Optimization Letters (2021) 15:153–170
https://doi.org/10.1007/s11590-020-01577-0

ORIGINAL PAPER

Variable fixing heuristics for solving multiple depot vehicle scheduling problem with heterogeneous fleet and time windows

Armando Teles Dauer[1] · Bruno de Athayde Prata[2]

Received: 23 April 2019 / Accepted: 27 March 2020 / Published online: 6 April 2020

Abstract
This paper aims at presenting the multiple depot vehicle scheduling problem with heterogeneous fleet and time windows (MDHFVSP-TW). We used a time-space network (TSN) to perform the modeling of MDHFVSP-TW, along with two methodologies to reduce its size and, therefore, its complexity. Along with size reduction methods, a mixed integer programming (MIP) heuristic with variable fixation was presented. Its operation is based on the use of the solution for this problem with relaxed variables as a basis for the removal of arcs from the problem, reducing its size and enabling its resolution in reasonable computational time. Extensive tests were performed for a collection of randomly generated instances. Subsequently, a case study arising from a real instance from a Brazilian city is presented. The computational results showed that the proposed heuristic and size reduction methods obtained good performance, providing high-quality solutions in an adequate computational time.

Keywords Combinatorial optimization · Public transport systems · Mixed integer linear programming · Machine learning

1 Introduction

Three main components shape a mass transit system: the passenger, who needs to know how to go from an origin to a destiny and how long it will take; the crew, who needs to know their working scheduling for the day and if they will need to change to different vehicles; and the vehicles, that must be allocated in a certain way so that

✉ Bruno de Athayde Prata
baprata@ufc.br

Armando Teles Dauer
armandodauer@hotmail.com

1 Federal University of Ceará, Campus de Crateús, Crateús, Brazil

2 Department of Industrial Engineering, Federal University of Ceará, Campus do Pici, Fortaleza, Brazil

the number of vehicles available meet the demand. Finding an efficient schedule that satisfies all the requirements of this system, regarding all the constraints, can be a complex and arduous task.

Ceder [4] evidences that transportation planning process is based on four essential components, usually in the following order: a network design, a defined timetable, the allocation of vehicles to the trips, and the assignment of a crew. The best-case scenario is when all components are planned simultaneously, maximizing the efficiency of the system. However, scheduling a mass transit system is a complex process, and applying an integrated approach to real-world size instances is extremely hard and demands a large amount of effort and computational time.

Ibarra et al. [16] explain that the focus of this process is to minimize the overall cost, to reduce the usage of the vehicles, to decrease the fuel consumption and the drivers' wage. The main problem can be divided into three smaller problems: the vehicle scheduling problem (VSP); the crew scheduling problem (CSP) and the crew rostering problem (CRP).

The VSP determines which group of trips will be assigned to each bus aiming to reduce the number (capital cost minimization) and the use of vehicles (operational cost minimization). The complexity of this problem is influenced by several characteristics, including: the number of depots, single depots VSP (SDVSP) can be solved in polynomial time, while multiple depot VSP (MDVSP) is an NP-hard problem as demonstrated by Bertossi et al. [2]; homogeneous or heterogeneous fleet (if the system operates different capacity vehicles with distinct operational costs); the number and location of relief points along with specific characteristics of a system such as inter-routine and how deadheads are implemented.

The possibility of delay or advance the start of a trip in a few minutes is a common practice in some cities in Brazil. This technique, named time windows, is applied to provide a higher number of connections and combinations, increasing the number of feasible solutions. This technique also allows a new connection of two trips (which otherwise would not be possible) producing an optimized solution, improving the use of resources while minimizing operating costs.

The main contributions of this paper are as follows. Firstly, we propose a new variant of the VSP that includes multiple depots, heterogeneous fleet, and the application of time windows (MDHFVSP-TW). According to our thorough literature review, this variant has not been modeled yet, despite its practical and theoretical importance. Secondly, we present a solution method for the MDHFVSP-TW combining the time-space network (TSN) representation and a mixed integer programming formulation. To reduce the size of the TSN and the computational time required to solve this problem, we also propose two TSN size-reduction techniques. The first technique reduces the number of arcs based on their cost, eliminating arcs that, because of their high cost, have little chance of composing the final solution. The second method is based on machine learning algorithms, using the knowledge produced from previously solved instances to reduce the size of the time-space network assertively. After that, an extensive computational evaluation is conducted to measure the robustness of the proposed approaches. Finally, we present a case study of a small instance arising from Brazilian city demonstrating the ability and efficiency of the proposed algorithms to solve real-world problems.

The remainder of this paper is organized in the following sections. Section 2 presents a literature review on the MDVSP. Section 3 introduces the problem statement, the methodology for the generation of the TSN along with the proposed mathematical model. Section 4 presents the proposed approaches. Section 5 presents the computational results obtained in a set of randomly generated instances. Section 6 presents a case study. Section 7 summarizes the main findings and proposes some topics for future research.

2 Literature review

In this section, we present related approaches to the variant under study aiming to highlight the contributions provided by the current research.

Dell'Ammico et al. [5] study a variation of the multiple depot VSP (MDVSP), intending to obtain a polynomial–time heuristic method capable of providing a low-cost feasible solution, guaranteeing the use of a minimal number of vehicles. The proposed algorithm is based on the shortest path approach and works in stages, in each stage determining a number of forbidden arcs and then finding a feasible circuit through the shortest path algorithm. Forbes et al. [8] present an exact algorithm approach for the MDVSP based on a linear programming relaxation and a subsequent branch and bound stage. Ribeiro and Soumis [23] propose a solution for the MDVSP applying a column generation algorithm to solve a continuous relaxation of this problem.

Lobel [19] presents a solution methodology based on a column generation heuristic applied to LP relaxation of a multi-commodity flow formulation. This method denominated Lagrangean Pricing, performs on arcs of complete paths and not only in individual arcs. Banihashemi and Haghani [1] carried out a study on a variant of the MDVSP, including route time constraints such as fuel consumption. Two techniques are proposed of a combined solution of MDVSP sub-problem and size reduction methodologies to allow the solution of large-size instances.

Huisman et al. [15] study focuses on dynamic scheduling and proposes a different heuristic approach to the MDVSP named dynamic vehicle scheduling aiming to reduce the number of late starting trips. This approach is divided into two steps, first, the trips are allocated among the different depots through the solution of the static problem, and then a series of single depot problems are solved.

Hadjar and Soumis [13] propose a branch-and-cut algorithm to solve the MDVSP. This branch-and-cut approach is improved with a column generation methodology to solve the VSP linear-programming relaxation, a variable fixation technique that sets the value of some variables and a cutting planes technique to reduce the number of solutions. Oukil et al. [21] aims to solve the long horizon MDVSP using the already well proven efficient column generation methodology. However, the column generation methodology faces problems when the instance is highly degenerate. To find a way of dealing with this problem, the study combined this methodology with preprocessing variable fixing and stabilization. Pepin et al. [22] compares five approaches for solving the MDVSP. These approaches include truncated branch-and-cut, truncated column generation, Lagrangian heuristic, tabu search and large neighborhood search heuristic using truncated column generation.

Guedes et al. [12] propose a fast approach to solve very large instances of the MDVSP using a two-stage procedure: first there is a reduction of the state space followed by the application of a truncated column generation heuristic. Two methods are proposed to reduce the state space, the first one is based on a single depot scheduling for each depot and the second one relaxes the depot returning constraint, allowing the vehicle to finish its journey in a different depot from where it started. Xu et al. [24] study a variant of the MDVSP with departure-duration restrictions. These restrictions ensure that the vehicles return to the depot when crews reach their working time limits. These authors propose a large neighborhood search improved by embedding a shortest path faster algorithm as a preliminary exploration tactic approach. Kulkarani et al. [18] propose a model for the MDVSP based on a minimum cost multi-commodity network flow formulation, considering vehicles from different depots as different commodities. This formulation is based on two types of constraints, flow conservations constraints and cover constraints. The tests were carried on randomly generated large sized instances, with 16 depots and 3000 trips.

Ginter et al. [9] aims to solve an MDVSP combining an exact time-space network (TSN) and a heuristic methodology to solve large real-world instances. The TSN is treated as a network flow problem, and a decomposition process is applied to obtain an optimal flow vehicle scheduling along with a fix and optimize heuristic. Kliewer et al. [17] study MDHFVSP applying a new modeling TSN generation technique that avoids the exponential growth of the model with the increase of the timetable. Guedes and Borestein [10] propose a 4-step methodology for solving the MDHFVSP. The proposed method initially generates the TSN and the connection network, then reduce the number of variables applying a state space reduction. After the reduction, initial solutions are generated and, finally, a column generation algorithm solves the reduced problem to find near-optimal solutions.

Desaulniers et al. [6] formulate the MDVSP with time windows as an integer non-linear multi-commodity network problem proposing its solution through a heuristic approach combination of column generation and branch-and-bound techniques. The model proposed considers exact waiting cost, instead of the common minimal consideration of minimal waiting costs. Hadjar and Soumis [14] studied the MDVSP with the implementation of up to 30 min time windows. The chosen methodology to solve the problem was a branch-and-price technique and, once the use of time windows dramatically increases the size of the TSN, a dynamic size reduction was applied to speed up the branch-and-price algorithm.

On the basis of the literature review, we can observe that there is no other approach for the vehicle scheduling problem taking into consideration concomitantly multiple depots, heterogeneous fleet, and time windows.

3 Proposed mathematical formulation

In this model, to which we refer as Multiple Depot Heterogeneous Fleet Vehicle Scheduling Problem with Time Windows (MDHFVSP-TW), we intend to obtain the optimal vehicle allocation and trip scheduling in order to minimize the global cost of mass transport system operation. Hereafter, the notation used for the formulation of the model will be presented.

It is important to note that several arcs (and nodes) from a given trip may be represented in a TSN and only one can be included in the final solution. This multiple arcs from a same trip are the result of the application of the time windows and, although they share the same origin and ending place, their starting and ending times have a subtle difference from the ones indicated in the timetable. For example, for a given time window of 1 min, a trip may have three representations (set of *Nta* and *Ntd* nodes and the correspondent trip arc) in the TSN: the original, formed by nodes (and the correspondent trip arc) with the same time as indicated in the timetable, a second where the starting and ending times of the nodes (and the correspondent trip arc) are reduced in one minute, and a third one, where 1 min is added to the starting and ending times of the nodes (and to the correspondent trip arc). The group of this arcs, containing the arc with the original times and the other arcs created due the application of time windows, were named *time windows variations*.

Sets

- D Set of depots;
- K Set of vehicles;
- $Arcs$ Set of arcs;
- $Atasks$ Subset of $Arcs$, including only the task arcs, $Atasks \subseteq Arcs$;
- $AtaskID$ Subsets of an $Atask$. Each set contains the time windows variations of a given arc. $AtaskID \subseteq Atasks$;
- N Set of nodes;
- Ndo Set of first nodes of each depot, $Ndo \subset N$;
- Ndd Set of last nodes of each depot, $Nde \subset N$;
- Cap Set of categories of vehicle capacity;
- Cat Set of vehicles that belong to the same category Cap, $cat \subseteq Cap$;

Parameters

- VC^w Cost increase factor according to the type of vehicle w;
- C_a Cost of arc a;
- Cap^c Passenger capacity of vehicle w;
- Dem_{id} Demand for an arc part of a given $AtaskID_{id}$ group;
- Ava_d^{cat} Number of vehicles available of a given set cat at depot d;
- Dem_{id} Demand for all $Atask$ of group id;

Decision variables

$$Z_a^w = \begin{cases} 1, & \text{if vehicle} w \text{is allocated to Arc } a; \\ 0, & \text{otherwise.} \end{cases}$$

MDHFVSP-TW model

$$\min \sum_{a \in Arcs} \sum_{w \in K} C_a \, VC^w \, Z_a^w \tag{1}$$

Subject to:

$$\sum_{(j,s)\,\in\, N} Z^{w}_{i,t;j,s} - \sum_{(j,s)\,\in\, N} Z^{w}_{j,s;i,t} = \begin{cases} 1, & \text{if } (i,t) \in Ndo; \\ -1, & \text{if } (i,t) \in Ndd; \\ 0, & \text{otherwise;} \end{cases} \quad \forall w \in K\ (i,t) \in N \tag{2}$$

$$\sum_{w\,\in\, K} \ \sum_{b\,\in\, AtaskID_{id}} Z^{w}_{b} = 1 \qquad \forall\, id \in AtaskID; \tag{3}$$

$$\sum_{(j,s)\,\in\,\delta^{+}_{(i,t)}} Z^{w}_{(i,t;j,s)} = \sum_{(m,n)\,\in\,\delta^{-}_{(i,o)}} Z^{w}_{(m,n;i,o)} \quad \forall i \in D,\ \ w \in K; \tag{4}$$

$$\sum_{w\,\in\, K} \ \sum_{b\,\in\, AtaskID_{id}} Z^{w}_{b}\ Cap^{w} \geq Dem_{id} \qquad \forall\ id \in AtaskID; \tag{5}$$

$$\sum_{w\,\in\, cat} Z^{w}_{i,t:j,s} \leq Ava^{c}_{i} \qquad \forall\, i \in D,\ \ (i,t) \in Ndo,\ \ c \in Cat; \tag{6}$$

$$Z^{k}_{a} \in \{0, 1\} \qquad \forall\, k \in K,\ a \in Arcs. \tag{7}$$

The objective function (1) aims to minimize the cost of the solution. The value of the solution for this model is obtained by the multiplication of cost C_a for a selected arc a times the incremental value factor for the selected type of vehicle VC^{w}. Flow balance constraint set (2) ensures the continuity of arcs allocated for a vehicle. Task covering constraint set (3) ensures that for each $AtaskID$ groups, only one task will be selected to be part of the solution and to be assigned to a particular vehicle. The constraint set (4) determines that every vehicle will end the journey at their start depot. The vehicle capacity constraint set (5) determines that the selected vehicle for a trip must have a passenger capacity equal to or greater than the demand for that trip. The fleet size constraint set (6) ensures that the number of vehicles of a given type allocated of a certain depot must be lower than or equal to the available number of vehicles in that depot. Finally, constraint set (7) determine the domains of the decision variables.

4 Proposed approaches

The Size-reduction (SR) approach is based on the idea that there is a small chance of a high-cost arc being included in a competitive solution. Therefore, in order to reduce the size of the problem and speed up its solution, a percentage of these arcs is eliminated before the beginning of the analysis. This heuristic was firstly proposed by Fanjul-Peyro and Ruiz [7], for the unrelated parallel machine scheduling problem. The SR algorithm does not guarantee that the optimal global solution is found for a given instance and may not present high-quality solutions in some cases. However, for some cases (as the one in this study), this method can offer a fast and straightforward implementation at a low computational cost.

In order to determine our limit acceptable arc cost, all the arcs are organized in crescent order by its cost, from the lower to the higher cost arc. Then the value for a parameter α is defined, ranging from 0 to 1. The product of α and the total number of arcs will be used to select the limiting cost. For example, if α is equal to 0.80 and a total of 100 arcs in the instance's time window, the cost of the 80th will be the maximum allowed cost for the arcs.

The decision to restrict the analysis to connection and waiting arcs has proven to not be an adequate criterion through the tests. The real percentage of eliminated arcs was around 2% for α equal to 0.75 and approximately 5% for α equal to 0.50, a reduction that could not effectively impact the solution times. Consequently, the approach was modified, allowing the algorithm also to eliminate pull-in and pull-out arcs. This adjustment significantly affected the maximum permitted value for α, due to the risks mentioned in the previous paragraph. However, the real percentage of fixed arcs were much closer to the value of α if compared to the initial approach.

The proposed methodology follows the main idea of the SR approach explained previously: reduce the size of the TSN to decrease the problem's size. However, reducing the size of the TSN, if not done correctly, can substantially lower the quality of the provided solution. The use of a model based on previously solved MDHFVSP-TW, with similar instance characteristics, aims to obtain a valid response while mitigating the loss of quality of the solution. A machine learning technique applied to reduce the size of a TSN was not found during the literature review. Our idea is to use a machine learning reduction (MLR) to predict from which arc a vehicle will be allocated based on the trip characteristics. This prediction would allow removing the non selected arcs, therefore, reducing the size of the TSN.

In this study, we adopt supervised learning. Due to the unprecedented application of machine learning algorithms for TSN reduction, we do not know which algorithms are the best fit for our datasets. Thus, five commonly-used algorithms are under consideration: Logistic Regression (LR); Linear Discriminant Analysis (LDA); k-Nearest Neighbors (KNN); Classification and Regression Trees (CART); Gaussian Naive Bayes (NB); and Support Vector Machines (SVM).

The databases for the MLR were created using previously solved MDHFVSP-TW instances. After the selection of the most appropriate algorithm for each group and database, the arc elimination begins. This process is divided into two phases: The first phase is the input of the instance, the TSN generation, and its data mining to generate a dataset for each type of trip; In the second phase, the selected algorithm is applied for each case, using the information from the database to predict which arc will likely be in the final solution, removing the unselected arcs.

In MDHFVSP-TW model for the problem in this study, the variable Z is a large size binary matrix shaped by lines representing each arc in the TSN and by columns representing each available vehicle. Despite the large size of this problem, the density of the matrix Z at the optimal solution is usually low. Supported by this characteristic, the LP-and-Fix approach utilizes the relaxed solution of variable Z to reduce the number of arcs that need to be analyzed for the final solution. This approach is based on Maes et al. [20] algorithm for the multilevel capacitated lotsizing problem, although some adjustments were made to better suit the needs of the problem under study.

The initial step is to relax the integrality of decision variables X and Z and solve the problem. The following step is to determine which elements will be fixed and, for that, a real-valued parameter β varying from 0 to 1 is established. Then, for each element of Z that is not in the relaxed solution, a random real number between 0.00 to 1.00 is chosen. If the value is smaller or equal to β, the solution value of this arc is set to 0, removing it from the TSN. Finally, the integrality of the variables X and Z is restored and the problem is once more solved. Although the probability of a virtual arc having a value equal to 0 is small, removing this type of arc could turn the integer model unfeasible. Therefore, no virtual arcs are fixed.

The proposed methodology is a combination of the previously explained SR method and the LP-and-Fix implementation. First, the SR is applied, eliminating a percentage of the most expensive arcs and therefore, the relaxed execution, fixation and integer execution are performed, taking advantage of the smaller TSN provided by the size-reduction. Algorithm 1 presents the combination of the SR and LP-and-Fix methodologies.

Algorithm 1: Size-Reduction plus LP-and-Fix

```
Step 1:(initialization) set parameters $\alpha$ and $\beta$, input the TSN,
time limit and load the model.
Step 2:(cost organization) organize all costs from the lowest
to the highest in a vector $v$.
Step 3:(limit cost value) the limit cost value is equal to the
value of $(\alpha \times v)$th element of $v$.
Step 4:(arcs elimination) For each $a \in Arcs$, $if$ (cost $a$ > limit cost
value) then, $\sum_{(w \in k)} z_a^k = 0$.
Step 5:(variable relaxation) relax the integrality of
variables $X$ and $Z$ to accept value between 0.00 and 1.00.
Step 6:(relaxed model solving) solve the relaxed model.
Step 7:(element fixation) analyze the solution of the relaxed
model. For each element whose solution is equal to 0, randomly
pick a number between 0.00 and 1.00. If the number is smaller
or equal to $\beta$, fix the value of the element equal to zero.
Step 8:(variable integrality) restore the integrality.
Step 9:(solution) solve the integer model considering
previously fixed variables.
```

4.1 Machine learning reduction plus LP-and-fix (MLR + LP-and-fix)

Despite its promising early results, the reduction provided by applying the MLR approach to only two trip types may not be significantly effective for large instances. Therefore, machine learning reduction was combined with the LP-and-Fix technique previously presented. This methodology is divided into two stages: in stage 1, the dataset for a type of trip is mined from the database and a suitable machine learning algorithm is defined. This process is repeated for each cluster of instances with the same characteristics and each trip type. Then, in stage 2, we implement the selected algorithm to reduce the size of the TSN and finally solve the problem using the LP-and-Fix approach.

Given the large number of possible combinations, only two groups of trips were selected for the application of the machine leaning reduction method. Those groups were named Case 0 and Case 1. Case 0 trips have no connection or awaiting arcs before and after the trip, the only arcs linked are pull-in and pull-out arcs. Case 1 tasks have only one connection before its beginning and no other linked arcs besides pull-in and pull-out arcs. Algorithm 2 expresses the combination of the proposed machine learning reduction and LP-and-Fix techniques.

Algorithm 2: Machine Learning Reduction plus LP-and-Fix

```
Step 1:(initialization) set parameter $\beta$. Input the TSN, input
case 0 model, input case 1 model, time limit and load the
model.
Step 2:(case 0 analysis) Sort out the tasks of TSN that fit
case 0 and trough case 0 model define which arcs must be
preserved and which must be deleted from the TSN.
Step 3:(case 1 analysis) Sort out the tasks of TSN that fit
case 1 and trough case 1 model define which arcs must be
preserved and which must be deleted from the TSN.
Step 4:(re-inclusion of arcs) analyze all group of tasks that
present the same starting or ending node and, if the are no
pull-in or pull-out arcs and if the number of connection arcs
is less than the number of tasks, re-include the pull-in or
pull-out arcs correspondent.
Step 5:(variable relaxation) relax the integrality of
variables $X$ and $Z$ to accept value between 0 and 1.
Step 6:(relaxed model solving) solve the model.
Step 7:(element fixation) analyze the solution of the relaxed
model. For each element whose solution is equal to 0, randomly
pick a number between 0.00 and 1.00. If the number is smaller
or equal to $\beta$, fix the value of the element equal to zero.
Step 8:(variable integrality) restore the integrality.
Step 9:(solution) solve the model.
```

5 Computational results

The computational experiments have the primary objective of comparing the performance of the developed solution approaches, using randomly generated instances.

We implemented the TSN generation in Python 3.2 as well as the MLR, which used the scikit-learn library. The LP-and-Fix and SR algorithms were implemented in C++, using IBM ILOG CPLEX version 12.6 as a standard optimization package in Concert technology. All experiments were carried out in an Intel Core CPU i5-6400, 2.70 GHz, 8 GB RAM. The following notation is used to indicate the implemented algorithms: (1) LP-and-Fix: variable relaxation and zero fix; (2) SR: size-reduction; (3) SR + LP-and-Fix: size-reduction plus variable relaxation and zero fix; (4) MLR: machine learning reduction; and (5) MLR + LP-and-Fix: machine learning reduction plus variable relaxation and zero fix.

5.1 Instances generation

The trip generation methodology proposed by Carpaneto et al. [3] has been widely used in MDVSP literature. However, a few modifications were made to adapt this methodology to generate MDHFVSP instances. These modifications include the implementation of demand function developed by Guedes et al. [10] and a way of determining the number of vehicles of each type for the depots.

We considered three sets of instances for the experiments. The first set is based on the real-world instance of guedes2015 and it contains instances of nine relief points, five depots and three vehicles types with capacities of 84, 95 and 140 passengers. The second and third sets both include instances 10 relief points and consider four vehicles types with capacities of 70, 80, 100 and 160 passengers and. The second set is composed of 4-depot instances, while the third contains 6-depot instances.

5.2 Parameter tuning

To test several values of α and β, we generated a set containing five instances of 200 trips, 10 relief points, and 4 depots. Due to the size of the instance, all executions were limited to 3600 s. Each methodology was tested separately, and although these values can vary according to the number of arcs in the TSN, these results gave us an insight into the best suitable values for larger instances.

For α, the values of 0.75, 0.80 and 0.85, 0.90 were tested. Low values of alpha result in lower arc cost limit and, therefore, in a greater number of arcs removed from the TSN. The solutions for α equal to 0.75 and 0.80 made the TSN unfeasible for this set of instances while values of α equal 0.85 and 0.90 produced good results with average deviations from the optimal solutions (gap) of 15.94% and 16.13% respectively.

The set was also tested for values of β ranging from 0.10 to 0.90, comparing the average solution gap to the average computational time required for each value of β. Based on these results and later tests carried on larger instances, the most suitable values for the SR + LP-and-Fix proposed methodology is 0.90 for α and 0.80 for β. For the MLR + LP-and-Fix, the value of β was also maintained in 0.80.

5.3 Selection of machine learning algorithms

Six literature most commonly found machine learning algorithms were tested, regarding their accuracy, to determine the best suitable algorithm for each dataset. Although large-sized datasets usually improve machine learning algorithms accuracy and are highly desirable, due to the high computational costs, each restrict dataset was limited to 10 previously solved instances. The general datasets are the collection of restrict datasets with the same characteristics of relief points, depots, and vehicle types. Therefore, the general database for set 1, the general database was generated using 30 solved instances and, for set 2, 40 instances were used.

5.4 Lower bound

The proposed lower bound is based on the Lagrangian relaxation methodology. The constraint sets (4) and (5) were relaxed to reduce the complexity of this problem. The choice of relaxing this set of constraints is based on experiences in early empirical tests when this restriction was not implemented. Although the solution was not so far from the complete model solution, the computational times were significantly smaller.

To maintain the feasibility after this relaxation, the set of constraints 5 also needed to be relaxed.

5.5 Results and discussion

The application of time windows increases extremely the problem size. Considering the NP-hard characteristic of the HFMDVSP, an ample time window could substantially increase the computational cost of a feasible solution. This study applied a time window of 1 min (forward or backward) for every trip, demonstrating the potential of this application while maintaining an acceptable computational cost. The value for factor γ was defined as equal to 0.3 for all instances.

The proposed LP-and-Fix approach is not an exact methodology. Therefore, the final solution provided for each instance may not always be the same. For approaches that present this characteristic, it is desirable to solve several times each instance in order to compare their solutions and have a better understanding of the approach accuracy. However, due to the computational time required for each execution, given the size of the instances and the complexity of the problem, the cost for executing the same instance several times was prohibitive. We compensate for the lack of repeated executions with a large set of instances, with a total of 155 executions for the SR + LP-and-Fix methodology and 70 executions for the MLR + LP-and-Fix approach. For every execution, a computational time limit of 10,800 s was established.

A summary of Gap comparison between pure model and SR + LP-and-Fix application can be seen in the Table 1. However, we offer a detailed presentation of these results in the link Supplementary material—SR + LP-and-Fix results. The results for all solution times are shown in seconds of CPU time. The column model detail the solution cost and computational time needed for each instance without the heuristic application. The SR + LP-and-Fix column shows the value of the provided solution using this method, the computational time required for the solution of the relaxation (ZLP time), as well as the time required to solve the problem after the fixation of arcs and restored integrality of variables X and Z (ZIP time). The solution gap column compares the deviation of the obtained results from both methodologies to the lower bound value (linear relaxation).

The model method refers to the execution of the model on CPLEX without heuristics and, as can be seen, smaller instances can be solved within the specified parameters in most cases. It should be noticed the values of these solutions presented a small gap relative to the objective value provided by CPLEX; therefore we can treat these solutions as near optimal. These values can be used to evaluate the quality of the relaxed solution and how far these solutions are from the optimal solution, providing a better understanding of the results of the proposed methodologies.

The results of the experiments indicate that the SR + LP-and-Fix approach is capable of providing good quality solutions within the specified computational time limit. For the first set, the value gap is lower than 15.5%, while for the second and third sets the provided solutions gap is lower than 30%. The comparison of solutions of instances of same trip sizes from different sets makes possible to assume that the variety of vehicle types plays a more critical role in the problem complexity than the

Table 1 Summary of gap comparison between pure model and SR + LP-and-Fix application

Number of depots	Number of relief points	Number of tasks	Gap	
			Average model solution	Average SR + LP-and-Fix (%)
5	9	250	10.59%	15.03
		375	–	15.77
		530	–	14.28
4	10	250	6.22%	15.65
		375	3.91%	14.35
		500	–	11.78
		625	–	11.71
6	10	250	–	13.88
		375	–	15.37
		500	–	14.77

number of depots, once the solution gap of the second set is closer to the third set than the first set.

We offer a detailed presentation of these results in the link Supplementary material—MLR+LP-and-Fix results. We present the results obtained for the execution of the two variants of the MLR + LP-and-Fix methodology and compare the provided solutions with the model and SR + LP-and-Fix solutions. The results for all solutions times are shown in seconds of CPU time. Table 2 presents a summary of Gap comparison between pure model, SR + LP-and-Fix application and the two proposed MLR + LP-and-Fix methodologies.

The MLR + LP-and-Fix methodology presented solutions with a gap close to those performed by the SR + LP-and-Fix approach for small instances. However, for larger instances, especially for the instances of the size of 625 trips from set 2, the solutions obtained showed a worsening in their quality. This difference between the two methods may indicate that for larger instances the number of trip cases considered must increase to make the MLR approach as efficient as SR.

It can be observed that the database heavily influences the algorithms predictions and, for these executions, results obtained with the SR + LP-and-Fix approach were used as the database. These solutions contain small bias that may be passed on to the prediction criterion of the algorithm, decreasing the quality of the solutions. Therefore, the efficiency of this methodology may be improved through the use of a database with optimal solutions.

6 Case study

We carried out computational experiments on a real-world problem, from the transit network of Santa Maria, Brazil, previously studied by Guedes and Borestein [10]

Table 2 Summary of gap comparison between pure model, SR + LP-and-fix and MLR + LP-and-fix methodologies

Number of depots	Number of relief points	Number of tasks	Gap			
			Average model solution	Average SR + LP-and-Fix (%)	Average MLR + LP-and-Fix (general) (%)	Average MLR + LP-and-Fix (restrict) (%)
5	9	250	11.01%	15.09	21.78	21.43
		375	–	13.75	20.13	19.45
		530	–	11.59	18.40	20.24
4	10	250	8.53%	17.95	26.15	25.03
		375	5.13%	15.85	22.65	24.13
		500	–	12.79	21.04	19.34
		625	–	5.14	29.92	26.05

and Guedes and Borestein [11]. This bus transport system is operated by a private consortium formed by five different companies known as ATU. This instance consists of 529 daily trips divided into 20 lines and 9 relief points. The consortium fleet is composed by three different vehicle types: type A, an articulated bus with capacity for 140 passengers; type B with a capacity of 95 passengers; and type C, a smaller vehicle with capacity for 84 passengers.

Since we had no access to the number of vehicles of each type in the depots, this parameter was estimated using the same methodology explained in Sect. 5.1. The VC^w parameter for each vehicle, based on our estimate of the costs, is 1.0, 1.11 and 1.45 for vehicles of capacity equal to 84.95 and 140 respectively.

Table 3 shows the impact of a 1 min time window application in the final solution. We solved the instance using de SR + LP-and-Fix methodology using the same values of α and β defined in Sect. 5.2. We choose this methodology since the model alone could not solve it, and the other proposed method would require additional computational cost due to the need for a database. Although the time window application caused an increase in the computational cost, what is expected due to the rise in the problem's complexity, there is a reduction of 3.66% in solution cost, showing the potential of this application, even for small time windows. It is important to point that, applying only the MDHFVSP-TW pure model, the computational time required to obtain the optimal solution for this instance, without the application of time windows, was 37,830.3 s. For the instance with application of 1 min time windows, the MDHFVSP-TW pure model required 76,930.7 s to provide the optimal solution.

The MLR database was composed of instances with the same characteristics of the real instances: nine relief points, five depots, and three vehicle types. For the MLR + LP-and-Fix restrict database, as presented in Supplementary material—MLR+LP-and-Fix results we use the first ten instances of 530 trips, for general MLR + LP-and-Fix database, we use ten first instances of 250 and the first ten instances of 375 trips.

The selection of the machine learning algorithms for this instance followed the same procedure described in Sect. 5.3. Table 4 presents the results obtained for the proposed methodologies. Comparing the heuristic methods, the SR + LP-and-Fix acquired the best solution value with a value gap of 2.99%. However, the SR + LP-and-Fix approach could not prove the optimally of the solution found within the 10,800 s limit computing time.

General and restrict MLR + LP-and-Fix also obtained good solutions with vale gaps of 4.34% and 3.52%. Also, both methods could provide their final solutions in less than a third of the limit computing time. However, this instance presented no case 0 trip during the machine learning reduction technique, therefore, the elimination of arcs based on algorithm predictions was only valid for case 1 trips for this instance.

7 Conclusions

This paper addresses the multiple depot vehicle scheduling problem with heterogeneous fleet and time windows. The complexity of this model required the use of heuristics that does not guarantee the optimally of the solution, however, allows

Table 3 Time windows effect in real-world instance

Time windows	Optimal solution	Model			SR + LP-and-Fix		
		Solution	Time	Solution gap	Solution	Total time	Solution gap
No	8109.66	Unknown	10, 800	–	8321.83	2521.41	2.62%
1 min	7784.08	Unknown	10, 800	–	8017.02	10,800	2.99%

Table 4 Real-world instance executions for proposed approaches

Optimal solution	Model		SR + LP-and-Fix		MLR + LP-and-Fix (general)		MLR + LP-and-Fix (restrict)	
	Solution	Time	Solution	Time	Solution	Time	Solution	Time
7784.08	Unknown	10,800	8017.02	10,800	8121.69	3080.95	8057.82	3274.83
Solution gap	–		2.99%		4.34%		3.52%	

obtaining a good quality solution within an acceptable computing cost. Variable fixing heuristics are proposed, combined with two TSN size reduction methodologies, each combination was capable of finding good solutions within acceptable computational costs the evaluated test instances.

Despite a large number of instances and parameters analyzed, this study presents limitations. Since the LP-and-Fix approach is not an exact methodology, all instances should have been executed more than once to reduce possible solutions deviations. However, because of the prohibitive computational cost, this was not carried out for both randomly generated instances and real-world instances. Additionally, since we used randomly generated instances of an NP-hard problem, the computational cost for obtaining the optimal solutions for all instances was also prohibitive, which did not allow entirely accurate analysis of the methods, limiting the comparison to solutions obtained with the relaxation of the integrality constraints. The impossibility of achieving the optimal solution of the instances also made it necessary to use solutions of the SR + LP-and-Fix methodology for the construction of the database, resulting in a possible decision bias.

The following ideas can be suggested for future enhancements of this work: (1) improvement of the lower bound, since the linear relaxation is not a robust approach; (2) comparison of the proposed variable fixing heuristics with column generation approaches; (3) integration with a crew scheduling problem, developing an integrated and robust approach, which deals with two major problems of the mass transportation system area; (4) improvement of the machine learning reduction, increasing the database size without increasing the associated computational cost and expanding the number of trips of different characteristics analyzed; (5) inclusion of vehicle acquisition costs in the objective costs; (6) consideration of the vehicle maintenance costs impact on the vehicle allocation.

Acknowledgements This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work was partially funded by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq)through Grants 404232/2016-7 and 303594/2018-7.

References

1. Banihashemi, M., Haghani, A.: Optimization model for large-scale bus transit scheduling problems. Transp. Res. Rec. **1733**, 23–30 (2000). https://doi.org/10.3141/1733-04
2. Bertossi, A.A., Carraresi, P., Gallo, G.: On some matching problems arising in vehicle scheduling models. Networks **17**(3), 271–281 (1987). https://doi.org/10.1002/net.3230170303
3. Carpaneto, G., Dell'amico, M., Fischetti, M., Toth, P.: A branch and bound algorithm for the multiple depot vehicle scheduling problem. Networks **19**(5), 531–548 (1989). https://doi.org/10.1002/net.3230190505
4. Ceder, A.: Urban transit scheduling: framework, review and examples. J. Urban Plan. Dev. **128**(4), 225–244 (2002)
5. Dell'Amico, M., Fischetti, M., Toth, P.: Heuristic algorithms for the multiple depot vehicle scheduling problem. Manag. Sci. **39**(1), 115–125 (1993)
6. Desaulniers, G., Lavigne, J., Soumis, F.: Multi-depot vehicle scheduling problems with time windows and waiting costs. Eur. J. Oper. Res. **111**(3), 479–494 (1998). https://doi.org/10.1016/S0377-2217(97)00363-9

7. Fanjul-Peyro, L., Ruiz, R.: Size-reduction heuristics for the unrelated parallel machines scheduling problem. Comput. Oper. Res. **38**(1), 301–309 (2011). https://doi.org/10.1016/j.cor.2010.05.005. (Project Management and Scheduling)
8. Forbes, M., Holt, J., Watts, A.: An exact algorithm for multiple depot bus scheduling. Eur. J. Oper. Res. **72**(1), 115–124 (1994). https://doi.org/10.1016/0377-2217(94)90334-4
9. Gintner, V., Kliewer, N., Suhl, L.: Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. OR Spectrum **27**(4), 507–523 (2005). https://doi.org/10.1007/s00291-005-0207-9
10. Guedes, P.C., Borenstein, D.: Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. Comput. Ind. Eng. **90**, 361–370 (2015). https://doi.org/10.1016/j.cie.2015.10.004
11. Guedes, P.C., Borenstein, D.: Real-time multi-depot vehicle type rescheduling problem. Transp. Res. B Methodol. **108**, 217–234 (2018). https://doi.org/10.1016/j.trb.2017.12.012
12. Guedes, P.C., Lopes, W.P., Rohde, L.R., Borenstein, D.: Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem. Optim. Lett. **10**(7), 1449–1461 (2016). https://doi.org/10.1007/s11590-015-0944-x
13. Hadjar, A., Marcotte, O., Soumis, F.: A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. Oper. Res. **54**(1), 130–149 (2006). https://doi.org/10.1287/opre.1050.0240
14. Hadjar, A., Soumis, F.: Dynamic window reduction for the multiple depot vehicle scheduling problem with time windows. Comput. Oper. Res. **36**(7), 2160–2172 (2009). https://doi.org/10.1016/j.cor.2008.08.010
15. Huisman, D., Freling, R., Wagelmans, A.P.M.: A robust solution approach to the dynamic vehicle scheduling problem. Transp. Sci. **38**(4), 447–458 (2004). https://doi.org/10.1287/trsc.1030.0069
16. Ibarra-Rojas, O.F.D., Giesen, R., Muñoz, J.C.: Urban transit scheduling: framework, review and examples. Transp. Res. **77**(Part B), 38–75 (2015)
17. Kliewer, N., Mellouli, T., Suhl, L.: A time-space network based exact optimization model for multi-depot bus scheduling. Eur. J. Oper. Res. **175**(3), 1616–1627 (2006). https://doi.org/10.1016/j.ejor.2005.02.030
18. Kulkarni, S., Krishnamoorthy, M., Ranade, A., Ernst, A.T., Patil, R.: A new formulation and a column generation-based heuristic for the multiple depot vehicle scheduling problem. Transp. Res. B Methodol. **118**, 457–487 (2018). https://doi.org/10.1016/j.trb.2018.11.007
19. Lobel, A.: Vehicle scheduling in public transit and Lagrangean pricing. Manag. Sci. **44**(12–part–1), 1637–1649 (1998). https://doi.org/10.1287/mnsc.44.12.1637
20. Maes, J., McClain, J.O., Wassenhove, L.N.V.: Multilevel capacitated lotsizing complexity and LP-based heuristics. Eur. J. Oper. Res. **53**(2), 131–148 (1991). https://doi.org/10.1016/0377-2217(91)90130-N
21. Oukil, A., Amor, H.B., Desrosiers, J., Gueddari, H.E.: Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. Comput. Oper. Res. **34**(3), 817–834 (2007). https://doi.org/10.1016/j.cor.2005.05.011. (Logistics of Health Care Management)
22. Pepin, A.S., Desaulniers, G., Hertz, A., Huisman, D.: A comparison of five heuristics for the multiple depot vehicle scheduling problem. J. Sched. **12**(1), 17 (2008). https://doi.org/10.1007/s10951-008-0072-x
23. Ribeiro, C.C., Soumis, F.: A column generation approach to the multiple-depot vehicle scheduling problem. Oper. Res. **42**(1), 41–52 (1994). https://doi.org/10.1287/opre.42.1.41
24. Xu, X., Ye, Z., Li, J., Wang, C.: Solving a large-scale multi-depot vehicle scheduling problem in urban bus systems. Math. Problems Eng. **2018**, 13 (2018). https://doi.org/10.1155/2018/4868906