c
while $it < it_{max}$ *and* $inserted = false$ **do**
    $inserted = randomInsertion(p, s_0^c)$
    $it \leftarrow it + 1$
if $inserted = false$ **then**
    $sequentialInsertion(p, s_0^c)$ or $greedyInsertion(p, s_0^c)$, choose randomly between one of these procedures
if s_0^c *does not have an empty duty* **then**
    Insert empty duty in the last position in the s_0^c duty list
$s_1^c \leftarrow s_0^c$

Algorithm 7: *constructive_CSP*

Data: P, the set of pieces of work.
Result: s^c, a solution for the CSP.
$piecesOrder(P, sortPieces\_const)$ `// Defines the pieces' order in` P
Start s^c with just one empty duty
while $P \neq \emptyset$ **do**
    Let p be the first piece of work from P
    $P \leftarrow P \setminus \{p\}$
    $s^c \leftarrow s^c \cup \{p\}$ `// as described in Subsection V-E2, Algorithm 6`

given the high computational cost of evaluating the entire neighborhood at each iteration. Thus, we employ only random descent methods, described below:

1) *RandomDescentWithRelocate_CSP*: explores the neighborhood N_r^c (relocation of a piece of work);
2) *RandomDescentWithExchange_CSP*: explores the neighborhood N_e^c (exchange of pieces of work).

The *RandomDescentWithRelocate_CSP* method is a random descent for the CSP that works as follows. Given a CSP solution s^c, we apply a move to it, generating a neighbor s_1^c. If s_1^c represents an improvement for the current solution's value s^c, then this neighbor becomes the new current solution; otherwise, we randomly generate a new neighbor. If after it_r_{max} iterations we do not find an improved solution, then we finish this random descent method. The value of it_r_{max} is estimated by (16):

$$it\_r_{max} = \mid N_r^c(s^c) \mid \times perc\_r, \tag{16}$$

where:
(a) it_r_{max} is the maximum number of iterations with no improvement in the current solution;
(b) $N_r^c(s^c)$ is the set of neighbors of s^c concerning the neighborhood structure N_r^c;
(c) $perc\_r$ is the percentage of the size of the neighborhood $N_r^c(s^c)$ to be explored.

Note that this calculation links $it\_r_{\max}$ to the instance's size to be solved. Thus, as we explore only part of the neighborhood, there is a reduction in its evaluation cost.

As the Algorithm 8 shows, we end the VND-CSP method when there is no improvement in the two neighborhood structures, N_r^c and N_e^c. In this case, the method returns a local optimum concerning these two neighborhoods.

In the VND-CSP, we apply the *RandomDescentWithRelocate_CSP* and *RandomDescentWithExchange_CSP* procedures in this order. Initially, we apply a random descent in the neighborhood N_r^c. If there is no improvement in $it\_r_{\max}$ iterations, then we apply the *RandomDescentWithRelocate_CSP* method (that is, a random descent in the neighborhood N_e^c). If there is an improvement in this neighborhood, we return to the first neighborhood; otherwise, the method seeks an improved solution in a maximum of $it\_e_{\max}$ iterations. We estimate the value of $it\_e_{\max}$ similarly to (16), replacing $perc\_r$ with $perc\_e$. The VND method ends when there is no improvement in the current solution in both neighborhoods.

VI. COMPUTATIONAL EXPERIMENTS

The proposed matheuristic algorithm was developed in the C++ language and compiled with version 9.3.0 of gcc. The mathematical models developed for the MDVSP were solved using the standard configuration (except for the number of threads) of the Gurobi solver version 9.0.3. The experiments were performed on an Intel (R) Xeon (R) E5-2640 (2.50 GHz) microcomputer and under the 64-bit GNU Linux Ubuntu 20.04.1 LTS operating system. All experiments were run using a single thread only.

To perform the computational experiments, we use the instances of Huisman *et al.* [22], which are widely used in the literature. We also consider real-world instances originating from the public urban bus transport system from one of the largest Brazilian cities (Belo Horizonte/MG). In Sections VI-A and VI-B, we describe the characteristics of these instances and present the results

Algorithm 8: VND-CSP

Data: s^c, a solution for the CSP.
Result: s^c_*, the solution for the CSP after local search.
$exit \leftarrow false$
while $exit \neq true$ **do**
  // Random Descent in the neighborhood structure N^c_r
  for $i \leftarrow 1$ **to** it_r_{max} **do**
    Let $s^c_1 \in N^c_r(s^c)$
    if $f(s^c_1) < f(s^c)$ **then**
      $s^c \leftarrow s^c_1$
      $i \leftarrow 1$
    else
      $i \leftarrow i + 1$
  // Random Descent in the neighborhood structure N^c_e
  for $i \leftarrow 1$ **to** it_e_{max} **do**
    Let $s^c_2 \in N^c_e(s^c)$
    if $f(s^c_2) < f(s^c)$ **then**
      break
    $i \leftarrow i + 1$
  if $f(s^c_2) < f(s^c)$ **then**
    $exit \leftarrow false$
    $s^c \leftarrow s^c_2$
  else
    $exit \leftarrow true$
$s^c_* \leftarrow s^c$

obtained by the proposed algorithm, respectively. The benchmark and Brazilian real-world instances, executable code of the implemented algorithms, and all scripts to run the executable and solve the instances are available for download at http://sites.google.com/view/mdvcsp-instances (generally, it is not accessible from a Google Workspace account).

The cost values used in the evaluation functions presented for the MDVSP (13), CSP (14) and MDVCSP (15) are the same ones used in [3]. Table 4 shows these costs. According to it, there is a fixed cost of 1000 units for each vehicle and crew, a variable cost of 1 unit for each minute a vehicle is outside the depot, and a variable cost of 0.1 unit for each minute crew working time. So, these costs prioritize the minimization of the number of vehicles and crews employed. The reduction of operating costs (variable costs) is a secondary objective.

TABLE 4. Costs considered in the evaluation functions and their respective values.

Type of cost	Value
vehicleCost	1000
crewCost	1000
operationalCost	1
workingTimeCost	0.1

TABLE 5. Characteristics of the different duty types.

	Early		*Day*		*Late*		*Split*	
	Min	Max	Min	Max	Min	Max	Min	Max
Interval		16:30	8:00	18:14	13:15			19:30
Piece length	0:30	5:00	0:30	5:00	0:30	5:00	0:30	5:00
Break length	0:45		0:45		0:45		1:30	
Duty length		9:45		9:45		9:45		12:00
Working time		9:00		9:00		9:00		9:00

A. LITERATURE INSTANCES

In this section, we describe the instances of the literature in Subsection VI-A1, the tuning process of our algorithm in Subsection VI-A2, and compare the results of our algorithm with those of other methods from the literature in Subsection VI-A3.

1) INSTANCES DESCRIPTION

We use eight groups of instances that depict characteristics of European public transport companies. They are differentiated by the number of trips, which can be: 80, 100, 160, 200, 320, 400, 640, or 800 trips. Each group contains ten instances, totaling 80 instances. The first six groups of instances were made available in [49], and the last two in [50]. Huisman and Steinzen generated all of these instances from the software described in [53] and [22]. These instances are widely used in the literature (see Table 1 in Section II) and have the following characteristics:

- There are four depots;
- Depots have unlimited capacity. That is, the number of vehicles and crews available for each depot is unlimited;
- Each timetable trip can be serviced by any depot;
- Instances with 80, 160, and 320 trips have four *relief points*, and the others have five.

Additionally, according to Huisman *et al.* [22], whenever a trip starts or ends in the depot, a crew preparation time of 10 and 5 minutes is required, respectively. However, a trip can start or end at any other exchange point (outside the depot), and, in this case, the crew preparation time will be the travel time between the exchange point and the depot plus 15 minutes.

Still according to Huisman *et al.* [22], there are five types of duties: *tripper*, formed by only one piece of work lasting between 30 minutes and 5 hours; and four more types (*early*, *day*, *late*, and *split*), made up of two pieces of work. Table 5 details them, whose characteristics are as follows:

TABLE 6. Parameters of the proposed algorithm.

Parameter	Description	Tested and returned values
$percDutiesTested$	Percentage of crew members evaluated. It is used in Algorithm 6, described in Subsection V-E2.	$0.0, \mathbf{0.25}, 0.5, 0.75, 1.0$
$sortPieces\_const$	Defines whether or not the set of pieces of work should be ordered by the pieces' start time. It is used in Algorithm 7 ($constructive\_CSP$), presented in Subsection V-E3.	(a) Yes (b) **No**
$sortPieces\_pert$	Defines whether the set of pieces of work should be ordered by the pieces' start time. It is used in Algorithm 3 ($perturbation$), described in Subsection V-D.	(a) Yes (b) No (c) **Set randomly**
$perc\_r$	Percentage of the size of the neighborhood $N_r^c(s^c)$ to be explored. It is used in (16) of Subsection V-E4.	$0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, \mathbf{0.7}, 0.8, 0.9, 1.0$
$perc\_e$	Percentage of the size of the neighborhood $N_r^c(s^c)$ to be explored. It is defined in Subsection V-E4.	$0.0, 0.1, \mathbf{0.2}, 0.3, 0.4, 0.5, 0.6, 0.7$
l_{max}	Perturbation levels used in the ILS-MDVCSP matheuristic algorithm. Levels 1 to 6 are described in Subsection V-D.	(a) From 1 to 3 (b) From 1 to 4 (c) From 1 to 5 (d) **From 1 to 6**

- Interval: Interval of the day on which a duty can take place. The absence of a lower or upper limit indicates that there is no restriction on the time of beginning or end of the duty, respectively;
- Piece length: Minimum and maximum length allowed for a piece of work. It is noteworthy that during a piece of work, the crew will remain responsible for the same vehicle without interruption;
- Break length: Minimum break duration between the pieces of work of the duty;
- Duty length: Maximum duration of the duty considering all activities: preparation to start or end the duty, monitoring of the vehicle and mandatory breaks;
- Working time: Maximum time a crew can spend on the vehicle. It is the sum of the duration of the duty' pieces of work.

2) PARAMETER SETTINGS

For tuning the developed ILS-MDVCSP algorithm's parameters, we apply the irace package [54]. This software tunes the parameters of optimization algorithms. The authors developed it in the R language and implemented an extension of the Iterated F-race algorithm (I / F-Race) [55].

Irace receives as input a set of values assumed by the parameters, a set of instances for tuning those parameters, and a set of options for running irace.

In Table 6, we present the analyzed parameters, describe their meanings, the tested values, and highlight in bold the values returned by irace.

Among the eight groups of instances of the literature defined in Section VI-A1, we randomly choose an instance from each group of instances with 80, 200, 400, and 800 trips for the irace tuning phase.

We run irace in its default configuration, except for the option *maxExperiments*, for which we assign the value 250. This option defines the number of times that the ILS-MDVCSP algorithm will be executed during the tuning process.

From statistical tests, irace iteratively generates and tests different parameter configurations for the ILS-MDVCSP algorithm. At the end of its execution, irace returns the elite configuration, that is, the configuration for the parameters that provided the best average performance of the algorithm.

According to the values tested for each parameter in Table 6, there are more than 10500 possible parameter configurations. Therefore, the manual tuning of the ILS-MDVCSP algorithm's parameters would be highly costly and possibly inefficient, which justifies the use of irace.

For more details on the irace package, we recommend the user's guide [56].

3) RESULTS

All 80 instances were solved ten times by the ILS-MDVCSP algorithm. We fixed the run time of ILS-MDVCSP at 15 minutes for small instances (with 80, 100, 160, and 200 trips) and 40 minutes for the large ones (with 320, 400, 640, and 800 trips).

Table 7 reports the best results found by the proposed algorithm, comparing them with those of the literature methods, namely: HK_19 [26], KAA_12 [25], SGSK_10 [3], BLW_08 [23], and HFW_05 [22]. These papers deal with the MDVCSP as proposed in [22].

In Table 7, *cpu* is the average time, in minutes, reported for solving each group of instances. In line *cpu adj.*, we adjust this value to match the machine where the tests were performed with the machine described in [23], which has the most simple configuration machine considered in this comparison. For this match, we use the cpu benchmark website [57], which provides benchmark results for CPUs for more than 600,000 systems, covering more than 1200 different types of CPUs. This adjustment allows for a fair comparison of the run time of the approaches.

TABLE 7. Results from literature instances.

Approach	Group of instances							
	80	100	160	200	320	400	640	800
ILS-MDVCSP [1]								
cpu	15.00	15.00	15.00	15.00	40.00	40.00	40.00	40.00
cpu adj.	33.45	33.45	33.45	33.45	89.20	89.20	89.20	89.20
vehicles	**9.20**	**11.00**	**14.80**	**18.40**	**26.70**	**32.90**	**56.90**	**66.90**
crews	19.70	23.10	**31.70**	**38.50**	**55.80**	**67.90**	**119.40**	**142.20**
v + c	28.90	34.10	**46.50**	**56.90**	**82.50**	**100.80**	**176.30**	**209.10**
RPD (%)	3.21	2.40	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
HK_19 [2]								
cpu	15.24	16.78	>> 180	-	-	-	-	-
cpu adj.	32.77	36.08	>> 387,00	-	-	-	-	-
vehicles	9.50	11.40	-	-	-	-	-	-
crews	**18.50**	**21.90**	-	-	-	-	-	-
v + c	**28.00**	**33.30**	50.50	-	-	-	-	-
RPD (%)	**0.00**	**0.00**	8.60	-	-	-	-	-
KAA_12 [3]								
cpu	5.43	8.72	22.80	30.40	158.85	183.63	455.60	-
cpu adj.	11.13	17.88	46.74	62.32	325.64	376.44	933.98	-
vehicles	**9.20**	**11.00**	**14.80**	**18.40**	**26.70**	**32.90**	-	-
crews	19.20	22.80	**31.70**	39.00	56.40	69.50	-	-
v + c	28.40	33.80	**46.50**	57.40	83.10	102.40	178.50	-
RPD (%)	1.43	1.50	**0.00**	0.88	0.73	1.59	1.25	-
SGSK_10 [4]								
cpu	3.92	6.15	26.32	45.50	238.75	338.67	953.92	-
cpu adj.	4.31	6.77	28.95	50.05	262.63	372.53	1049.31	-
vehicles	**9.20**	**11.00**	**14.80**	**18.40**	**26.70**	**32.90**	**56.90**	-
crews	19.10	22.70	31.80	38.80	**55.80**	**67.90**	120.40	-
v + c	28.30	33.70	46.60	57.20	**82.50**	**100.80**	177.30	-
RPD (%)	1.07	1.20	0.22	0.53	**0.00**	**0.00**	0.57	-
BLW_08 [5]								
cpu	13.00	21.00	44.00	106.00	328.00	720.00	-	-
cpu adj.	13.00	21.00	44.00	106.00	328.00	720.00	-	-
vehicles	**9.20**	11.20	15.00	18.50	**26.70**	33.10	-	-
crews	20.40	24.50	32.70	40.50	56.10	68.90	-	-
v + c	29.60	35.70	47.70	59.00	82.80	102.00	-	-
RPD (%)	5.71	7.21	2.58	3.69	0.36	1.19	-	-
HFW_05								
cpu	-	-	-	-	-	-	-	-
cpu adj.	-	-	-	-	-	-	-	-
vehicles	**9.20**	**11.00**	**14.80**	**18.40**	-	-	-	-
crews	20.50	25.30	34.10	41.60	-	-	-	-
v + c	29.70	36.30	48.90	60.00	-	-	-	-
RPD (%)	6.07	9.01	5.16	5.45	-	-	-	-

[1] Intel Xeon E5-2640 2.50 GHz/4 GB using only one single thread.
[2] Intel Xeon X5650 2.67 GHz/4 GB using only one single thread.
[3] Dell OptiPlex 755, Intel Core 2 Duo 3.0 GHz/4 GB using only one single thread.
[4] Dell OptiPlex GX620, Intel Pentium IV 3.4 GHz/2 GB using only one single thread.
[5] Dell Precision 650, Intel Dual Xeon 3.0 GHz/4 GB using only one single thread.

Lines *vehicles*, *crews*, and $v + c$ of Table 7 report the average number of vehicles, crews, and sum of vehicles and crews for each group of instances, respectively. This table does not report the following characteristics used to evaluate a solution ((13) and (14)): vehicle operating time, the crew working time, and cost. We did so because Kliewer *et al.* [25], Steinzen *et al.* [3], Borndörfer *et al.* [23], and Huisman *et al.* [22] did not provide this information.

We use the Relative Percentage Deviation (RPD_i^{Alg}) to evaluate the average sum of vehicles and crews, $v + c$, generated by each method *Alg* for the group of instances i. It is calculated according to (17):

$$RPD_i^{Alg} = \frac{(v+c)_i^{Alg} - (v+c)_i^{best}}{(v+c)_i^{best}}, \tag{17}$$

where $(v + c)_i^{Alg}$ is the $v + c$ obtained by method *Alg* for the group of instances i and $(v + c)_i^{best}$ is the best known $v + c$ for the group of instances i. Then, the RPD_i^{Alg} informs the deviation percentage of the $v + c$ found by method *Alg* concerning the best known $v + c$ for the group of instances i.

TABLE 8. Characteristics of the Belo Horizonte instances.

#id	Instances	Number of trips	Time of trips (h:m)	Number of relief points	Fleet size	Number of depots
1	D01_MON	260	443:19	3	41	1
2	D01_FRI	260	453:39			
3	D01_SAT	172	270:40			
4	D01_SUN	90	158:33			
5	D02_MON	468	527:02	3	35	1
6	D02_FRI	468	524:27			
7	D02_SAT	359	388:32			
8	D02_SUN	298	315:00			
9	D03_MON	206	406:32	2	29	1
10	D03_FRI	203	407:43			
11	D03_SAT	130	255:20			
12	D03_SUN	108	218:02			
13	D01-D02_MON	728	970:21	4	D01: 41 D02: 35	2
14	D01-D02_FRI	728	978:06			
15	D01-D02_SAT	531	659:02			
16	D01-D02_SUN	388	473:33			
17	D01-D02-D03_MON	934	1376:53	5	D01: 41 D02: 35 D03: 29	3
18	D01-D02-D03_FRI	931	1385:49			
19	D01-D02-D03_SAT	661	914:21			
20	D01-D02-D03_SUN	496	691:35			

In Table 7, we present the RPD for each method and group of instances that we are considering. Moreover, the best results for each evaluation criterion (number of vehicles, crews, vehicles plus crews, and RPD) are highlighted in bold. Note that the methods in the literature did not handle some large instances. In these cases, no information appears in Table 7. Besides, Huisman *et al.* [22] did not report the run time of their algorithm and Kliewer *et al.* [25] did not detail the average number of vehicles and crews separately for the group of instances with 640 trips.

Concerning the average sum of vehicles and crews $v + c$ of Table 7, we can see that the proposed algorithm gives the smallest values for six of the eight analyzed groups of instances. Furthermore, our algorithm is the only one to find the best results on instance groups with 200, 640, and 800 trips. It only loses to HK_19, KAA_12, and SGSK_10 in small instances, involving 80 and 100 trips.

Regarding the average number of vehicles in Table 7, the ILS-MDVCSP algorithm was able to obtain the smallest values for all the groups of instances. So, after the MDVSP solving in optimality by the proposed algorithm, the improvement step of the solution for MDVCSP has not compromised the quality of the vehicle scheduling.

The HK_19 method is the only one that obtains the exact solution to the problem. However, this approach was able to solve only small instances, with 80 or 100 trips. Of the 20 instances considered, they found the optimal solutions in four instances, and, for seven instances, the lower limit gap was less than 0.5%. As shown in Table 7, they tested instances with 160 trips, but the procedure for generating columns at the root node consumed alone more than three hours on average. Thus, for large instances, the solution process is very time-consuming and fails to generate better quality solutions than those found in the present work.

Regarding the adjusted run time of Table 7, as the instance's size increases, our approach becomes substantially less costly than the others presented in the literature. For the largest group of instances treated by KAA_12 and SGSK_10, with 640 trips, the ILS-MDVCSP algorithm performed best with less than 10% of the processing time they used. Besides, our algorithm was the only one to treat the group of instances with 800 trips. These observations show the ability of the proposed algorithm to handle large instances satisfactorily.

B. BELO HORIZONTE INSTANCES

1) INSTANCES DESCRIPTION

The real-world instances considered in this work come from a given region of Belo Horizonte/MG, Brazil. In these instances, there are three depots (D01, D02, and D03), which operate on four days of the week with different timetables (Monday, Friday, Saturday, and Sunday). Timetables from Tuesday to Thursday are the same as Monday.

The companies studied firstly assign trips to depots. Then, they generate the vehicle and crew schedules sequentially, considering one depot at a time. In Table 8, instances 1 to 12 were provided by some companies. The other instances were created by us from the original ones and considered two or three depots together. In all instances, each depot contains a limited fleet of identical vehicles. Table 8 shows the characteristics of these instances. For each instance, we report the number of trips, the total time of trips (in the format hours: minutes), the number of relief points,

TABLE 9. Results from the Belo Horizonte instances.

Day	Feature	Depots D01 and D02			Depots D01, D02 and D03		
		Company	ILS-SDVCSP	ILS-MDVCSP	Company	ILS-SDVCSP	ILS-MDVCSP
MONDAY	vehicles	75.0	70.0	67.0	104.0	96.0	92.0
	crews	-	139.9	131.9	-	199.5	187.8
	v + c	-	209.9	198.9	-	295.5	279.8
	vehicle time	58,221.0	51,276.3	49,997.7	82,613.0	73,986.1	70,937.8
	crew time	-	62,984.9	61,714.0	-	91,991.7	88,322.0
	D01: v / c	41.0 / -	40.0 / 71.8	32.0 / 65.6	41.0 / -	40.0 / 71.8	41.0 / 84.8
	D02: v / c	34.0 / -	30.0 / 68.1	35.0 / 66.3	34.0 / -	30.0 / 68.1	35.0 / 66.8
	D03: v / c	-	-	-	29.0 / -	26.0 / 59.6	16.0 / 36.2
	total cost	-	267,474.9	255,069.1	-	378,685.4	359,570.0
FRIDAY	vehicles	75.0	70.0	67.0	104.0	98.0	94.0
	crews	-	144.4	135.9	-	202.8	189.3
	v + c	-	214.4	202.9	-	300.8	283.3
	vehicle time	58,686.0	52,923.6	51,660.0	83,149.0	75,693.1	72,626.6
	crew time	-	64,717.9	63,897.0	-	94,103.0	90,639.0
	D01: v / c	40.0 / -	39.0 / 74.4	32.0 / 69.7	40.0 / -	39.0 / 74.4	41.0 / 84.3
	D02: v / c	35.0 / -	31.0 / 70.0	35.0 / 66.2	35.0 / -	31.0 / 70.0	35.0 / 67.1
	D03: v / c	-	-	-	29.0 / -	28.0 / 58.4	18.0 / 37.9
	total cost	-	273,795.5	260,949.7	-	385,903.5	364,990.5
SATURDAY	vehicles	50.0	43.0	40.0	66.0	57.0	51.0
	crews	-	89.2	81.4	-	123.3	108.2
	v + c	-	132.2	121.4	-	180.3	159.2
	vehicle time	39,552.0	35,421.5	34,065.3	54,872.0	49,454.3	45,807.7
	crew time	-	44,127.7	42,146.1	-	63,006.0	56,395.0
	D01: v / c	23.0 / -	23.0 / 39.1	5.1 / 8.5	23.0 / -	23.0 / 39.1	16.0 / 34.7
	D02: v / c	27.0 / -	20.0 / 50.1	34.9 / 72.9	27.0 / -	20.0 / 50.1	35.0 / 73.5
	D03: v / c	-	-	-	16.0 / -	14.0 / 34.1	0.0 / 0.0
	total cost	-	172,034.4	159,680.0	-	236,055.0	210,647.2
SUNDAY	vehicles	32.0	26.0	25.0	50.0	39.0	38.0
	crews	-	63.8	58.5	-	92.8	82.6
	v + c	-	89.8	83.5	-	131.8	120.6
	vehicle time	28,413.0	24,886.5	23,982.2	41,495.0	36,158.3	33,578.9
	crew time	-	31,602.3	29,595.5	-	46,567.3	41,228.7
	D01: v / c	11.0 / -	10.0 / 22.5	0.0 / 0.0	11.0 / -	10.0 / 22.5	3.1 / 7.5
	D02: v / c	21.0 / -	16.0 / 41.3	25.0 / 58.5	21.0 / -	16.0 / 41.3	34.9 / 75.1
	D03: v / c	-	-	-	18.0 / -	13.0 / 29.0	0.0 / 0.0
	total cost	-	117,846.8	110,441.8	-	172,615.1	158,301.8

the size of the available fleet, and the number of depots considered.

2) RESULTS

The hybrid algorithm ILS-MDVCSP developed for the MDVCSP was adapted to handle the SDVCSP (single-depot vehicle and crew scheduling problem) (ILS-SDVCSP). To this end, we use the same mathematical model developed for the MDVSP to solve the SDVSP and consider only the ILS-MDVCSP perturbation levels 2 to 6, described in Section V-D. We prevented the depot exchange.

We ran the ILS-SDVCSP and ILS-MDVCSP algorithms ten times for each instance. The results reported below refer to the averages obtained. We set the time t for each algorithm to run at $t = d$ hours, with d being the number of depots of the instance.

Table 9 shows the features of the solutions obtained for the D01 and D02 depots; and D01, D02, and D03 depots. For the company and the ILS-SDVCSP algorithm, the data correspond to the union of the solutions obtained for each depot solved separately (instances 1 to 12 in Table 8). That is, only the ILS-MDVCSP solved the instances with, respectively, 2 and 3 depots simultaneously (instances from 13 to 20 in Table 8).

The attributes that have not yet been described and appear in Table 9 are: *vehicle time* – the total operating time of the vehicles (in minutes), *crew time* – the working time considering all crews (in minutes), and *total cost* – which represents the cost of the solution calculated as specified in Subsection V-C. We also show separately the number of vehicles (v) and crews (c) used by each depot (i.e., *Di: v / c* for the i-th depot).

As the companies did not provide their crew schedule, we did not report this information. Moreover, we consider the same types of duties proposed by Huisman *et al.* [22] for defining the crew scheduling in the ILS-SDVCSP and ILS-MDVCSP algorithms.

According to Table 9, for all instances, the ILS-SDVCSP algorithm solutions are better than those used by the companies considering the features *vehicles* and *vehicle time*. When we analyze the depots D01, D02, and D03 together, the ILS-SDVCSP saved on average 9 vehicles per day and

reduced on average by about 10% the daily operating time of the vehicles (considering the four days of the week with different timetables). This result shows that although the companies did not provide the crew schedules, their fleet's higher operational time indicates the need for more labor time (crews) concerning the solutions from the ILS-SDVCSP (see constraint C8 of Section III).

Table 9 also shows that the ILS-MDVCSP was the approach that obtained the best results. Its solutions are of higher quality than those of the companies and ILS-SDVCSP for all evaluation criteria. Regarding the companies' solutions, when we analyze the depots D01, D02, and D03 together, the ILS-MDVCSP saved on average 12 vehicles per day and reduced on average by about 15% the daily operating time of the vehicles (considering the four days of the week with different timetables). When we compared the solutions of the ILS-SDVCSP and ILS-MDVCSP algorithms, we observed that ILS-MDVCSP generated better vehicle and crew schedules for all instances. Thus, we show the relevance of considering more than one depot simultaneously, as already pointed out by the literature, and the potential of the matheuristic proposed in this work, particularly in the context of real-world and large-scale problems.

We also note that, for the instances with fewer trips, referring to Saturday and Sunday, it is possible to completely avoid using the depots without violating the other depots' capacities. In this way, a reduction in these depots' operating costs is allowed on these two days of the week.

VII. CONCLUSION

This work addressed the MDVCSP. This problem involves public transport companies by medium and large buses, which have more than one depot to manage their resources, i.e., the vehicle fleet and crews. In the MDVCSP solution, we deal with two problems in an integrated manner: the MDVSP and the CSP. That is, we simultaneously define vehicle and crew schedules. The objective is to minimize the costs involved and, at the same time, respect the operational restrictions and work regulations. The MDVCSP is an NP-hard optimization problem, and to solve it, we propose a matheuristic algorithm. Our algorithm, called ILS-MDVCSP, uses the Iterated Local Search framework to combine two methods: a Branch-and-Bound method to solve the MDVSP in optimality and a Variable Neighborhood Descent-based method to treat the associated CSPs.

For the tests, we initially used a set of instances well known in the literature. We compared our approach against the main strategies in the literature that addressed the same problem. Our matheuristic algorithm was able to treat instances with 800 trips, a dimension not yet addressed by the current specialized literature. Besides, it obtained the best results for six groups of instances out of the eight groups considered. The run time of our algorithm was shorter for most groups of instances, and, as the instance's size increased, our approach became substantially less costly as compared to the literature.

We also solved the MDVCSP of a region in the city of Belo Horizonte, MG, Brazil. To address the particularities of this problem, we proposed a mathematical formulation based on a time-space network to represent the MDVSP. Regarding the companies' solutions, our algorithm's solutions were considerably better.

Furthermore, we compared the solutions obtained from two integrated approaches: SDVCSP and MDVCSP. So, we developed two algorithms: ILS-SDVCSP and ILS-MDVCSP. The ILS-MDVCSP outperformed the ILS-SDVCSP for all instances.

Therefore, the experiments showed the effectiveness of our matheuristic algorithm to deal with real-world and large-scale problems. We also verified that solving the MDVSP and CSP problems in an integrated manner reduces costs concerning these problems' sequential resolution. Moreover, by considering more than one depot at the same time, we can further reduce costs.

The MDVCSP is a complex problem and little explored in the literature. In practice, this problem's efficient resolution can bring high savings to the public bus transport sector.

In future work, we intend to do tests considering different evaluation functions and cost values for the MDVCSP. In the problems we solved, the main objective was to reduce the number of vehicles and crews. However, depending on the context, companies may need to prioritize other schedule' characteristics. So, we plan to consider different real-world scenarios. Moreover, we aim to compare the performance of our algorithm against other methods based on mathematical programming and metaheuristics. Our goal will be to identify the potentials of different optimization techniques to address MDVCSP and related problems. Finally, we note that public bus transport companies often have to deal with delays and interruptions in schedule due to traffic conditions, mechanical troubles with vehicles, crews' no show, and other unexpected events. These delays and interruptions affect the quality of the service provided, user satisfaction, and the companies' operating costs. Thus, it would be interesting to propose an approach for MDVCSP that dynamically updates the schedule to adapt it to some identified unexpected events in real-time.

REFERENCES

[1] G. Desaulniers and M. D. Hickman, "Public transit," in *Handbooks in Operations Research and Management Science*, vol. 14, C. Barnhart and G. Laporte, Eds. Amsterdam, The Netherlands: Elsevier, 2007, pp. 69–127.

[2] O. J. Ibarra-Rojas, F. Delgado, R. Giesen, and J. C. Muñoz, "Planning, operation, and control of bus transport systems: A literature review," *Transp. Res. B, Methodol.*, vol. 77, pp. 38–75, Jul. 2015.

[3] I. Steinzen, V. Gintner, L. Suhl, and N. Kliewer, "A time-space network approach for the integrated vehicle- and crew-scheduling problem with multiple depots," *Transp. Sci.*, vol. 44, no. 3, pp. 367–382, Aug. 2010.

[4] M. Mesquita, M. Moz, A. Paias, and M. Pato, "A decomposition approach for the integrated vehicle-crew-roster problem with days-off pattern," *Eur. J. Oper. Res.*, vol. 229, no. 2, pp. 318–331, Sep. 2013.

[5] R. Borndörfer, C. Schulz, S. Seidl, and S. Weider, "Integration of duty scheduling and rostering to increase driver satisfaction," *Public Transp.*, vol. 9, nos. 1–2, pp. 177–191, Jul. 2017.

[6] S. Carosi, A. Frangioni, L. Galli, L. Girardi, and G. Vallese, "A matheuristic for integrated timetabling and vehicle scheduling," *Transp. Res. B, Methodol.*, vol. 127, pp. 99–124, Sep. 2019.
[7] S. S. G. Perumal, T. Dollevoet, D. Huisman, R. M. Lusby, J. Larsen, and M. Riis, "Solution approaches for integrated vehicle and crew scheduling with electric buses," *Comput. Oper. Res.*, vol. 132, Aug. 2021, Art. no. 105268.
[8] S. Er-Rbib, G. Desaulniers, I. El Hallaoui, and A. Bani, "Integrated and sequential solution methods for the cyclic bus driver rostering problem," *J. Oper. Res. Soc.*, vol. 72, no. 4, pp. 764–779, Apr. 2021.
[9] M. Liang, W. Wang, C. Dong, and D. Zhao, "A cooperative coevolutionary optimization design of urban transit network and operating frequencies," *Expert Syst. Appl.*, vol. 160, Dec. 2020, Art. no. 113736.
[10] M. Ball, L. Bodin, and R. Dial, "A matching based heuristic for scheduling mass transit crews and vehicles," *Transp. Sci.*, vol. 17, no. 1, pp. 4–31, Feb. 1983.
[11] I. Patrikalakis and D. Xerocostas, "A new decomposition scheme of the urban public transport scheduling problem," in *Proc. 5th Int. Comput.-Aided Transit Scheduling, Workshop*, 1992, pp. 407–425.
[12] K. Haase, G. Desaulniers, and J. Desrosiers, "Simultaneous vehicle and crew scheduling in urban mass transit systems," *Transp. Sci.*, vol. 35, no. 3, pp. 286–303, Aug. 2001.
[13] R. Freling, D. Huisman, and A. P. M. Wagelmans, "Models and algorithms for integration of vehicle and crew scheduling," *J. Scheduling*, vol. 6, no. 1, pp. 63–85, 2003.
[14] B. Laurent and J. K. Hao, "Simultaneous vehicle and crew scheduling for extra urban transports," in *Proc. New Frontiers Appl. Artif. Intell., Int. Conf. Ind., Eng. Other Appl. Appl. Intell. Syst. (IEA/AIE)*, vol. 5027, N. T. Nguyen, L. Borzemski, A. Grzech, and M. Ali, Eds. Berlin, Germany, 2008, pp. 466–475, doi: 10.1007/978-3-540-69052-8_49.
[15] E. M. L. Simões, G. R. Mateus, and M. J. F. Souza, "Algoritmo para programação integrada de veículos e tripulações no sistema de transporte público por Ônibus," in *Proc. 13th Simpósio Brasileiro de Pesquisa Operacional*, 2011, pp. 1459–1471.
[16] N. Kliewer, T. Mellouli, and L. Suhl, "A time–space network based exact optimization model for multi-depot bus scheduling," *Eur. J. Oper. Res.*, vol. 175, no. 3, pp. 1616–1627, Dec. 2006.
[17] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman, "A comparison of five heuristics for the multiple depot vehicle scheduling problem," *J. Scheduling*, vol. 12, no. 1, pp. 17–30, 2009.
[18] L. Desfontaines and G. Desaulniers, "Multiple depot vehicle scheduling with controlled trip shifting," *Transp. Res. B, Methodol.*, vol. 113, pp. 34–53, Jul. 2018.
[19] S. Kulkarni, M. Krishnamoorthy, A. Ranade, A. T. Ernst, and R. Patil, "A new formulation and a column generation-based heuristic for the multiple depot vehicle scheduling problem," *Transp. Res. B, Methodol.*, vol. 118, pp. 457–487, Dec. 2018.
[20] A. T. Dauer and B. D. A. Prata, "Variable fixing heuristics for solving multiple depot vehicle scheduling problem with heterogeneous fleet and time Windows," *Optim. Lett.*, vol. 15, no. 1, pp. 153–170, Feb. 2021.
[21] A. Gaffi and M. Nonato, "An integrated approach to ex-urban crew and vehicle scheduling," in *Computer-Aided Transit Scheduling*. Berlin, Germany: Springer, 1999, pp. 103–128.
[22] D. Huisman, R. Freling, and A. P. M. Wagelmans, "Multiple-depot integrated vehicle and crew scheduling," *Transp. Sci.*, vol. 39, no. 4, pp. 491–502, Nov. 2005.
[23] R. Borndörfer, A. Löbel, and S. Weider, "A bundle method for integrated multi-depot vehicle and duty scheduling in public transit," in *Computer-Aided Systems in Public Transport*, M. Hickman, P. Mirchandani, and S. Voß, Eds. Berlin, Germany: Springer, 2008, pp. 3–24.
[24] M. Mesquita and A. Paias, "Set partitioning/covering-based approaches for the integrated vehicle and crew scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 5, pp. 1562–1575, 2008.
[25] N. Kliewer, B. Amberg, and B. Amberg, "Multiple depot vehicle and crew scheduling with time Windows for scheduled trips," *Public Transp.*, vol. 3, no. 3, pp. 213–244, Mar. 2012.
[26] M. Horváth and T. Kis, "Computing strong lower and upper bounds for the integrated multiple-depot vehicle and crew scheduling problem with branch-and-price," *Central Eur. J. Oper. Res.*, vol. 27, no. 1, pp. 39–67, Mar. 2019.
[27] I. Steinzen, M. Becker, and L. Suhl, "A hybrid evolutionary algorithm for the vehicle and crew scheduling problem in public transit," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 3784–3789.
[28] A. A. Bertossi, P. Carraresi, and G. Gallo, "On some matching problems arising in vehicle scheduling models," *Networks*, vol. 17, no. 3, pp. 271–281, 1987.
[29] M. Fischetti, S. Martello, and P. Toth, "The fixed job schedule problem with working-time constraints," *Oper. Res.*, vol. 37, no. 3, pp. 395–403, Jun. 1989.
[30] G. R. Raidl, "Decomposition based hybrid metaheuristics," *Eur. J. Oper. Res.*, vol. 244, no. 1, pp. 66–76, Jul. 2015.
[31] V. Maniezzo, T. Stützle, and S. Voss, "Matheuristics: Hybridizing metaheuristics and mathematical programming," in *Annals of Information Systems*, vol. 10. Cham, Switzerland: Springer, 2010.
[32] M. Gnägi and P. Baumann, "A matheuristic for large-scale capacitated clustering," *Comput. Oper. Res.*, vol. 132, Aug. 2021, Art. no. 105304.
[33] T. Stützle and R. Ruiz, "Iterated local search," in *Handbook Heuristics*, R. Marti, P. Pardalos, and M. Resende, Eds. Cham, Switzerland: Springer, 2018, pp. 579–605.
[34] P. Hansen, N. Mladenović, R. Todosijević, and S. Hanafi, "Variable neighborhood search: Basics and variants," *EURO J. Comput. Optim.*, vol. 5, no. 3, pp. 423–454, Sep. 2017.
[35] R. Borndörfer, A. Löbel, and S. Weider, "A bundle method for integrated multi-depot vehicle and duty scheduling in public transit," Konrad-Zuse Zentrum Fuer Informationstechnik, Berlin, Germany, Tech. Rep. ZR 04-14, 2004.
[36] S. E. G. Elias, *The Use of Digital Computers in the Economic Scheduling for Both Man and Machine in Public Transportation*, no. 49. Manhattan, KS, USA: Kansas State Univ., 1964.
[37] F. Kirkman, "Problems of innovation in the transport industry: A bus scheduling program," in *Proc. PTRC Public Transp. Anal. Seminar, Planning Transp. Res. Comput. Co. Ltd.*, vol. 1, 1968, pp. 1–15.
[38] J. L. Saha, "An algorithm for bus scheduling problems," *J. Oper. Res. Soc.*, vol. 21, no. 4, pp. 463–474, Dec. 1970.
[39] A. Wren, "Bus scheduling: An interactive computer method," *Transp. Planning Technol.*, vol. 1, no. 2, pp. 115–122, Sep. 1972.
[40] J. M. P. Booler, "A method for solving crew scheduling problems," *J. Oper. Res. Soc.*, vol. 26, no. 1, pp. 55–62, Apr. 1975.
[41] A. Wren, *Computer Scheduling of Public Transportation: Urban Passenger Vehicle and Crew Scheduling*. Amsterdam, The Netherlands: Elsevier, 1981.
[42] R. Freling, A. P. M. Wagelmans, and J. M. P. Paixão, "An overview of models and techniques of integrating vehicle and crew scheduling," in *Computer-Aided Transit Scheduling* (Lecture Notes in Economics and Mathematical Systems), vol. 471, N. H. M. Wilson, Ed. Berlin, Germany: Springer-Verlag, 1999, pp. 441–460.
[43] R. Freling, C. G. E. Boender, and J. M. P. Paix ao, "An integrated approach to vehicle and crew scheduling," Econ. Inst., Erasmus Univ. Rotterdam, Rotterdam, The Netherlands, Tech. Rep. 9503/A, 1995.
[44] R. Freling, "Models and techniques for integrating vehicle and crew scheduling," Ph.D. dissertation, Tinbergen Inst., Erasmus Univ. Rotterdam, Rotterdam, The Netherlands, 1997.
[45] V. Boyer, O. J. Ibarra-Rojas, and Y. Á. Ríos-Solís, "Vehicle and crew scheduling for flexible bus transportation systems," *Transp. Res. B, Methodol.*, vol. 112, pp. 216–229, Jun. 2018.
[46] A. Andrade-Michel, Y. A. Ríos-Solís, and V. Boyer, "Vehicle and reliable driver scheduling for public bus transportation systems," *Transp. Res. B, Methodol.*, vol. 145, pp. 290–301, Mar. 2021.
[47] B. Amberg, B. Amberg, and N. Kliewer, "Robust efficiency in urban public transportation: Minimizing delay propagation in cost-efficient bus and driver schedules," *Transp. Sci.*, vol. 53, no. 1, pp. 89–112, Feb. 2019.
[48] L. Kang, S. Chen, and Q. Meng, "Bus and driver scheduling with mealtime Windows for a single public bus route," *Transp. Res. C, Emerg. Technol.*, vol. 101, pp. 145–160, Apr. 2019.
[49] D. Huisman. (2003). *Random Data Instances for Multiple-Depot Vehicle Crew Scheduling*. Accessed: Sep. 6, 2019. [Online]. Available: http://people.few.eur.nl/huisman/instances.htm
[50] I. Steinzen. (2007). *Instances for Integrated Vehicle and Crew Scheduling Problems With Multiple Depots*. Accessed: Sep. 6, 2019. [Online]. Available: http://dsor.uni-paderborn.de/index.php?id=bustestset&L=0
[51] C. Ciancio, D. Laganà, R. Musmanno, and F. Santoro, "An integrated algorithm for shift scheduling problems for local public transport companies," *Omega*, vol. 75, pp. 139–153, Mar. 2018.
[52] I. Steinzen, "Topics in integrated vehicle and crew scheduling in public transport," Ph.D. dissertation, Fakultät für Wirtschaftswissenschaften der, Universität Paderborn, Paderborn, Germany, 2007.

[53] D. Huisman, "Integrated and dynamic vehicle and crew scheduling," Ph.D. dissertation, Tinbergen Inst., Erasmus Univ. Rotterdam, Rotterdam, The Netherlands, 2004.
[54] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Caceres, T. Stützle, and M. Birattari, "The irace package: Iterated racing for automatic algorithm configuration," *Oper. Res. Perspect.*, vol. 3, pp. 43–58, Jan. 2016.
[55] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, *F-Race Iterated F-Race: An Overview*. Berlin, Germany: Springer, 2010, 311–336.
[56] M. López-Ibáñez, L. P. Cáceres, J. Dubois-Lacoste, T. Stützle, and M. Birattari, "The irace package: User guide," IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR/IRIDIA/2016-004, 2016.
[57] PassMark Software Pty Ltd. (1998). *Cpu Benchmark Website*. Accessed: Jan. 18, 2021. [Online]. Available: https://www.cpubenchmark.net

EMILIANA MARA LOPES SIMÕES received the B.Sc. degree in computer science from the Federal University of Ouro Preto (UFOP), Ouro Preto, Minas Gerais, Brazil, in 2007, and the M.Sc. degree in computer science from the Federal University of Minas Gerais (UFMG), Belo Horizonte, Minas Gerais, in 2009, where she is currently pursuing the Ph.D. degree in electrical engineering. She is also a Professor with the Institute of Science and Technology, Federal University of the Vales do Jequitinhonha e Mucuri (UFVJM), Diamantina, Minas Gerais. Her main research interests include techniques of operation research, optimization, computational intelligence, scheduling, and public transport.

LUCAS DE SOUZA BATISTA received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering from the Federal University of Minas Gerais (UFMG), Brazil, in 2007, 2009, and 2011, respectively. He is currently an Associate Professor with the Department of Electrical Engineering, UFMG, and a member of the Operations Research and Complex Systems Laboratory (ORCS Lab), UFMG. He is involved with the electrical, systems, aerospace , and control and automation engineering courses, and the graduate program in electrical engineering. His research interests include optimization, computational intelligence, evolutionary computation, decision-making theory, and applications to engineering design problems. He has authored/coauthored over 50 peer-reviewed papers in journals and conferences, receiving peer recognition for his work published in conferences, winning best paper award certificates at the 14th Brazilian National Meeting on Artificial and Computational Intelligence (ENIAC 2017), the 13th Meeting of the Japanese Society of Evolutionary Computation (JSEC 2017), and the 18th IEEE Congress on Evolutionary Computation (CEC 2011).

MARCONE JAMILSON FREITAS SOUZA received the B.Sc. degree in metallurgical engineering from the Federal University of Ouro Preto (UFOP), Brazil, in 1982, and the M.Sc. and Ph.D. degrees in systems engineering and computing from the Federal University of Rio de Janeiro, Brazil, in 1989 and 2000, respectively. He also did a postdoctoral internship at the Institute of Computing, Fluminense Federal University, Brazil, in 2008. He is currently a Full Professor with the Department of Computing, UFOP. He has authored/coauthored 74 full articles in journals, 18 book chapters, and 228 full papers in conferences. His research interests include metaheuristics, scheduling, timetabling, open-pit mining, vehicle routing, public transport, machine learning applications, and operations research in the health area. His awards and honors include a research productivity fellowship granted by the Brazilian Council for Scientific and Technological Development (CNPq) in transport and production engineering, the Winner's Award of the Competition on Solution Methods for the Bi-Objective Traveling Thief Problem at the Tenth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2019), the Winner's Prize of the International Timetabling Competition 2011–2012 at the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012), and the Best Paper Award Certificates in the Area of Artificial Intelligence and Decision Support Systems at the 16th and 21st International Conference on Enterprise Information Systems (ICEIS 2014 and 2019).