Titre: Title:	A Machine Learning Preprocessor to Speed Up the Solution of a Bus Scheduling Problem with Controlled Trip Shifting
Auteur: Author:	Mozhgan Moeintaghavi
Date:	2023
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Moeintaghavi, M. (2023). A Machine Learning Preprocessor to Speed Up the Solution of a Bus Scheduling Problem with Controlled Trip Shifting [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10834/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10834/
Directeurs de recherche: Advisors:	Guy Desaulniers, & Quentin Cappart
Programme: Program:	Maîtrise recherche en génie industriel

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

A Machine Learning Preprocessor to Speed up the Solution of a Bus Scheduling Problem with Controlled Trip Shifting

MOZHGAN MOEINTAGHAVI

Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Génie industriel

Mars 2023

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

A Machine Learning Preprocessor to Speed up the Solution of a Bus Scheduling Problem with Controlled Trip Shifting

présenté par **Mozhgan MOEINTAGHAVI**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Antoine LEGRAIN, président

Guy DESAULNIERS, membre et directeur de recherche

Quentin CAPPART, membre et codirecteur de recherche

Louis-Martin ROUSSEAU, membre

ACKNOWLEDGEMENTS

First of all, I am grateful to my research director, Guy Desaulniers, for approving the supervision of my master's thesis at Polytechnique. Quentin Cappart, my co-director, deserves a special mention for his assistance and guidance. Their excellent responsiveness and insightful counsel had a significant impact on the success of this project. I also appreciate GIRO Inc for providing the financial funding for this project.

Additionally, I would like to express my gratitude to Louis-Martin Rousseau and Antoine Legrain for agreeing to participate on the jury for my master's degree.

RÉSUMÉ

À ce jour, l'optimisation s'est avérée être l'une des stratégies les plus efficaces pour accroître la volonté des gens d'utiliser les transports en commun, améliorant ainsi leurs avantages pour l'environnement, la société et l'économie en général. Pour ce faire, les entreprises de transport public ont augmenté leurs ressources afin d'optimiser leurs systèmes, par exemple en utilisant des techniques d'optimisation et l'intelligence artificielle, qui non seulement aident les entreprises à réduire les coûts, mais améliorent également l'expérience des passagers.

Pour s'assurer que chaque trajet est couvert au moindre coût possible, les systèmes de transport en commun gèrent la conception des lignes de bus, y compris le choix des horaires empruntés par chaque bus et l'emplacement de chaque gare routière. Une fois que l'horaire, qui précise la fréquence des départs des autobus et l'heure de début de chaque voyage d'autobus, a été déterminé, les autobus sont affectés aux voyages. Ensuite, en fonction de leur disponibilité, les chauffeurs sont jumelés aux bus.

De nombreux algorithmes exacts et heuristiques ont été proposés pour gérer ces étapes de planification, et certains auteurs ont même combiné certaines de ces étapes pour générer de meilleures solutions.

Le problème de planification de véhicules à plusieurs dépôts (MDVSP) résout un problème de planification de bus où les véhicules sont stockés dans deux dépôts ou plus. Dans ce mémoire, on étudie le MDVSP avec Controlled Trip Shifting (MDVSP-CTS), qui permet des ajustements contrôlés des horaires et envisage la possibilité de modifier légèrement l'heure de départ des trajets. D'autre part, le décalage des trajets doit être limité pour éviter un impact négatif sur le niveau de service aux passagers. Selon des travaux antérieurs effectués dans [1], le programme MDVSP-CTS est résolu à l'aide d'une heuristique à deux phases. L'algorithme heuristique à deux phases vise à atteindre des temps de traitement applicables à l'industrie du transport.

La première phase est résolue à l'aide d'une heuristique de génération de colonnes pour obtenir les horaires des véhicules, tandis que la deuxième phase est résolue à l'aide d'un solveur de programmation linéaire mixte en nombres entiers (MILP) pour trouver le meilleur horaire basé

sur les horaires de bus précédemment calculés dans la phase I.

Par rapport au modèle MDVSP, le modèle MDVSP-CTS est plus grand car il intègre des copies de voyage pour permettre le changement de voyage. Ici, une approche heuristique est utilisée pour réduire les temps de calcul, qui peuvent être encore accélérés en prenant en compte un sous-ensemble approprié de trajets qui peuvent être décalés.

Pour accélérer ce modèle d'optimisation, on utilise un modèle de prédiction pour nous aider à déterminer automatiquement quels trajets sont des candidats appropriés pour la sélection en tant que trajets avec décalage.

Pour prédire si un trajet doit être décalé ou non, on développe un modèle d'apprentissage automatique qui peut être formé à l'aide de données du monde réel et peut prédire les trajets qui pourraient avoir un changement d'horaire. Par conséquent, on est en mesure d'ajuster dynamiquement l'heure de départ des trajets de bus potentiels prédits par le modèle.

Pour résoudre le modèle d'apprentissage automatique, on divise aléatoirement les données en instances d'apprentissage, de validation et de test. Enfin, on utilise la précision du modèle comme critère pour évaluer le succès des prédictions.

Cette étude fournit et discute une comparaison de trois scénarios : autoriser le déplacement de tous les déplacements, autoriser uniquement les déplacements dont on prévoit qu'ils ont le potentiel d'être déplacés, et autoriser également uniquement les déplacements dont le déplacement est déterminé en fonction du premier scénario. Il est prometteur de constater que notre modèle d'apprentissage automatique a pu accélérer le modèle d'optimisation avec une réduction du temps de calcul d'un peu plus de 50% pour nos instances de test, tout en maintenance une qualité de solution similaire en moyenne.

Le modèle d'apprentissage automatique suggéré, dans lequel nous avons utilisé les trajets décalés prédits extraits comme sous-ensemble de trajets à décaler, a exécuté les algorithmes heuristiques plus rapidement lorsque nous avons utilisé un sous-ensemble des trajets réels à décaler. Ce résultat a été obtenu malgré le fait que la précision du modèle de prédiction était de 89%, alors que la précision des trajets décalés réels est de 100%.

Dans l'ensemble, on peut conclure qu'en améliorant davantage le modèle d'optimisation exis-

tant, cette méthode accélère l'algorithme d'optimisation des horaires de bus qui était auparavant proposé pour les systèmes de planification du transport en commun par bus.

ABSTRACT

To date, optimization has proven to be one of the most effective strategies for raising people's willingness to use public transportation, hence enhancing its advantages for the environment, society, and economy in general. In order to accomplish this, public transportation companies have raised their resources in order to optimize their systems, such as by utilizing optimization techniques and artificial intelligence, which not only helps companies decrease costs but also enhances the passenger experience.

To ensure that every trip is covered at the lowest possible cost, public transportation systems manage the design of bus lines, including choosing the schedules taken by each bus and the locations of each bus terminal. Once the timetable, which specifies the frequency of bus departures and the time each bus trip will begin, has been determined, the buses are assigned to the trips. Then, based on their availability, the drivers are paired with the buses.

Many exact and heuristic algorithms have been proposed to handle these planning steps, and some authors have even combined some of these steps to generate better solutions.

The Multiple Depot Vehicle Scheduling Problem (MDVSP) addresses a bus scheduling problem where the vehicles are stored in two or more depots. In this thesis, we study the MDVSP with Controlled Trip Shifting (MDVSP-CTS), which allows for controlled schedule adjustments, and considers the option of slightly changing the trip departure time. On the other hand, trip shifting needs to be kept under control to prevent a negative impact on the level of passenger service. According to previous work done in [1], the MDVSP-CTS program is solved using a two-phase heuristic. The two-phase heuristic algorithm aims to achieve processing times that can be applied to the transportation industry.

The first phase is solved using a column generation heuristic to obtain the vehicle schedules, while the second phase is solved using a Mixed Integer Linear Programming (MILP) solver to find the best timetable based on the previously-computed bus schedules in phase I.

As compared to the MDVSP model, the MDVSP-CTS model is larger since it incorporates trip copies to allow trip shifting. Here, a heuristic approach is used to reduce computation times, which can be further sped up by taking into account a suitable subset of trips that can be shifted.

To speed up this optimization model, we use a prediction model to assist us in automatically determining which trips are suitable candidates for selection as shifting trips.

To predict whether a trip should be shifted or not, we develop a machine learning model that can be trained using real-world data and can predict trips that might have a change in their

schedule to shift ahead or backward. Therefore, we are able to dynamically adjust the departure time for the potential bus trips predicted by the model.

To solve the machine learning model, we randomly split the data into train, validation and test instances. Finally, we use the model's accuracy as a criterion to assess how successful the predictions are.

This study provides and discusses a comparison of three scenarios : allowing all trips to be shifted, allowing only trips that are anticipated to have the potential to be shifted, and also allowing only trips that are determined to be shifted based on the first scenario. It is promising to observe that our machine learning model was able to speed up the optimization model with a computational time reduction of over 50%, while preserving the same solution quality on average.

The suggested machine learning model, in which we used the extracted predicted shifted trips as the subset of trips to be shifted, ran the heuristic algorithms faster when we used a subset of the actual trips to be shifted. This result was obtained despite the fact that the accuracy of the prediction model was 89%, whereas the accuracy of the actual shifted trips is 100%.

Overall, it can be concluded that by further enhancing the existing optimization model, this method speeds up the bus schedule optimization algorithm that was previously offered for bus transit planning systems.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

MDVSP	Multiple Depots Vehicle Scheduling Program
MDVSP-CTS	Multiple Depots Vehicle Scheduling Program with Controlled Trip Shifting
VCSP	Vehicle and Crew Scheduling Problem
VSP	Vehicle Scheduling Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows
IP	Integer Programming
ML	Machine Learning
DL	Deep Learning
CG	Column Generation
RMP	Restricted Master Problem
PP	Pricing Problem
LP	Linear Programming
NN	Neural Network
AI	Artificial Intelligence
BaP	Branch and Price
CPU	Central Processing Unit
dh	deadheading

LIST OF APPENDICES

CHAPITRE 1 INTRODUCTION

A persistently key aspect affecting both passengers and public transit operators is how to optimize the quality of public transportation services while lowering the overall expenses, besides improving the efficiency. Public transit operates at its maximum and has the highest level of passenger satisfaction when it is well-scheduled and optimized. This will not only enhance the demand but also, increase the revenue for transit operators, enhance passenger satisfaction, reduce the costs, and provide overall environmental, social, and economic advantages. Nowadays, Artificial Intelligence (AI) plays a significant role in the economic growth and development of almost every system in the nation, and public transit system is no exception [3]. For instance, it is known that there is a limited budget and time for the bus networks to operate everyday, and the objective is to maximize the performance while taking into account the constraints (such as cost limitations). So, public transit operators have been increasing their budget for the optimization of their system using the most cutting-edge techniques as a result of how technology has transformed everyone's lifestyles and daily activities. There are various operational planning phases in public transport (here, the bus transit system).

In the first phase, every bus company must adhere to the procedures used to design the bus lines, including determining the bus routes, bus stops, and the location of each bus terminal. The objective of defining different bus lines is to make sure that it could be a suitable service level for the population (passengers) in different areas of the city.

In the second phase, when we have the bus lines, then we determine the timetable. The timetable indicates for each bus line the frequency of bus departures (which may vary during the day) and at what time each bus trip will start. In order to meet the needs of bus network passengers, this phase entails choosing the number of departures on each line and their timings. Since the bus trips in each bus line depart at frequent basis, passengers can simply switch between various buses where there is a possible connection between the buses. Once more, this is connected to the service level taking into account the resources available. In these first two phases, the goal is to maximize the service quality to the passengers while taking into account the available resources. In the next two phases (vehicle scheduling and driver scheduling), the goal is to cover the trips at minimum cost.

In the third phase, which is called "vehicle scheduling", once we have determined the timetable, and we have bus trips, we assign the buses to those trips to ensure that all trips chosen based on the bus schedule in the previous phase are covered. So, in this phase, the route for each of these vehicles is specified.

Finally, in the last phase, the drivers are assigned to the buses. It is not necessarily true that each bus has a single driver assigned to it because, throughout the period of a day, a bus may have multiple drivers assigned to it (a bus might operate for 20 hours while a driver will work a maximum of, e.g., 10 hours). Furthermore, the drivers' workdays must adhere to the rules of the collective agreement, such as the maximum amount of driving time allowed prior to a break or the end of the workday, and also the daily maximum for changes.

To solve these operational planning phases, a wide variety of exact and heuristic algorithms have been devised. Some efforts have been made to combine some of the phases indicated above. As an example, consider Vehicle and Crew Scheduling Problem (VSCP) [4], and also Timetabling and Vehicle Scheduling Problem [5]. In this research, we are interested in combining the timetabling and vehicle scheduling problems into a single one called MDVSP with Controlled Trip Shifting (MDVSP-CTS). This integration results in a significant increase in computation time. We have therefore made an effort to lessen it by outlining other approaches.

As the aim of this master's thesis, we worked on defining an automated prediction model to speed up an optimization algorithm by predicting a subset of potential trips in which their departure time can be shifted. Some authors have previously designed an algorithm to solve this problem [1]. This optimization algorithm which has been designed and fully introduced for the public transit system by the work of [2] is solving a bus scheduling problem in which there exist two or more depots where the vehicles (buses) would be stored. This problem, which is called the Multiple Depot Vehicle Scheduling Problem (MDVSP), will be described in Section 1.1.
In this research, we consider a variant of the MDVSP called the MDVSP with Controlled Trip Shifting (MDVSP-CTS) that considers the possibility of slightly changing the departure time of the trips in order to reduce the operational cost. On the other hand, passenger service quality must not deteriorate too much, and therefore, trip shifting must be controlled. This problem was introduced in [2], where a two-phase heuristic approach is presented to solve this model. Here, Machine Learning (ML) is the approach used for trying to accelerate an existing solution method for the MDVSP-CTS. In fact, the goal is to speed up the algorithm that solves this optimization problem, and one method to do this is by reducing the subset of trips that could potentially be shifted. Hence, as an enhancing solution to this algorithm, we decided to propose a prediction model. As was explained above, the bus corporation distributes all of its vehicles among multiple "depots" to store them.

1.1 Overview of MDVSP and MDVSP-CTS

Finding a set of routes that cover all trips while being cost-effective is the basis of solving the Multi Depot Vehicle Scheduling Problem (MDVSP). Minimizing overall operating expenses is the objective function in the MDVSP optimization model (which includes the sum of bus fixed costs, bus travel, and waiting costs which is explained in detail in Section 3.1. Additionally, the constraints ensure that only one vehicle is allowed for each trip, that vehicle availability per depot is respected, and that each bus route must begin and end at the same depot. The key to this problem is to create each vehicle's schedule so that it covers all desired bus trips while reducing the expenses of the transportation system. Expenses for each bus used include both fixed and variable costs (travel and waiting costs). Due to the policy of storing the vehicles in various depots, MDVSP-CTS is particularly challenging to solve (class of NP-hard problems) [6].

As previously investigated in [1], MDVSP with Controlled Trip Shifting takes into account changing the start times of selected bus trips in order to minimize the operational costs. Here, a "Bus schedule" (or "$route$") is a sequence of bus trips performed by a bus during a day. A bus performs a "$Deadhead$" when it is moving between two trips without any passengers on board. It is anticipated that as a result, the transportation system will require fewer buses and spend less on deadheading. Also, it is marked as controlled shifting since it takes into account some restrictions to only adjust the trip timetable in cases where we do not impose undesirable modifications to the trip timetable, which affects passenger service quality. In fact, by proposing heuristic and exact algorithms, and also the column generation method, MDVSP-CTS optimization model aims to minimize the operational costs without affecting the quality level of the bus schedules and passenger satisfaction.
In order to prevent the service quality from deteriorating, we aim to maintain constant headways (the time intervals between two subsequent departures on the same line). As a result of changing the trip departure timings, it is essential that the passengers would be satisfied with the modified trip schedule.

1.2 Motivation and research objectives

As shown in [1], MDVSP-CTS can yield much better solutions than MDVSP, reducing the number of required buses and the operational costs. However, a two-phase heuristic was developed because it is difficult to solve MDVSP-CTS compared to the classic MDVSP. So, the main goal of this project is to reduce the computational times. To do so, a machine learning model is designed to learn and therefore, predict which trips have a high potential for being shifted and consider only those trips for possible shifting in the column generation algorithm.

In light of this, we determine all of the features that may be utilized to represent each bus trip. The training set includes generated data as well as a variety of bus trips we have taken from synthetic data provided to us with a generator [7], which in turn was developed using actual data provided by Giro Inc., our industrial partner. In terms of the output, we establish a binary variable for each trip that indicates whether or not the trip should be shifted. We create a model capable of managing a set of trips as input, train the model using supervised learning, and use machine learning to forecast a subset of bus trips in which trip shifting is taken into account. We have used Neural networks as our prediction model in this regard.

Finally, the MDVSP-CTS optimization model is run by the Gencol solver proposed in [1] using predicted data after the output of the machine learning model has been extracted, and the outcomes are then examined. In fact, by utilizing predictions from machine learning, we aim to enhance the optimization algorithm developed in [1]. To achieve this objective, we were fortunate to work in collaboration with Giro Inc, as our financial supporter and industrial partner in this project.

1.3 Thesis structure

In this thesis, a new approach to optimization is introduced to the work of Desfontaines and Desaulniers [1], in order to speed up the algorithm proposed for MDVSP-CTS. It is divided into six chapters. A review of the relevant literature and the most recent research on this topic are included in Chapter 2. Chapter 3 then provides more specifics on the problem statement and mathematical formulations. Chapter 4 describes the research methodologies for optimization (Column Generation and Heuristics) as well as machine learning, and Chapter 5 presents the experimental results. The conclusion is outlined in the last chapter (Chapter 6), and a list of the recommendations that result from this work is supplied at the end of Chapter 6.

CHAPITRE 2 LITERATURE REVIEW

Here, we review the most recent literature on the topics related to this research. This section is divided into two parts: In the first part, we proceed by outlining some of the prior research done on MDVSP and MDVSP-CTS. In the second part, we examine the earlier studies on machine learning in optimization.

2.1 Multi Depot Vehicle Scheduling Problem (MDVSP)

In the MDVSP, the companies' vehicles are distributed through multiple depots [1]. In other words, MDVSP, which is NP-hard, is referred to as the VSP (Vehicle Scheduling Problem), in which the bus service is split among two or more depots. [6] and [8] provide a thorough analysis of Vehicle Scheduling Problem (VSP) with one or more depots. The following two surveys have both been thoroughly examined for the existing approaches on MDVSP: [9], and [10].

Since there are frequently multiple solutions with the same cost for the MDVSP, this can significantly slow down the resolution process. In order to lessen degeneracy, [11] apply a stabilisation approach to MDVSP. This stabilization is based on the control of the dual costs suggested in [12].

To model MDVSP, several mathematical programming techniques have been developed that are mostly based on Integer Programming (IP). The MDVSP is given a new formulation as a set partitioning problem with side restrictions, whose continuous relaxation is amenable to being solved via column generation (CG). See [6] and [13]. Some articles such as [14] and [15] have expanded the scope of the method of [13].

In [14], the authors suggest a branch-and-bound method that combines cutting planes, variable fixing, and column generation to solve MDVSP. In addition, taking into account the optimization of formulation of a multi-commodity network flow, or MDVSP, the authors of [15] explore various approaches to solve MDVSP within a branch-and-cut framework. They compare the cutting plane separation sub-algorithms that were presented in [14]. Hence, a novel criterion for fixing some variables in the formulation of the multi-commodity flow was investigated in this work. Finally, the authors conclude by highlighting the computational experiments that consider both the advantages and disadvantages of the approach. Similar to this, some recent works have attempted to incorporate the creation of bus lines or the assignment of passengers to MDVSP. In fact, a strategy for simultaneously determining departure schedules, passenger assignments, and vehicle schedules is suggested by [16]. However, since only round trips on the same line are permitted, bus schedules are extremely limited, and consequently simple. The theoretical work

of creation and classification of potential meatheuristics to concurrently solve the creation of bus lines, timetables, and bus schedules is completed by [17].

By using a column generation algorithm to solve the linear relaxation, a different approach was developed by [18]. The author's approach is based on a multi-flow formulation. Here, the generated columns match the values of the flow on the connection arcs. The author then employs two Lagrangian relaxations of the MDVSP along with the traditional evaluation of the reduced cost of arcs to generate the variables. Most of the time, it produces integer solutions, and in the remaining cases, a rounding technique allows for the formation of an optimal or nearly optimal integer solution [18]. Additionally, many heuristics have been developed in order to speed up the solution to the problems with a larger data, and satisfy the demands of bus companies. In [19], five heuristics for the various MDVSP solution techniques are described and investigated. The first three employ techniques for integer linear programming, while the remaining two methods rely on local search metaheuristics. While using heuristic approaches, in order to increase the quantity and quality of connections, [20] attempted to solve an MDVSP while slightly modifying the schedules. In order to reconstruct the solution, a large neighborhood search and a heuristic method were developed. In [5], a simulated-annealing matheuristic is used with many vehicle type options per trip to lower the cost of bus schedules.

Based on [21], it is suggested that the MDVSP with time shifting can be solved using both exact and heuristic methods. The shifted trips are represented in this study as copies of the original bus trips using a space-time network. Only copies that allow for a new connection are kept in this case. In addition, the multi-flow model is solved using an integer programming solver. The process is then sped up by suggesting a heuristic strategy for selecting the trips that are potential to be shifted. By penalizing the copy arcs, authors try to stick as close to the original schedule as possible.

To solve MTVSP-CTS, based on [1], the problem is first described using a mixed integer programming model, with the objective function being to minimize the operating costs while maximizing the overall timetable quality. In the second step, a two-phase matheuristic approach is presented to solve the MDVSP-CTS problem. This approach uses a column generation heuristic in phase I, and a mixed integer programming in phase II. In the end, the MDVSP-CTS is solved for the synthetic data generated by the simulator, which is in turn based on the actual data from GIRO Inc., a leading provider of optimization software for public transit organizations. To solve the optimization model, a two-phase heuristic is proposed, and the results show a significant reduction in the number of necessary vehicles as well as a minor change in the timetable as compared to the classical MDVSP [1]. According to the tests done on cases with 300 or 500 bus trips, the bus transport company can save up to three vehicles (buses) with this approach. Also,

a real-world problem requiring more than 1000 trips was solved in one hour, which resulted in a decrease in both cost and time [1]. In light of this, the MDVSP-CTS is a practical problem that can be used in transportation systems.
As was noted in [1], there are some ideas to advance this work. It was indicated that the phase I, column generation process needs to be sped up in the first step, thus we chose to investigate the concept of using machine learning predictions on the MDVSP-CTS optimization algorithm.

2.2 Machine learning in operations research

Since the early stages of the creation of digital computers in the late 1950s, researchers have been working to create intelligent machines that can match and possibly even surpass humans in reasoning and making intelligent decisions [22]. The objective is to make it possible for the machines to learn from the past, and solve challenging problems under circumstances that are unlike previous observations. The prediction process in Machine Learning (ML) is carried out operationally, employing data-derived information and adhering to predetermined criteria; Optimization is undeniably one of its fundamental components [23].

Both Machine learning and Operations research, two growing sub-fields of Artificial Intelligence, have a wide range of applications in computer science. Unlike the latter, which seek the best alternative or solution to a particular problem, the former's strategies strive to learn information from data or experience. ML and Optimization are frequently hybridized, interacting with each other and themselves, in order to be employed automatically and effectively aligning with the real goal of artificial intelligence [24]. The various forms that optimization and machine learning interact are fully described in this work.

Computers can learn through machine learning, and as a result, automatically improve. Being widely used in a variety of sectors, ML is the foundation of artificial intelligence and data science, which integrates computer science and statistics [25]. Based on [26], as a result of the shortage of resources, inventive techniques like machine learning that save time, energy, and resources are growing more and more in demand.

Based on [27], three dimensions can be used to examine the connection between operations research and machine learning: (a) applying it to management science problems, (b) using it to solve optimization problems included in the management science problems, and (c) formulating machine learning problems as optimization problems.

Using machine learning to solve complex optimization problems, particularly NP-hard integer constrained optimization, is a significant area of research at the intersection of machine learning and operations research [28], [29], [30], [23], and [31].

Machine learning models are developed in this field to supplement current methods that take advantage of Combinatorial Optimization (CO) via structure detection, branching, and heuristics [27]. For instance, in [32], machine learning applications to the field of CO, specifically the Travelling Salesman Problem (TSP), are discussed. The various ML approaches used to solve the TSP are examined, debated, and their advantages and disadvantages are explored in this work.

In [31], a broadly applicable solution is developed for speeding up the computation time of the Branch-and-Price (BaP) algorithm, which is a quite robust and exact approach for solving complex combinatorial problems. According to the case studies in this paper, the pricing problem solution algorithm takes more than 90% of the overall computing time. In fact, the repetitive nature of the pricing problem requires it to re-solve the identical problem with only the input dual prices being different. The authors of this work have demonstrated that the solution space of pricing problems can be shortened in subsequent iterations by making use of the learning from earlier iterations of the pricing problem. To obtain this result, an online machine learning method is developed. So, the main contribution offered in [31] is its novel application of a machine learning technique to speed up the time it takes to solve pricing problems.

In addition, according to [23], regardless of the application domain in which optimization algorithms are utilised, the use of ML approaches in algorithmic design is also one of the topics that receives the most searches in the field of mathematical optimization. In this paper, the authors highlighted how machine learning improved the approaches of mathematical programming towards solution. Additionally, in [33], the advantages of learning and heuristics are integrated, and potential sub-problems are found to enhance upon, significantly reducing the computation time necessary to solve the model. A competitive sub-solver on sub-problems can generate good solution quality without suffering the high computational costs related to running the sub-solver on large problem instances. In this study, a learning-based system is provided that learns which sub-problems to allocate to a sub-solver while solving large VRPs. The authors state that the suggested technique speeds up competing VRP solvers on problems up to 3000 in size. In comparison to non-learning selection procedures, as well as observing a 1.5x to 2x speedup compared to non-learning selection procedures.

In the article [34], a ML-based pricing heuristic is introduced with the idea of accelerating the Column Generation (CG) approach for the problems. This technique is not only employed for vehicle and crew scheduling problem (VCSP) and Vehicle Routing Problem with Time Windows (VRPTW), but it is also applicable to a broad variety of problems. In fact, it entails reducing the network size by only preserving the most promising arcs that have a strong likelihood of being part of proper columns. Using supervised learning, the ML model is trained using the data gathered from prior runs. The authors constructed a new, scaled-down network that is, on

average, 15% to 25% the size of the original network [35] and [34].

The development of intelligent public transportation networks is a crucial step in the creation of sustainable cities. Understanding the origin-destination and travel behaviors of passengers is essential for improving service levels and encouraging people to use buses. According to [36], the stops of bus passengers can be predicted using machine learning, utilizing data from the passengers' smart cards. This method is based on the gradient boosting decision tree algorithm. A better understanding of the predicted passenger flows is essential since it will make the bus system operate more sustainably and efficiently in the long run. The bus system can actually be optimized using this approach.

Vehicle routing optimization is a crucial component of intelligent transportation systems and a popular area of study in many nations [37]. According to researchers, artificial intelligence and machine learning play a major role in solution of the vehicle routing problem (VRP). In addition to helping people realise the unity of resources, the environment, and efficiency, research on vehicle routing optimization also supports the growth of the logistics sector and the orderly development of the social economy. These social problems include energy shortages, traffic problems, and air pollution. It is increasingly utilized in processes like scheduling, resource allocation, decision-making, and optimization [37]. Machine learning has recently expanded at an enormous speed, attracting many researchers. One of the fundamental elements of machine learning is optimization. The majority of machine learning algorithms basically work by creating an optimization model and using the provided data to learn the parameters of the objective function [38]. In [39], the authors have examined how machine learning can be used to create partially learnt Combinatorial Optimization (CO) algorithms, being determined that a new era is just getting started in the field of operations research. In [30], a theoretical framework for analyzing a decision-making process is provided. A machine learning approach is proposed for modeling the likelihood that a heuristic will be successful, and useful rules are developed to use the ML models to dynamically choose whether to run a heuristic at each node of the search tree. The presented method in this article experimentally increases the primal performance of a state-of-the-art Mixed Integer Programming (MIP) solver by up to 6% and up to 60%, respectively.

CHAPITRE 3 PROBLEM STATEMENT AND FORMULATION

In this chapter, we first discuss the problem statement of MDVSP-CTS in Section 3.1. In Section 3.2, the model is described in two parts. In the first part, the MDVSP with trip shifting mathematical model is presented in Subsection 3.2.1. In the second part, the MDVSP with controlled trip shifting model is provided in Subsection 3.2.2. Finally, the underlying graph for MDVSP-CTS is described in Subsection 3.2.3 for a better understanding of the network structure.

3.1 Problem statement of the MDVSP-CTS

The MDVSP-CTS is about bus scheduling, which computes a set of bus schedules for a given day and a set of available buses (possibly assigned to different depots) in a way that the operational costs are minimized. The "operational costs" comprise a fixed cost for each bus operated, connection costs for subsequent bus trips, as well as the cost of transportation from the depot to the start and end locations of each bus schedule (travel and waiting costs). In comparison to the classical MDVSP (without trip shifting), the computation time increases significantly in order to be practical. In addition, MDVSP with Controlled Trip Shifting considers altering the start times of specific trips in order to reduce the operational costs. It is designated as controlled shifting because it takes into consideration some constraints to only adjust the trip timetable in situations when we do not impose unfavourable changes to the trip timetable, which would negatively affect the passenger satisfaction. In fact, it is crucial that the passengers are satisfied with the modified schedule since the trip departure times may have shifted. The idea behind this research is to use machine learning to speed up the present heuristic MDVSP-CTS algorithm because it takes longer to compute comparing with the classical MDVSP. The following paragraphs provide the fundamental definitions used in developing the MDVSP-CTS optimization model:

"Bus scheduling with trip shifting" is similar to bus scheduling except that each trip's timetable is allowed to be slightly changed. For instance, a trip which is pre-scheduled to leave at 8 am, might depart at 7:58 am, 8 am, or 8:02 am instead. It is crucial to note that we only have a limited set of discrete alternatives for this modification. "tmax" denotes the allowance of the change in each trip's start time. In this research, we put a two-minute maximum allowance for change in the schedule of each bus trip. So, each trip may be shifted up to 2 minutes because the value of $tmax$ is set to 2. "Headway" is the duration between two consecutive trips on the same bus line (for instance, if there are consecutive trips on a line that start at 8am and 8h15am, there will be a 15-minute headway between these trips). The headways are often stable during specific hours

of the day. For instance, the headway would be 15 minutes between 9:30am and 3:30pm, and 5 minutes between 3:30pm and 6:30pm. There is a headway before and after each particular trip, unless it is the first or last bus in the line. A "penalty" must be considered when modifying a trip's departure time. This is due to the fact that some passengers might miss some possible bus connections, and therefore, we would face a "missed connection". Also, if the initial headway is not maintained successfully, we must consider a penalty, which will be fully described in Subsection 3.2.2. A "deadheading" time is the duration between the "end of the bus line" to the "beginning of another bus line" (changing terminals or changing bus lines).

3.2 MDVSP-CTS mathematical formulation

Based on [2], the proposed model for MDVSP-CTS is presented in two parts. MDVSP with trip shifting is considered in the first part, and its development with controlled trip shifting is considered in the second part. For convenience, Appendix A contains all notations used in defining all models. The notations are explained in the paragraphs that follow.

We consider m as the number of bus trips to be covered. The set of trips is denoted as $V=\{v_1, v_2, \ldots, v_m\}$. A trip v starts at departure time t_v. The time at which the trip arrives at its end location is indicated as t_e^v. A set of trips v that each bus can cover, departing from and returning to the same depot d_p is called as a "Bus schedule". We have set a value of p to display the number of depots, and the set of depots is denoted: $D=\{d_1, \ldots, d_p\}$. The capacity of each depot is listed as $b_1, b_2, \ldots, b_p$. The capacity of the depot denotes the number of available buses in each depot.

In the first part, we only consider modifying the trip departure times. To do so, we create a *copy* k_v^i of v for each possible start time t_v^i of v. K_v represents a set of copies of trip v, along with the original trip. $K_v = \{k_v^1, \ldots, k_v^{|K_v|}\}$ with $t_v^i < t_v^{i+1}$ for $i \in \{1, \ldots, |K_v|-1\}$. In the second step, we model the MDVSP with controlled trip shifting.

Each use of a bus has a fixed cost of c_f, besides a variable c_a which shows the cost for each minute spent waiting at the bus station, and a cost c_v for each minute spent on an empty trip (It can be a deadheading, or between the depots and terminals). There is no cost for the bus waiting at the depots. Also, all vehicles in the depots are identical.

A *pull-out* trip is the first trip on a schedule; the bus leaves the depot and drives to the starting point of the first trip with no passengers. When the vehicle returns to its depot, a schedule symmetrically ends with a *pull-in* trip. For each depot $d \in D$, S^d denotes the set of all feasible schedules for the vehicles, and $S = \bigcup_{d \in D} S^d$. For each depot $d \in D$ and schedule $s \in S^d$, the parameter c_s denotes the cost of each schedule s (including the fixed cost for the vehicle.)

The MDVSP aims to find a set of bus schedules that cover each trip v exactly once, and take into

consideration the capacity of each depot. Also, finding a least-cost set of vehicle schedules that respects vehicle availability per depot $d \in D$ and covers exactly one copy of each trip $v \in V$ is the goal of the MDVSP-CTS, which will be fully described in Subsection 3.2.2. V^d is introduced as the set of trips that can be assigned to the buses at depot $d \in D$ since there may be some compatibility restrictions between the vehicles in a depot and the trips.

We define a binary parameter a_{vs}, $v \in V$, $s \in S$, that is equal to 1 if and only if schedule s covers trip v. It is assumed that a schedule cannot contain two or more copies of the same trip since the start times of the copies of the same trip would be relatively close. In addition, we define another binary variable y_s that would take a value of 1 if and only if schedule s is selected. Every pair of sequential trips v and v' covered by copies $k_v^i \in K_v$ and $k_{v'}^j \in K_{v'}$ have a *connection* with each other in this pattern. A vehicle may make this connection if it can travel the distance v starting at time t_v^i before finishing v' starting at time $t_{v'}^j$.

3.2.1 MDVSP with trip shifting

The MDVSP with trip shifting can be expressed as the following set partitioning model:

$$\min \quad \sum_{s \in S} c_s y_s \tag{3.1}$$

$$\text{s.t.} \quad \sum_{s \in S} a_{vs} y_s = 1, \qquad \forall v \in \mathcal{V} \tag{3.2}$$

$$\sum_{s \in S^d} y_s \leq b_d, \qquad \forall d \in \mathcal{D} \tag{3.3}$$

$$y_s \in \{0,1\}, \qquad \forall s \in S \tag{3.4}$$

The objective function (3.1) is to minimize the overall operational costs. Constraints (3.2) impose that each trip is covered by exactly one vehicle. In addition, constraints (3.3) restrict the number of vehicles used by the number of vehicles available at each depot. Finally, the constraints (3.4) show the integrality of the binary variable y_s since this is a Zero-one integer programming (or binary integer programming) problem [40]. The definitions of S^d and a_{vs} take into account for the fact that multiple copies are allowed for each trip $v \in V$. The model (3.1)-(3.4) is similar to the set partitioning formulation suggested by [13] for the classical MDVSP (without trip shifting). It is frequently challenging to solve the IP model (3.1)-(3.4) since the size of S increases exponentially when we have a larger number of trips as m has been defined as the number of trips. Instead of complete generation of set S, and to dynamically generate a subset of schedules, we employ a column generation method, which is discussed in Subsection 4.1.2.

3.2.2 MDVSP with controlled trip shifting

The quality level of the original timetable should not be negatively affected by the changes made to the trip departure times, even if there is a little adjustment and the alternative trip departure times are quite close to the initial ones. So, in order to regulate the quality of timetables, we take three factors into consideration.

The first is the sum of the absolute differences between the original trip start time and the new trip start time for all bus trips. This criterion seeks to restrict the number of changes made to the original schedule. So, for each $v \in V$, we therefore introduce two non-negative variables q_v^+ and q_v^-, which stand for the amount of time that trip v is shifted forward and backward, respectively. Also, in order to regulate the quality, we add an integer variable x_v that represents the amount of time by which the start time of trip v is shifted for each trip $v \in V$. This variable can be either 0, positive, or negative, and is defined as below:

$$x_v = q_v^+ - q_v^-, \qquad \forall v \in V \tag{3.5}$$

So, in order to maintain the original schedule for each trip v, the variables q_v^+ and q_v^- must take the value 0. Therefore, these variables face a unit penalty cost of α^T in the objective function.

The variables x_v, $v \in V$ are constrained by the schedules that are chosen in the solution. We define e_{vs}^k as a binary parameter (for each schedule $s \in S$, and for each trip $v \in V$) which takes the value 1 if schedule s enters the sub-network of trip v by copy k, and 0 otherwise. Similarly, o_{vs}^k is a binary parameter that takes the value 1 if schedule s exits the sub-network of trip v by copy k, and 0 otherwise.

We restrict the value of x_v according to the first and the last copy node used to enter and leave the sub-network representing trip v in the schedule s that is covering this trip. Therefore, Constraints 3.6 can be added to the optimization model as a generalized example for imposing the variables x_v.

$$\sum_{k \in K_v} \sum_{s \in S} (t_v^k - t_v) e_{vs}^k y_s \leq x_v \leq \sum_{k \in K_v} \sum_{s \in S} (t_v^k - t_v) o_{vs}^k y_s, \qquad \forall v \in V \tag{3.6}$$

The second criterion is to maintain a steady headway for each bus line. The set H is defined as all pairs of trips for which the initial headway should be maintained. To do so, in order to compute the positive and negative deviations from the initial headway between v and v', two non-negative variables $h_{vv'}^+$ and $h_{vv'}^-$ are defined.

$$x_{v'} - x_v = h^+_{vv'} - h^-_{vv'}, \qquad \forall (v, v') \in H \tag{3.7}$$

Although maintaining the initial headway is not required, failure to do so will result in penalties for the trips v and v'. To keep the initial headway, $h^+_{vv'}$ and $h^-_{vv'}$ must take value 0. The variables $h^+_{vv'}$ and $h^-_{vv'}$ in the objective function have a positive coefficient α^H to maintain the initial headways as much as possible.

Finally, as the third criterion, because some targeted trips are intended to facilitate specific connections between specific bus lines, we take into consideration the quality maintenance of these targeted passenger connections. Here, P is introduced to represent the set of pairs of trips for which we want to guarantee a satisfactory passenger connection at the spots where the trips overlap. Let $(v, v') \in P$ represent such a pair, and we assume that the ideal relative shifting $x'_v - x_v$ between the start timings of these trips equals a pre-determined value $\tau_{vv'}$, resulting in an ideal passenger connection. Hence, two non-negative variables $r^+_{vv'}$ and $r^-_{vv'}$ are introduced to compute the positive and negative deviations from the ideal shifting value $\tau_{vv'}$.

$$x_{v'} - x_v - \tau_{vv'} = r^+_{vv'} - r^-_{vv'}, \qquad \forall (v, v') \in P \tag{3.8}$$

A connection between trips v and v' is considered as ideal if $r^+_{vv'} = r^-_{vv'} = 0$. Even if there is a slight difference between $x_v - x_{v'}$ and $\tau_{vv'}$, the connection could still be considered possible but less convenient. On the other hand, if $x_{v'} - x_v$ is too far from $\tau_{vv'}$, the connection would be inconvenient. (It is either considered a missed connection or a lengthy waiting time is required.)

A penalty function $f^P_{vv'}$ is defined to model this for the pair of trips $(v, v') \in P$. The penalty for (v, v') is then given by $\alpha^P f^P_{vv'}(x_{v'} - x_v - \tau_{vv'})$. Here, α^P is a non-negative weight, which allows the modification of the passenger connection penalty costs. For the function $f^P_{vv'}$, the range of possible values of $x_{v'} - x_v - \tau_{vv'}$ is the interval $[-U^-_{vv'}, U^+_{vv'}]$. On the other hand, $[-u^-_{vv'}, u^+_{vv'}]$ is the range of values for which the connection is considered convenient. A penalty $\alpha^P \beta_{vv'}$ is thus introduced for a connection that is considered as inconvenient. This happens when $x_{v'} - x_v - \tau_{vv'}$ is outside the interval $[-u^-_{vv'}, u^+_{vv'}]$, while a unit penalty $\alpha^P \gamma^-_{vv'}$ is paid for a negative deviation in the interval $[-u^-_{vv'}, u^+_{vv'}]$. Similarly, a unit penalty $\alpha^P \gamma^+_{vv'}$ is defined for a positive deviation. Moreover, we introduce two binary variables $z^+_{vv'}$ and $z^-_{vv'}$ to linearize the function $f^P_{vv'}(x_{v'} - x_v - \tau_{vv'})$ for a connection by a pair of trips $(v, v') \in P$. These binary variables are equal to 1 if there exists an inconvenient connection because $x_{v'} - x_v - \tau_{vv'} \in [-U^-_{vv'}, -u^-_{vv'}[$, or if $x_{v'} - x_v - \tau_{vv'} \in [u^+_{vv'}, U^+_{vv'}[$, respectively. According to [2], the definition of the function $f^P_{vv'}(x_{v'} - x_v - \tau_{vv'})$ is clearly illustrated using Figure 3.1.

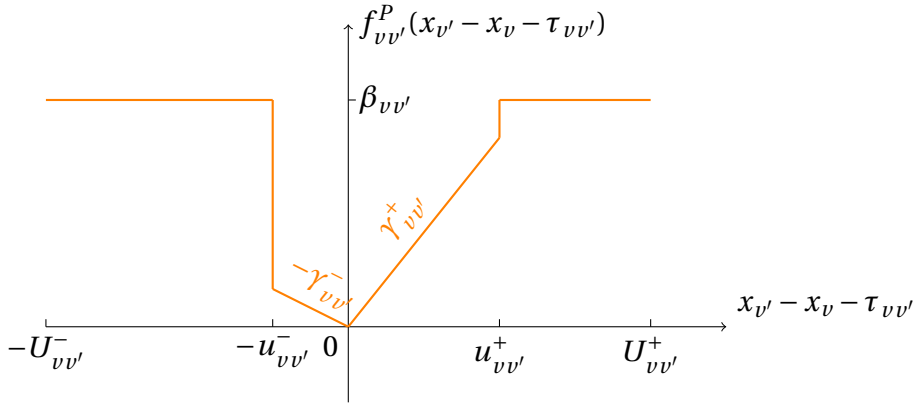

Figure 3.1 Definition of function $f^P_{vv'}(x_{v'} - x_v - \tau_{vv'})$

Moreover, non-negative variables $w^+_{vv'}$ and $w^-_{vv'}$ are introduced to compute the positive and negative deviations in $[-u^-_{vv'}, u^+_{vv'}]$, which results in a linear penalty. The time step (here, tmax), which refers to the maximum amount of shifting in time that can be applied to each trip's departure time, is the upper bound of the amount of start time adjustment that can be made.

Therefore, we can formulate the MDVSP-CTS as the following IP problem:

$$\min \quad \sum_{s\in S} c_s y_s + \alpha^T \sum_{v\in\mathcal{V}} (q^+_v + q^-_v) + \alpha^H \sum_{(v,v')\in H} (h^+_{vv'} + h^-_{vv'}) + \alpha^P \sum_{(v,v')\in P} (\beta_{vv'}(z^+_{vv'} + z^-_{vv'}) + \gamma^-_{vv'} w^-_{vv'} + \gamma^+_{vv'} w^+_{vv'}) \tag{3.9}$$

subject to

$$\sum_{s\in} a_{vs} y_s = 1, \qquad \forall v \in V \tag{3.10}$$

$$\sum_{s\in S^d} y_s \le b_d, \qquad \forall d \in D \tag{3.11}$$

$$\sum_{k\in K_v} \sum_{s\in S} (t^k_v - t_v) e^k_{vs} y_s \le x_v \le \sum_{k\in K_v} \sum_{s\in S} (t^k_v - t_v) o^k_{vs} y_s, \qquad \forall v \in V \tag{3.12}$$

$$x_v = q^+_v - q^-_v, \qquad \forall v \in \mathcal{V} \tag{3.13}$$

$$x_{v'} - x_v = h^+_{vv'} - h^-_{vv'}, \qquad \forall (v,v') \in H \tag{3.14}$$

$$x_v - x_{v'} - \tau_{vv'} = r^+_{vv'} - r^-_{vv'}, \qquad \forall (v,v') \in P \tag{3.15}$$

$$w^+_{vv'} \ge r^+_{vv'} - z^+_{vv'} U^+_{vv'}, \qquad \forall (v,v') \in P \tag{3.16}$$

$$w^-_{vv'} \ge r^-_{vv'} - z^-_{vv'} U^-_{vv'}, \qquad \forall (v,v') \in P \tag{3.17}$$

$$r^+_{vv'} - u^+_{vv'} \le z^+_{vv'} (U^+_{vv'} - u^+_{vv'}), \qquad \forall (v,v') \in P \tag{3.18}$$

$$r^-_{vv'} - u^-_{vv'} \le z^-_{vv'} (U^-_{vv'} - u^-_{vv'}), \qquad \forall (v,v') \in P \tag{3.19}$$

$$x_v \in \mathbb{Z}, \quad q_v^-,\ q_v^+ \geq 0, \qquad \forall v \in V \tag{3.20}$$

$$h_{vv'}^-,\ h_{vv'}^+ \geq 0, \qquad \forall (v,v') \in H \tag{3.21}$$

$$z_{vv'}^-,\ z_{vv'}^+ \in \{0,1\}, \quad r_{vv'}^-,\ r_{vv'}^+,\ w_{vv'}^-,\ w_{vv'}^+ \geq 0, \qquad \forall (v,v') \in P \tag{3.22}$$

$$y_s \in \{0,1\}, \qquad \forall s \in S \tag{3.23}$$

In this model, the objective (3.9) is to minimize a weighted multi-objective function of the operational costs and the penalties applied for shifting the selected trips, deviations from initial headways, and deviations from planned schedules for ideal passenger connections. Constraints (3.10) and (3.11) ensure that all trips are covered with the available vehicles in each depot. Inequalities (3.12) link the schedule choice (variables y_s) to the possible deviations of the trip start times (variables x_v). Constraints (3.13) and (3.14) define the variables q_v^+, q_v^-, $h_{vv'}^+$, and $h_{vv'}^-$, for the penalization of trip shifting and deviations from the original headways which are preferred to remain constant. Penalties for passenger connections are derived through the constraints (3.15)–(3.19). Constraints (3.16)–(3.19) linearize the penalty functions $f_{vv'}^P(x_{v'} - x_v - \tau_{vv'})$. In particular, for a pair of trips $(v,v') \in P$, $w_{vv'}^+$ is set equal to $r_{vv'}^+$ by the corresponding inequality (3.16) unless the connection is inconvenient, where $z_{vv'}^+ = 1$. Also, the relations (3.17) determine the values of the variables $w_{vv'}^-$. If $r_{vv'}^+ > u_{vv'}^+$, then $z_{vv'}^+$ is set to 1 thanks to the corresponding inequality (3.18). Similarly, Inequalities (3.19) set the values of the variables $z_{vv'}^-$. Finally, Constraints (3.20)–(3.23) restrict the domains of the variables.

3.2.3 Underlying network

If we consider the classic MDVSP network as a graph, each node represents a trip that needs to be covered. Each arc shows a possible connection between two bus trips. For each depot $d \in D$, a graph G_d is built containing a source node and a destination node. For each trip of V, we have a trip node that is connected to the source and destination. We can create an arc connection between nodes v_i and v_j, if and only if we are able to chain trip v_j after trip v_i, or if the time it takes to travel between stations E_i and S_j is less than $t_{v_j}^s - t_{v_i}^a$. The cost of each arc leaving from the source node is equal to the fixed cost (c_f) of operating a bus plus the cost of traveling from the depot to the start location of the trip. The cost of a return arc is the same as a departure arc. Figure 3.2 illustrates the general hybrid network presented in [1] for MDVSP-CTS, in which we consider copies for each trip that is selected to be shifted. Each shifted trip has a chain of arcs linking copy nodes that connect it, and the chains can be connected by one or more connection arcs. Also, a feasible schedule for each bus assigned to depot d is represented by a path in G_d from n_d^0 to n_d^1. This schedule has a cost of $c_s = \sum_{(i,j)\in A(s)} c_{ij}$, where $A(s)$ designates the path's set of arcs denoting s, as well as c_{ij} which denotes the cost for each arc (i,j).

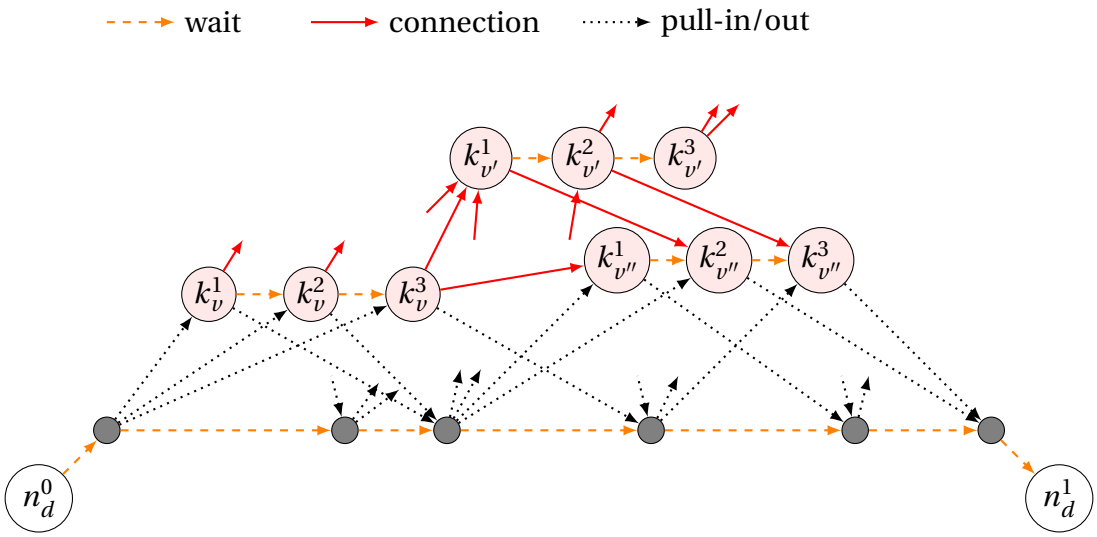

Figure 3.2 Hybrid network structure for MDVSP-CTS

Table 3.1 Costs of the arcs in G_d

Arc (i,j)	Type	c_{ij}
$(n^0_d, DN.first)$	First arc	ψ
$(DN.last, n^1_d)$	Last arc	0
$(dn, dn.next)$	Wait arc between depot nodes	0
$(k^i_v, n^{In}(k^i_v))$	pull-in arc from trip copy k^i_v	$\rho^T \tilde{T}_{vd}$
$(n^{Out}(k^i_v), k^i_v)$	pull-out arc to trip copy k^i_v	$\rho^T T_{dv}$
(k^i_v, k^{i+1}_v)	Wait arc between the trip copies	$\rho^W(t^{i+1}_v - t^i_v)$
$(k^i_v, k^j_{v'})$	Connection arc between the trip copies k^i_v and $k^j_{v'}$	$\rho^T \tilde{T}_{vv'} + \rho^W(t^j_{v'} - t^i_v - T_{vv'})$

N_d contains a trip node for each trip $v \in V^d$. In addition, the source node n^0_d and sink node n^1_d stand for the beginning and end of a schedule, respectively. The costs of each arc, as well as their type are shown in Table 3.1 as presented in [1]. Here, all times are presented in minutes. Also, a list of depot (gray) nodes, DN, is used to build the network G_d for a particular depot $d \in D$. $DN.first$ and $DN.last$ show the first and last elements in this list. For each trip v, nodes n^{out}_v and n^{in}_v represent a departure from the depot to arrive at the start of the trip exactly at time t_v and a return to the depot from the end of the trip, respectively. The duration between the end of the trip v and depot d is measured in $\tilde{T}_{vd}$. In contrast to T_{vd} and $T_{vv'}$, $\tilde{T}_{vd}$ and $\tilde{T}_{vv'}$ do not include the duration of trip v. However, $\tilde{T}_{vv'}$ is the travel time between the end location of trip v and the start location of trip v'. The paid travelling time is presented by $\tilde{T}_{vv'}$ in the arc cost of a connecting arc $(k^i_v, k^j_{v'})$, however, the paid waiting time is calculated as $t^j_{v'} - t^i_v - T_{vv'}$.

CHAPITRE 4 SOLUTION ALGORITHMS

In this chapter, the two-phase heuristic algorithm that has been proposed in the work of [2] is detailed in Section 4.1, and models in each phase are shown in Subsections 4.1.1 and 4.1.4, respectively. In Subsection 4.1.2, an overview of the applied column generation is presented. In addition, the heuristic approach is detailed in Subsection 4.1.3. Finally, Section 4.2 describes in detail how this process was accelerated using machine learning. First, data pre-processing techniques used are detailed in Subsection 4.2.1, and then the properties of the designed machine learning model is provided in Subsection 4.2.2. Also, Subsection 4.2.3 details the training, and the testing of the designed prediction model. Finally, Subsection 4.3 elaborates on different stages for acceleration in the MDVSP-CTS optimization model.

4.1 The two-phase heuristic of Desfontaines and Desaulniers [2]

Model (3.9)–(3.23) has a fairly large size for cases in real life. Consequently, Desfontaines and Desaulniers [2] proposed a two-phase heuristic to solve the MDVSP-CTS in order to produce computing times that are suitable for the industry. A column-generation heuristic is used in the first phase to compute the vehicle schedules, and a mixed integer programming model is used in the second phase to discover the best timetable given the computed vehicle schedules. To improve the likelihood of discovering a higher-quality timetable in the second phase, penalties are imposed in the first phase to the costs of specific arcs in the networks G_d, $d \in D$ in order to minimize the impact of the selected schedules in Phase I on the timetable selection in Phase II.

4.1.1 Phase I: A column generation heuristic

In the first phase of this two-phase approach, the MDVSP-CTS is solved utilizing trip shifting without much control and is modelled as an integer program (3.1)–(3.4). A column generation heuristic is used to calculate the bus schedules. Hence, bus schedules are fixed in Phase I. In fact, this model has a staggering number of variables. Therefore, as the number of trips m increases, it becomes impractical to explicitly generate every schedule in S. This barrier is

overcame by employing a column generation method to solve its linear relaxation, known as the master problem (MP), integrating it with a diving heuristic in order to obtain an integer solution. Furthermore, while solving large-scale instances, significant degeneracy occurs throughout the column generation process. It is feasible to mitigate degeneracy by introducing effective variables and expressing the amount to which each trip can be under- or over-covered, based on a constraint perturbation method described in [41]. This approach is described in detail in [2].

4.1.2 Column generation

Column generation is an iterative technique that solves a Restricted Master Problem (RMP) plus one or more pricing problems at each iteration (see, [42]). In RMP, only the linear relaxation of model (3.1)–(3.4) is considered. In the Master Problem (MP), $(\lambda_v)_{v\in V}$ and $(u_d)_{d\in D}$ are the dual variables associated with constraints (3.2) and (3.3).

The reduced cost $\tilde{c}_s$ of each schedule $s \in S^d$ which is assigned to a bus of depot $d \in D$ is as follows:

$$\tilde{c}_s = c_s - \sum_{v\in V} a_{vs}\lambda_v - u_d = \sum_{(i,j)\in A(s)} \tilde{c}_{ij} \tag{4.1}$$

where $\tilde{c}_{ij}$, $(i,j) \in A_d$, is the modified cost of each arc (i,j) is presented as below:

$$\tilde{c}_{ij} = \begin{cases} c_{ij} - u_d & \text{if } i = n_d^0 \\ c_{ij} - \lambda_v & \text{if } i \text{ is a copy node of trip } v \text{ and } j \text{ is not} \\ c_{ij} & \text{otherwise.} \end{cases} \tag{4.2}$$

As previously noted in Chapter 3, the start and end of each schedule are represented by the source node n_d^0 and sink node n_d^1, respectively. These arc costs are changed after each iteration in accordance with the values of the dual variables in the current RMP's solution. Following that, a dynamic programming algorithm for the shortest path problem on an acyclic network is used to solve each pricing problem [43].

In addition, for a given trip v, all arcs exiting a copy node have a penalty that decreases with the corresponding departure time (i.e., the earlier the worst) and all arcs entering a copy node have

a penalty that increases with the corresponding time (i.e., the earlier the better). In this way, the algorithm tends to choose early entering arcs and late exiting arcs, leaving more flexibility to determine the best schedule in Phase II.

4.1.3 Diving heuristic

Column generation can be implemented into a branch-and-bound framework to provide an exact branch-and-price algorithm in order to reach an optimal solution. See, [42] and [44]. According to [2], a diving heuristic [45] is used to find an integer solution. This is due to the fact that even for small-sized instances, too many branch-and-bound nodes are required to be analysed. Here, column generation solves a MP modified by fixed variables at each iteration and is continued until an integer solution is obtained. Alternatively, before proceeding to the next iteration, a variable y_s with the highest fractional value in this solution is picked and set to 1. This heuristic descends into the search tree, and investigates a single branch (probably with several nodes) until the column generation algorithm ends with an integer solution. Here, the schedule of a vehicle is permanently fixed after each node. As a result, the number of nodes to investigate is dependent on the number of vehicles.

4.1.4 Phase II: Solving a mixed integer program

In the second phase, by taking into consideration the schedules discovered in the first phase, the departure schedules are optimized. The goal of the second phase is to control the modifications to the schedules. Then, the initially planned bus schedules are used to adjust trip departure times in light of the timetable quality objectives. Finally, the following integer problem can be presented to find an ideal timetable subject to all the constraints, which limit the bus trip departure times based on the vehicle schedules chosen in Phase I. In a schedule chosen in Phase I, $\mathcal{CV}$ denotes the set of all pairs of trips that are completed consecutively (without returning to the depot). If $(v, v') \in \mathcal{CV}$, then the values of x_v and $x_{v'}$ should be selected in Phase II to guarantee that the connection between trips v and v' remains feasible. In this case, we would have a possible connection. $T_{vv'}$ is defined as the minimum time required between the departure times of v and v' so that the connection between v and v' stays feasible. In fact, $T_{vv'}$ presents

the duration of trip v plus the travel time between the end location of v and the start location of v'. As a result, the following inequalities are included further into the model:

$$(t_{v'} + x_{v'}) - (t_v + x_v) \geq T_{vv'}, \qquad \forall (v, v') \in \mathcal{CV} \tag{4.3}$$

$\mathcal{PO}$ and $\mathcal{PI}$ denote the sets of all trips that are carried out immediately following a pull-out trip and immediately before a pull-in trip, respectively. The MILP solved in Phase II is the following. The index of the copy $k_v^{i(v)} \in K_v$ of the trip v chosen in the Phase I solution is denoted by $i(v)$ for each $v \in \mathcal{PO} \cup \mathcal{PI}$.

$$\min \quad \alpha^T \sum_{v \in \mathcal{V}} (q_v^+ + q_v^-) + \alpha^H \sum_{(v,v') \in H} (h_{vv'}^+ + h_{vv'}^-) + \alpha^P \sum_{(v,v') \in P} (\beta_{vv'}(z_{vv'}^+ + z_{vv'}^-) + \gamma_{vv'}^- w_{vv'}^- + \gamma_{vv'}^+ w_{vv'}^+) \tag{4.4}$$

$$\text{s.t.} \quad (t_{v'} + x_{v'}) - (t_v + x_v) \geq T_{vv'}, \qquad \forall (v, v') \in \mathcal{CV} \tag{4.5}$$

$$x_v \geq t_v^{i(v)} - t_v, \qquad \forall v \in \mathcal{PO} \tag{4.6}$$

$$x_v \leq t_v^{i(v)} - t_v, \qquad \forall v \in \mathcal{PI} \tag{4.7}$$

$$d_v^{min} \leq x_v \leq d_v^{max}, \qquad \forall v \in V \tag{4.8}$$

$$x_v = q_v^+ - q_v^-, \qquad \forall v \in \mathcal{V} \tag{4.9}$$

$$x_{v'} - x_v = h_{vv'}^+ - h_{vv'}^-, \qquad \forall (v, v') \in H \tag{4.10}$$

$$x_v - x_{v'} - \tau_{vv'} = r_{vv'}^+ - r_{vv'}^-, \qquad \forall (v, v') \in P \tag{4.11}$$

$$w_{vv'}^+ \geq r_{vv'}^+ - z_{vv'}^+ U_{vv'}^+, \qquad \forall (v, v') \in P \tag{4.12}$$

$$w_{vv'}^- \geq r_{vv'}^- - z_{vv'}^- U_{vv'}^-, \qquad \forall (v, v') \in P \tag{4.13}$$

$$r_{vv'}^+ - u_{vv'}^+ \leq z_{vv'}^+ (U_{vv'}^+ - u_{vv'}^+), \qquad \forall (v, v') \in P \tag{4.14}$$

$$r_{vv'}^- - u_{vv'}^- \leq z_{vv'}^- (U_{vv'}^- - u_{vv'}^-), \qquad \forall (v, v') \in P \tag{4.15}$$

$$x_v \in \mathbb{Z}, \quad q_v^-,\ q_v^+ \geq 0, \qquad \forall v \in V \tag{4.16}$$

$$h_{vv'}^-,\ h_{vv'}^+ \geq 0, \qquad \forall (v, v') \in H \tag{4.17}$$

$$z_{vv'}^-,\ z_{vv'}^+ \in \{0, 1\}, \quad r_{vv'}^-,\ r_{vv'}^+,\ w_{vv'}^-,\ w_{vv'}^+ \geq 0, \qquad \forall (v, v') \in P \tag{4.18}$$

As the novelties of this model compared with MDVSP-CTS, here, the objective function (4.4) minimizes the total of the penalties regarding the shifted trips, deviations from the intended headways, and deviations from the targeted timetables to maintain a good satisfaction level for passenger connections. Constraints (4.5), (4.6) and (4.7) guarantee the consistency with fixed schedules in phase I. Constraints (4.6) and (4.7) are imposed in phase II to ensure that the pull-out and pull-in trips are feasible without lengthening the wait time outside a depot. Constraints (4.8) restrict the variables x_v to remain within their range of potentially changing values, where d_v^{min} and d_v^{max} represent the smallest and largest potential deviations from the original schedule for trip $v \in V$. In addition, constraints (4.9)–(4.18) are identical to their counterparts (3.13)-(3.22).

4.2 Machine learning

In this thesis, the main concept is to choose a subset of trips that can be shifted. In practice, we limit the trips to those with this possibility of shifting. In order to accomplish this, we examine each trip's potential for being shifted based on the predictions provided by a trained machine learning model. To design a training model, first we define the essential features for the predictions. The machine learning model that analyses specific trip's features (characteristics) aims to determine whether it needs to be shifted or not, and is explained in detail and underlined in the following paragraphs.

To enable the machine learning model to learn and get trained, we first define a list of features for each bus trip in order to obtain predictions by the model. These characteristics come from the result of the algorithm that was previously ran on the data provided by Giro Inc. as well as from the generated data that we used to do experiments on large-scale instances. The list of features that were defined for each bus trip, and used to train the machine learning model is shown in Table 4.1.

In our research, we took into account the potential bus connections over a 20-minute time frame. Additionally, within 10 minutes, we determine the number of connections that could be made without deadheading (dh). The developed machine learning model can judge the features for a given trip, and predict whether it should be shifted or not. The steps are explained and underlined in the following paragraphs. The objective is to apply a supervised learning method, and the labels, namely, "Shifted or not" are binary. In order to train the model, we use a set of

Table 4.1 List of the features

Feature name	**Definition**	**Domain of possible values**
Trip name	Each bus trip is given a specific ID	string
Start location	Each bus trip has a specific start location	string
End location	Each bus trip has a specific end location	string
Start time	Each bus trip has a specific start (departure) time) (in minutes)	[240,1379]
End time	Each bus trip has a specific end (arrival) time) (in minutes)	[271,1438]
Duration	End time - Start time (in minutes)	[6,64]
The previous headway	Trip's start time - previous trip's start time (in minutes)	[10,427]
The next headway	Next trip's start time - trip's start time (in minutes)	[10,427]
Bus line	Each trip is assigned to a specific bus line	string
# of possible connections with dh	trips that can connect to this end location for the passengers to switch bus line, and arrive to the destination by a connection trip with their current trip within 20 min	[0,14]
# of possible connections without dh	non-empty trips that can connect to this end location for the passengers to switch bus line, and arrive to the destination by a connection trip with their current trip within 10 min	[0,3]
tmax	The parameter specifying the maximum amount of time (in minutes) for shifting the departure time of each trip	{2}
# of missed connections	the missing trips starting at 1 or 2 minutes ago (Maximum of $tmax$ minutes) by less than tmax	[0,10]
# of possible connections before with dh	trips that can connect to this start location for the passengers to switch bus line, and arrive to the destination by a connection trip with their current trip within 20 min	[0,14]
# of possible connections before without dh	non-empty trips that can connect to this start location for the passengers to switch bus line, and arrive to the destination by a connection trip with their current trip within 10 min	[0,3]
# of preceding arrivals	arrivals that arrive at the same terminal of the trip within an interval of 1 or 2h	[0,21]
# of succeeding departures	departures that depart at the same terminal of the trip within an interval of 1 or 2h	[0,22]
Shifted or not	Binary output which is set to 1 if the trip is shifted	$\{0,1\}$

shifted trips, which the CG heuristics algorithm generated from original data, and which specify the trips that should be shifted.

4.2.1 Data pre-processing

The data must be pre-processed before it is used for the model. Data pre-processing techniques transform the data into a format that can be processed by the classification algorithms more quickly and efficiently. This pre-processing phase is carried out to scale and normalize the data of each instance separately due to the evident variations in the feature value range from one instance to another [34]. One important issue was to ensure no missing values exist. These missing values mean that the data would be unusable for the model. To deal with this problem, based on domain knowledge, they are replaced with a large value. This change would be applied to all features with null values and all data frames. Also, machine learning models deal with numerical values, and cannot handle string objects. Therefore, "trip name", "start location", and "end location" are dropped from the data frames. As a result, the data would not have any missing or string values, and can be utilized by machine learning models. Finally, in order to improve the performance of the machine learning model, "Min Max Scaler" [46] is used for the scaling of the data. This scaler is only fitted on training data as test and validation data should remain hidden from the model, however, the transformation is done on all data sets.

4.2.2 Prediction model: Neural Network

The chosen model for this problem is a Neural Network (NN). The popular Python library Tensorflow [47] has been chosen for the implementation of this model. Aside from the input layer, the model uses three hidden layers. These layers include 300, 150, and 70 neurons respectively. Dropout layers are added after each of the hidden layers, with a rate of 0.4 to prevent overfitting. The model needs to determine whether a trip should be shifted or not. Therefore, as this is a binary classification problem, a single neuron with the "$Sigmoid$" activation function was chosen for the output. The activation function for the hidden layers is "$ReLU$", due to the resulting decrease in training time.

The "$Adam$" optimizer is chosen with a learning rate of 0.001. The loss function is "binary cross entropy" as this is a binary classification problem. Accuracy is measured alongside the loss while training the model. The value of 128 has been chosen for the "Batch size" in training as it is not only too large, but also not too small. The result shows that the training times would not be too high, and the decrease in accuracy would be minimal.

The characteristics of the proposed neural network are clearly listed in the table 4.2.

Table 4.2 The characteristics of the proposed neural network model

Learning rate	10^{-4}
Epochs	1000
Batch size	128
Loss function	Binary cross entropy
Activation function	ReLU
Architecture	300*300*150*150*70*70*1
Optimizer	Adam

4.2.3 Training and Testing the data

The training duration for the model is 1000 epochs to be long enough. However, the "Early stopping" mechanism is used. The validation loss is monitored during the training, and thus the best state is saved. A smallest loss is preferable. However, it can be determined when the early stopping mechanism would decide that the loss would not get smaller with more training. The Keras [48] implementation of this mechanism has a parameter named "patience", which decides how many epochs can be tolerated. This parameter specifies the "Number of epochs with no improvement after which training will be stopped". To clarify, when, for instance, the patience has been chosen to be 30 epochs, if there has been no improvement of the validation loss for 30 consecutive epochs, then the training would be stopped. Finally, a confusion matrix (NxN) is used to evaluate the outcomes and precision of a classification machine learning model, and has been observed using the Scikit-learn implementation [46]. As N is referred to the number of class labels, therefore, we have a 2 x 2 confusion matrix in this case since this is a binary classification [49]. (The binary output indicates whether or not the trip is shifted.) It consists of the following four attributes: TP, TN, FP, and FN. True positive (TP) is referred to the number of correctly predicted positive outcomes (In this case, the trips that are correctly predicted to be shifted). The percentage of correctly predicted negative outcomes known as true negatives (TN). False positives (FP) is also referred to the number of incorrectly positive forecasts. Finally, the number of incorrectly predicted negative outcomes is noted as false negative (FN). It is important to note that different orderings of FN, FP, TN, and TP exist for confusion matrices. The Scikit-learn version [46] is provided in Figure 5.3. The plots and the confusion matrix are fully explained in detail in Section 5.2 of this document. As the outcome of our designed machine learning model, the predictions are generated for all the trips of each instance. These predictions are in the form of a list of 0s and 1s. As a final step, the outputs are extracted to allow the machine learning model's prediction to be used with the CG heuristic optimization algorithm presented in the work of [2]. The results of these optimization tests are provided in detail in Section 5.3.1.

4.3 Optimization pipeline

As previously stated in Chapter 1, and specifically in Section 1.2, the main goal of this thesis is to reduce the size of the MDVSP-CTS model by using machine learning predictions, which in turn reduces the CPU time required by the two-phase heuristic proposed in [2].

Assuming that the prediction model has been trained, the following optimization pipeline is used when an MDVSP-CTS instance has to be solved.

1- Read the input data and compute all the features required to represent each bus trip (see, Table 4.1).

2- Use the trained prediction model to determine a subset W of trips that can be shifted. A trip belongs to W if the model returns a probability larger than or equal to 0.3 that the trip will be shifted.

3- Run the phase I column generation heuristic (see Sections 4.1.1 to 4.1.3) of the two-phase heuristic considering a reduced network obtained by limiting the trips that can be shifted to those in set W. In the reduced network, there is a single copy node for each trip in W and multiple copy nodes for the trips in $V \setminus W$ (see, Figure 3.2)

4- Considering the solution obtained in the previous step, solve a restricted version of the phase II MILP model (4.4)-(4.18). The restriction is induced by setting all x_v variables equal to 0 for all trips $v \in W$.

5- The solutions computed in the two phases are combined to yield the final solution. Each bus schedule is obtained by taking a sequence of bus trips from the Phase I solution and its corresponding schedule computed in Phase II.

CHAPITRE 5 EXPERIMENTAL RESULTS

In this chapter, various instances of the input data are described in Section 5.1. Additionally, the accuracy and loss function plots regarding the developed machine learning model are provided in Section 5.2. Finally, a detailed evaluation of the MDVSP-CTS optimization tests where the Machine Learning model was applied in order to decrease the computing time of the two-phase heuristics provided in Section 5.3 and 5.3.1.

5.1 Instances

In this thesis, we have used generated MDVSP-CTS instances based on real-life data from the STM provided by our industrial partner, Giro Inc., in which the networks correspond to some subsets of bus lines. The instance generator of [7] takes these lines as input, and the actual trip frequencies on each line that may vary during the day. Then, to generate an instance with a targeted number of trips, the frequencies are adjusted on each line, and the departure times of each trip are determined with some randomness while respecting regular headways.

For training the machine learning model, we needed a large number of MDVSP-CTS instances, and we divided the set of instances available. We used 10 distinct bus networks. 100 instances have been generated for each network. Hence, in order to propose an efficient solution of the existing MDVSP-CTS optimization problem, we aimed to design a machine learning model using data from 1000 instances. So, we split the instances as follows: 70% for training, 15% for validation and 15% for testing in a randomized way. Table 5.1 lists the characteristics for each network, including average, minimum and maximum number of trips, number of depots, and number of lines per instance. Regarding the provided information in Table 5.1, it should be noted that the number of depots is the only difference between networks T1 and T1b, and also for the other networks. Network T1 has a single depot, whereas Network T1b has two depots for the buses to be stored. Also, the given times are in seconds, obtained with the two-phase heuristic algorithm in [1]. The results in the last six rows were obtained by solving the instances using the two-phase heuristic applied to the full model (i.e., all trips can be shifted).

Table 5.1 The characteristics of each bus network

Network name	**T1**	**T1b**	**T2**	**T2b**	**T3**	**T3b**	**T4**	**T4b**	**T5**	**T5b**
# of bus lines	8	8	6	6	8	8	8	8	6	6
# of depots	1	2	1	2	1	2	1	2	1	2
Maximum # of trips	512	512	228	228	174	174	512	512	246	246
Minimum # of trips	396	396	191	191	124	124	368	368	190	190
Average # of trips	446	446	208	208	149	149	419	419	213	213
Minimum # of shifted trips	0	0	0	0	18	88	37	0	1	0
Maximum # of shifted trips	163	153	33	31	71	143	290	81	72	70
Average # of shifted trips	37	37	12	12	43	113	81	20	21	18
Minimum CPU time	0	323	32	114	1	20	336	0	3	9
Maximum CPU time	684	2525	158	353	6	123	2164	3294	14	37
Average CPU time	254	773	60	200	3	64	687	1605	7	20

5.2 Machine learning results

Two plots of the loss and accuracy of the model on the training and test data for each epoch of the training are clearly illustrated in Figures 5.1 and 5.2. On one hand, the plot in Figure 5.1 shows the evolution of training loss and validation loss over epochs. On the other hand, Figure 5.2 highlights the trend of training and validation accuracy per epoch. Moreover, the confusion matrix is observed in Figure 5.4 by the legend shown in Figure 5.3.

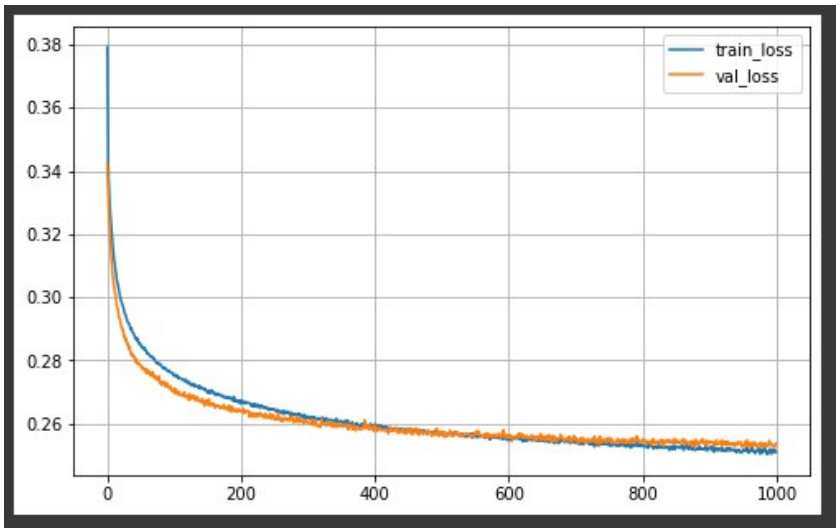

Figure 5.1 Loss over epochs

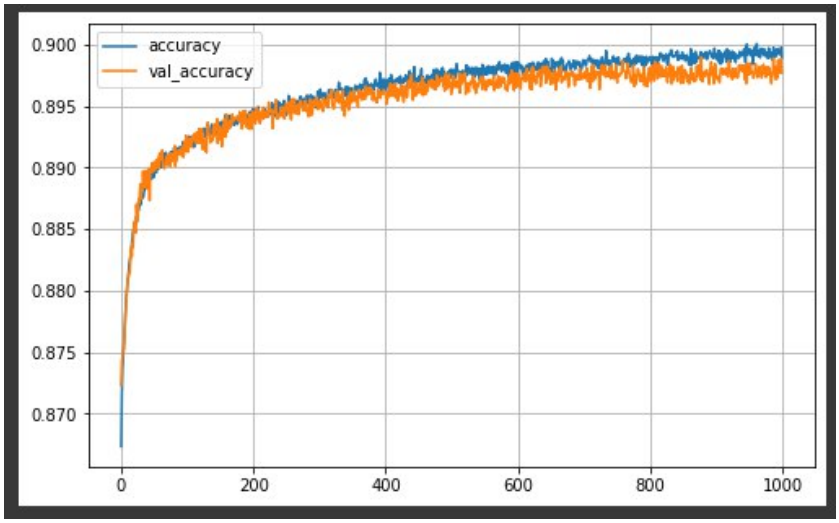

Figure 5.2 Accuracy over epochs

TN	FP
FN	TP

Figure 5.3 Legend for the confusion matrix

34701	2558
2260	3333

Figure 5.4 Confusion matrix

In this case, reducing the number of false negatives takes precedence over reducing the number of false positives. This is because, in cases of false negatives, we fail to make those trips that needed to be rescheduled. However, the only issue with false positives is that we are unable to reduce computation time. The recommendation to lower the quantity of false negatives is clearly detailed in Section 6.2 as a limitation in this research.

5.3 New heuristic results

In order to demonstrate the impact on the CPU times, and the reduction based on three different scenarios, 15 instances of an arbitrary network T4 are chosen as the selected instances for the implementation of the optimization tests. This network was chosen as an instance among all 10 networks to show the potential of the method when considering an optimal subset of trips that can be shifted. To show this potential, we have considered only a smaller subset of instances from the same series, such as "Network T4" test instances since most of the networks were designed for training and can be solved quite rapidly. This decision was made because according to Table 5.1, the solution report required a significant amount of computing time while also requiring a large number of shifted trips compared to other networks.

We solve the program using three distinct cases for each instance. The detailed comparison tables and discussions are provided in Subsection 5.3.1.

5.3.1 Three scenarios: Original, Subset, and Neural

Three different scenarios are assessed in order to better understand the impact of using machine learning on MDVSP-CTS optimization problem.

— **Original**: The MDVSP-CTS is firstly solved while taking into account all trips to be shifted. In this scenario, as all trips are considered to be potential for being shifted or not, there is no restriction on the trips that may be selected for a shift in their schedule.

— **Subset**: In the second scenario, we limited this case to only allow the shifting to the subset of trips specified from the solution of the original input data, which is retrieved from the output of the optimization algorithm in the first scenario. This scenario is used to assess the potential of the proposed approach.

— **Neural**: As opposed to the second scenario, we consider the limitation of allowing only those trips from the machine learning model's prediction to be changed. According to the binary classification model, which identifies the class of shifted trips, this technique effectively uses our predicted trips which have a value 1 in their output label.

Tables 5.2, 5.3, and 5.4 clearly illustrate the comparison between these three algorithms in terms

of CPU time and total cost. The values in all tables that are in red indicate a loss, whilst those in black indicate a gain. Based on Tables 5.3, and 5.4, it can be deduced that in the second scenario, limiting the trips that might potentially be shifted based on those specific shifted trips that we acquired by the result of the solution of our original data resulted in a 50.64% reduction in computing time. In comparison, the reduction is 53.03% when we restrict the trips based on the predictions produced by the developed machine learning model.

The outcomes shown in Tables 5.2 and 5.4 allow for three key observations. First, limiting the trips that might be considered shifted ones has an impact on total CPU time. It is evident that using the developed machine learning pre-processor allows us to reduce the execution time of the MSVSP-CTS algorithm by an average of 53%. Second, except for the two instances $R0$ and $R9$, we have a gain in terms of total cost. Third, since the accuracy was a high value of nearly 90% as shown in Figure 5.2, and we were able to cut the calculation time in half, it was demonstrated that the features (Section 4.2) for the neural network model were well-defined.

We achieved an average reduction of 50.64% of CPU time, as shown in Table 5.3, where the accuracy of predictions is regarded to be 100% because we employed the precise subset of trips to restrict the potential of trips being selected to shift. Finally, Table 5.5 compares the CPU time and total cost for the second and third scenarios. Despite the outcomes being relatively equivalent, the superior performance in the third scenario (neural network model) is apparent.

Based on Tables 5.2 and 5.4, the optimization model predicts a maximum value for the number of shifted trips with a value of 124 when the original case is taken into account. However, when we use the machine learning model, the maximum value for the selected shifted trips is 62, which is reduced by half. In the original scenario, there are a minimum of 39 shifted trips, whereas there are only 8 selected shifted trips in the scenario where we applied a neural network.

According to Table 5.3, the maximum number of shifted trips is 98, whereas the minimum number of shifted trips is 29. However, based on Table 5.4, in contrast to the second scenario where we employed an accurate subset of shifted trips, it is evident that the machine learning model projected fewer additional trips to be shifted. Hence, when compared to the first scenario, these values reveal fewer shifted trips, but more shifted trips when compared to the situation where

Table 5.2 The optimization results for the first scenario (original data)

ORIGINAL				
Instance	Number of trips	CPU time (seconds)	Total cost	Shifted trips
R0	442	364.05	21144.6	47
R1	423	385.82	23460.4	95
R2	407	318.79	23166.2	74
R3	412	299.95	23118.2	39
R4	368	202.17	20295.2	91
R5	411	400.16	22359.4	91
R6	410	328.9	21147.4	59
R7	455	514.62	23229.4	55
R8	439	502.76	23250.2	59
R9	457	572.25	21036	38
R10	387	273.17	22434.6	111
R11	415	380.93	23264.6	73
R12	433	458.74	21329	80
R13	512	836.64	25619.8	124
R14	405	310.82	23368.2	104
Average	425	410	22548	76

Table 5.3 The optimization results for the second scenario (subset model)

SUBSET					**Solution cost**	**CPU time**
Instance	**CPU time (in seconds)**	**Total cost**	**Shifted trips**	**Size of the subset**	**Gain/Loss**	**Gain**
R0	220.79	21134.8	42	47	0.05%	39.35%
R1	197.61	23365.6	81	95	0.40%	48.78%
R2	131.57	23134.6	68	74	0.14%	58.73%
R3	151.3	23066.4	36	39	0.22%	49.56%
R4	110.59	20278.8	83	91	0.08%	45.30%
R5	180.32	22293	82	91	0.30%	54.94%
R6	153.04	21057.2	51	59	0.43%	53.47%
R7	262.99	23166.8	49	55	0.27%	48.90%
R8	205.87	23199	48	59	0.22%	59.05%
R9	280.37	21964.4	29	38	-4.41%	51.01%
R10	148.41	22438.6	98	111	-0.02%	45.67%
R11	185.4	23231.6	66	73	0.14%	51.33%
R12	224.07	21332.6	70	80	-0.02%	51.16%
R13	418.84	25411.2	93	124	0.81%	49.94%
R14	147.76	23328.6	85	104	0.17%	52.46%
Average	201.26	22560.2	65	76	-0.08%	**50.64%**

Table 5.4 The optimization results for the third scenario (predicted data by neural networks model)

NEURAL					Solution cost	CPU time
Instance	**CPU time (seconds)**	**Total cost**	**Predicted shifted trips**	**Shifted trips**	**Gain/Loss**	**Gain**
R0	198.36	23113	56	24	-9.31%	46%
R1	200.6	23313.8	82	57	0.62%	48%
R2	129.39	23020	75	8	0.63%	59%
R3	152.19	23052	66	24	0.29%	49%
R4	97.59	20128.4	69	37	0.82%	52%
R5	163.62	22193	63	45	0.74%	59%
R6	158.8	21058.4	61	23	0.42%	52%
R7	252.45	23148	72	29	0.35%	51%
R8	203.56	23187.4	63	25	0.27%	60%
R9	267.05	21959.8	65	20	-4.39%	53%
R10	139.85	22396	75	62	0.17%	49%
R11	148.89	23242.6	78	43	0.09%	61%
R12	220.38	21267.6	75	39	0.29%	52%
R13	396.72	25313.2	88	61	1.20%	53%
R14	147.1	23308.8	84	60	0.25%	53%
Average	191.8	22646.8	71	37	-0.50%	**53%**

Table 5.5 CPU time and Solution cost comparison between Subset and Neural networks prediction

	Subset vs. Neural (CPU Time)	Solution cost
	REDUCTION (FROM NEURAL)	Gain/Loss
	-11.31%	8.56%
	1.49%	-0.22%
	-1.68%	-0.50%
	0.58%	-0.06%
	-13.32%	-0.75%
	-10.21%	-0.45%
	3.63%	0.01%
	-4.18%	-0.08%
	-1.13%	-0.05%
	-4.99%	-0.02%
	-6.12%	-0.19%
	-24.52%	0.05%
	-1.67%	-0.31%
	-5.58%	-0.39%
	-0.45%	-0.08%
Average	**-5.30%**	**0.37%**

we utilized a neural network model. According to Table 5.4, it is clear that we have lost in terms of solution cost in only two instances R_0 and R_9. Due to the heuristic nature of the approach, in the other instances, we have a better solution cost that was expected. For example, R_{11} has achieved the greatest gain in CPU time which has a value of 61%, and is greater than the average. Whereas instance R_0, with a value of 46%, yields the lowest gain in the CPU time. Also, instance R_0 faces the smallest gain in solution cost. The largest gain in solution cost for the instance R_4, however, is only about 1%. So, in terms of the solution cost, we get the conclusion that the gain is not significant. Also, the number of shifted trips is significantly lower in this scenario than it was in the original scenario, as was previously indicated, which accounts for the rise in solution cost. However, we faced that some instances had opposite behaviors. Table 5.5 shows that, when the exact subset of shifted trip results is compared to the predictions made by machine learning models, it is found that, in almost all cases, CPU time is reduced, with the exception of three cases (R_1, R_3, and R_6), which experienced very little gain with values of 1.49%, 0.58%, and 3.63%, all of which are below 4%. The fact that we experienced a greater reduction in CPU time when we used forecasts to be allowed to shift rather than allowing the exact subset of shifted trips is an important observation. As a result, both in terms of CPU time and solution cost, the neural network prediction model performed better. However, the amounts of gain and loss found in the solution cost are very comparable for both methods.

CHAPITRE 6 CONCLUSION AND RECOMMENDATIONS

In this chapter, we summarize the work that has been done in this research in Section 6.1. Before highlighting several recommendations for further research in Section 6.3, some significant limitations are mentioned in Section 6.2.

6.1 Summary of works

This master's thesis further optimizes the integrated bus scheduling model with controlled trip shifting in the work of [2] by reducing the computation time to an average of 53% for the 15 selected instances. So, it is possible to reduce the operational costs by trip shifting, however, it is essential to consider the time, budget, and resource limitations as well as the possible passenger connections because doing so helps to maintain a high level of quality that the transportation provider offers to the passengers.

According to the optimization findings in Table 5.5, it is determined that using the predicted shifted trips by the built machine learning pre-processor with an average accuracy of nearly 90% (which was shown in Figure 5.2) will enable us to reach the solution faster than using the precise prediction with a 100% accuracy. Finally, we get to the conclusion that this machine learning prediction model allows for further enhancement of MDVSP-CTS optimization model. Regarding our machine learning prediction, a pre-processed, transformed version of the set of all bus trips is served as the model's input data. Additionally, the predictions that indicate whether each trip should be shifted or not must be made by the trained model. Consequently, the optimization algorithm described in the work of [2] can run more quickly by restricting the potential trips that can be shifted in the original input data thanks to the machine learning model, which is a binary classification one.

6.2 Limitations of the research

First, it should be emphasized that using real data would result in more accurate results for a better conclusion. Also, it should be mentioned that although the accuracy of the artificial

neural network model is close to 90%, this might theoretically be increased still by alternative approaches. Another way to look at this research's limitations is from the perspective of the confusion matrix. In this problem, minimizing the number of false negatives has a priority over the number of false positives. To better understand this, false negatives are trips that are discarded from the predictions, while in reality, they should be shifted. Also, false positives are the trips that are added to the suggestions made for being shifted, but in reality, they should not be considered. Using this definition, it should be clear that while more false negatives would mean fewer shifted trips in the final result of the Gencol solver, which implements the CG method, the increase in the number of false positives would lead to an increase in the computation time, which is considered the less consequential outcome in this situation. One way to deal with this problem is to bias the predictions made for the trips towards 1 (shifted). However, while this leads to an increase in the number of true positives, and also a decrease in the number of false negatives which is great, false positives increase and the accuracy decreases. The bias that has been made in this research only decreases the accuracy by 1 percent but decreases the false negatives by a relatively good amount. It would be possible to go one step further as the number of false negatives is decreased by this method. So, the benefits of our model's predictions would decrease because while the original trips list has no false negatives, it has a lot of false positives, and without removing them, there would be no reduction in computation time.
The objective of this study is to reduce the computation time by giving a perfect list of the shifted trips, which is the output of the Gencol solver for the original input data where all trips have the allowance to be considered for shifting. In essence, the model should only eliminate trips that are truly unnecessary, and then eliminate each and every one of these trips. On the one hand, if the model is excessively conservative, certain trips that should be discarded would not be eliminated. On the other hand, if the model is excessively negligent, certain trips that should be maintained would be removed.

6.3 Future Research

Future work would be made to improve the results of our current neural network model, and improve the accuracy, the loss, and the confusion matrix from the perspective that has been

stated. An improvement on the model by the likes of increasing the number of neurons or changing the structure would probably do little to improve the results. The proposed model has been tuned to make sure it is neither underfitting nor overfitting. In contrast, working further on the data could lead to promising improvements. One suggestion is to increase the volume of the data. Another suggestion is to work on Data Engineering. In fact, creating more calculated features from the current ones could help the model make better decisions and predictions. One idea that might also increase the model accuracy, and further decrease the loss, is to use a smaller learning rate, besides training for longer epochs.

Another idea is related to the data pre-processing step to experiment with various data scalers and normalization techniques.

REFERENCES

[1] L. Desfontaines et G. Desaulniers, “Multiple depot vehicle scheduling with controlled trip shifting,” *Transportation Research Part B: Methodological*, vol. 113, n^{o}. 1, p. 34–53, juin 2018.

[2] L. Desfontaines, “Problème d’horaire d’autobus avec dépôts multiples et modification contrôlée des heures de début des voyages,” Mémoire de maîtrise, École Polytechnique de Montréal, July 2017. [En ligne]. Disponible: https://publications.polymtl.ca/2622/

[3] R. Abduljabbar, H. Dia, S. Liyanage et S. A. Bagloee, “Applications of artificial intelligence in transport: An overview,” *Sustainability*, vol. 11, n^{o}. 1, 2019. [En ligne]. Disponible: https://www.mdpi.com/2071-1050/11/1/189

[4] K. Haase, G. Desaulniers et J. Desrosiers, “Simultaneous vehicle and crew scheduling in urban mass transit systems,” *Transportation Science*, vol. 35, n^{o}. 3, p. 286–303, 2001. [En ligne]. Disponible: https://doi.org/10.1287/trsc.35.3.286.10153

[5] A. van den HEUVEL, J. van den AKKER et M. Van Kooten, “Integrating timetabling and vehicle scheduling in public bus transportation,” *Reporte Técnico UU-CS-2008-003, Department of Information and Computing Sciences, Utrecht University, Holanda*, 2008.

[6] A. A. Bertossi, P. Carraresi et G. Gallo, “On some matching problems arising in vehicle scheduling models,” *Networks*, vol. 17, n^{o}. 3, p. 271–281, 1987. [En ligne]. Disponible: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230170303

[7] J. Brasseur, “Accélération d’une méthode d’agrégation dynamique de contraintes par apprentissage automatique pour le problème de construction d’horaires de conducteurs d’autobus,” Mémoire de maîtrise, École Polytechnique de Montréal, August 2022.

[8] S. Bunte et N. Kliewer, “An overview on vehicle scheduling models,” *Public Transport*, vol. 1, p. 299–317, 2009.

[9] G. Desaulniers et M. D. Hickman, “Chapter 2 public transit,” dans *Transportation*, ser. Handbooks in Operations Research and Management Science, C. Barnhart et

G. Laporte, édit. Elsevier, 2007, vol. 14, p. 69–127. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S0927050706140025

[10] O. Ibarra-Rojas, F. Delgado, R. Giesen et J. Muñoz, "Planning, operation, and control of bus transport systems: A literature review," *Transportation Research Part B Methodological*, vol. 77, p. 38–75, 12 2014.

[11] A. Oukil, H. B. Amor, J. Desrosiers et H. El Gueddari, "Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems," *Computers Operations Research*, vol. 34, n°. 3, p. 817–834, 2007, logistics of Health Care Management. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S0305054805001590

[12] O. du Merle, D. Villeneuve, J. Desrosiers et P. Hansen, "Stabilized column generation," *Discrete Mathematics*, vol. 194, n°. 1, p. 229–237, 1999. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S0012365X98002131

[13] C. C. Ribeiro et F. Soumis, "A column generation approach to the multiple-depot vehicle scheduling problem," *Operations research*, vol. 42, n°. 1, p. 41–52, 1994.

[14] A. Hadjar, O. Marcotte et F. Soumis, "A branch-and-cut algorithm for the multiple depot vehicle scheduling problem," *Operations Research*, vol. 54, n°. 1, p. 130–149, 2006. [En ligne]. Disponible: https://doi.org/10.1287/opre.1050.0240

[15] M. Groiez, G. Desaulniers, A. Hadjar et O. Marcotte, "Separating valid odd-cycle and odd-set inequalities for the multiple depot vehicle scheduling problem," *EURO Journal on Computational Optimization*, vol. 1, n°. 3, p. 283–312, 2013. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S2192440621000228

[16] G. Laporte, F. A. Ortega, M. A. Pozo et J. Puerto, "Multi-objective integration of timetables, vehicle schedules and user routings in a transit network," *Transportation Research Part B: Methodological*, vol. 98, p. 94–112, 2017. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S0191261515301776

[17] A. Schöbel, "An eigenmodel for iterative line planning, timetabling and vehicle scheduling in public transportation," *Transportation Research Part C: Emerging Technologies*, vol. 74, p. 348–365, 2017. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S0968090X1630242X

[18] R. E. Kraut, R. E. Rice, C. Cool, R. S. Fish et A. Lobel, “Vehicle scheduling in public transit and lagrangean pricing,” *Management Science*, vol. 44, p. 1637–1649, 1998.

[19] A. Pepin, G. Desaulniers, A. Hertz et D. Huisman, “A comparison of five heuristics for the multiple depot vehicle scheduling problem,” *Journal of Scheduling*, vol. 12, n^o. 1, p. 17–30, 2009.

[20] H. L. Petersen, A. Larsen, O. B. G. Madsen, B. Petersen et S. Ropke, “The simultaneous vehicle scheduling and passenger service problem,” *Transportation Science*, vol. 47, n^o. 4, p. 603–616, 2013. [En ligne]. Disponible: http://www.jstor.org/stable/43666912

[21] N. Kliewer, T. Mellouli et L. Suhl, “A time-space network based exact optimization model for multi-depot bus scheduling,” *European Journal of Operational Research*, vol. 175, n^o. 3, p. 1616–1627, 2006. [En ligne]. Disponible: https://EconPapers.repec.org/RePEc:eee:ejores:v:175:y:2006:i:3:p:1616-1627

[22] R. J. Solomonoff, “An inductive inference machine,” dans *IRE Convention Record, Section on Information Theory*, vol. 2. Institute of Radio Engineers New York, 1957, p. 56–62.

[23] A. Lodi et G. Zarpellon, “On learning and branching: a survey,” *Top*, vol. 25, n^o. 2, p. 207–236, 2017.

[24] H. Song, I. Triguero et E. Özcan, “A review on the self and dual interactions between machine learning and optimisation,” *Progress in Artificial Intelligence*, vol. 8, n^o. 2, p. 143–165, 2019.

[25] M. I. Jordan et T. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science*, vol. 349, p. 255 – 260, 2015.

[26] D. Weichert, P. Link, A. Stoll, S. Rüping, S. Ihlenfeldt et S. Wrobel, “A review of machine learning for the optimization of production processes,” *The International Journal of Advanced Manufacturing Technology*, vol. 104, p. 1889 – 1902, 2019.

[27] C. Gambella, B. Ghaddar et J. Naoum-Sawaya, “Optimization problems for machine learning: A survey,” *European Journal of Operational Research*, vol. 290, n^o. 3, p. 807–828, 2021. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S037722172030758X

[28] P. Bonami, A. Lodi et G. Zarpellon, “Learning a classification of mixed-integer quadratic programming problems,” dans *International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2018, p. 595–604.

[29] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser et B. Dilkina, “Learning to branch in mixed integer programming,” dans *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, n^{o}. 1, 2016.

[30] E. B. Khalil, B. Dilkina, G. L. Nemhauser, S. Ahmed et Y. Shao, “Learning to run heuristics in tree search.” dans *Ijcai*, 2017, p. 659–666.

[31] R. Václavík, A. Novák, P. Ščha et Z. Hanzálek, “Accelerating the branch-and-price algorithm using machine learning,” *European Journal of Operational Research*, vol. 271, n^{o}. 3, p. 1055–1069, 2018.

[32] U. J. Mele, L. M. Gambardella et R. Montemanni, “Machine learning approaches for the traveling salesman problem: A survey,” *2021 The 8th International Conference on Industrial Engineering and Applications(Europe)*, 2021.

[33] S. Li, Z. Yan et C. Wu, “Learning to delegate for large-scale vehicle routing,” *CoRR*, vol. abs/2107.04139, 2021. [En ligne]. Disponible: https://arxiv.org/abs/2107.04139

[34] M. Morabit, G. Desaulniers et A. Lodi, “Machine-learning-based arc selection for constrained shortest path problems in column generation,” *CoRR*, vol. abs/2201.02535, 2022. [En ligne]. Disponible: https://arxiv.org/abs/2201.02535

[35] ——, “Machine-learning-based column selection for column generation,” *Transp. Sci.*, vol. 55, n^{o}. 4, p. 815–831, 2021. [En ligne]. Disponible: https://doi.org/10.1287/trsc.2021.1045

[36] T. Tang, R. Liu et C. Choudhury, “Incorporating weather conditions and travel history in estimating the alighting bus stops from smart card data,” *Sustainable Cities and Society*, vol. 53, p. 101927, 2020. [En ligne]. Disponible: https://www.sciencedirect.com/science/article/pii/S2210670719321274

[37] W. Huang, Y. Chen et X. Zhu, “Vehicle routing optimization based on multimedia communication and intelligent transportation system,” *Journal of Advanced Transportation*, vol. 2022, p. 1–9, 03 2022.

[38] S. Sun, Z. Cao, H. Zhu et J. Zhao, “A survey of optimization methods from a machine learning perspective,” *IEEE Transactions on Cybernetics*, vol. 50, n^{o}. 8, p. 3668–3681, 2020.

[39] Y. Bengio, A. Lodi et A. Prouvost, “Machine learning for combinatorial optimization: a methodological tour d’horizon,” *European Journal of Operational Research*, vol. 290, 08 2020.

[40] A. V. Cabot et A. P. Hurter, “An approach to zero-one integer programming,” *Operations Research*, vol. 16, n^{o}. 6, p. 1206–1211, 1968. [En ligne]. Disponible: https://doi.org/10.1287/opre.16.6.1206

[41] A. Charnes, “Optimality and degeneracy in linear programming,” *Econometrica*, vol. 20, n^{o}. 2, p. 160–170, 1952. [En ligne]. Disponible: http://www.jstor.org/stable/1907845

[42] J. Desrosiers et M. Lübbecke, “A primer in column generation. g. desaulniers, j. desrosiers, mm solomon, eds. column generation,” 2005.

[43] L. Zeng et M. Zhao, “Improved dynamic programming for the shortes path problem with resource constraints in dag,” dans *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, 2018, p. 2278–2282.

[44] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P. Savelsbergh et P. H. Vance, “Branch-and-price: Column generation for solving huge integer programs,” *Operations Research*, vol. 46, n^{o}. 3, p. 316–329, 1998. [En ligne]. Disponible: http://www.jstor.org/stable/222825

[45] C. Joncour, S. Michel, R. Sadykov, D. Sverdlov et F. Vanderbeck, “Column generation based primal heuristics,” *Electronic Notes in Discrete Mathematics*, vol. 36, p. 695–702, 2010.

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, G. Louppe, P. Prettenhofer, R. Weiss, R. J. Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot et E. Duchesnay, “Scikit-learn: Machine learning in python,” *ArXiv*, vol. abs/1201.0490, 2011.

[47] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. A. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu et X. Zhang, “Tensorflow: A system for large-scale machine learning,” *ArXiv*, vol. abs/1605.08695, 2016.

[48] F. Chollet *et al.* (2015) Keras. [En ligne]. Disponible: https://github.com/fchollet/keras

[49] T. T. Teoh et Z. Rong, *Classification*. Singapore: Springer Singapore, 2022, p. 183–211. [En ligne]. Disponible: https://doi.org/10.1007/978-981-16-8615-3_11

APPENDIX A MATHEMATICAL NOTATIONS

$V = \{v_1, v_2, \ldots, v_m\}$: Set of trips

m: Number of trips

t_v: Scheduled start time for trip v, $\forall v \in V$

k_v^i: Copy of trip v created for each possible start time t_v^i for trip v

K_v: Set of copies of trip v

D: Set of depots

p: Number of depots

b_d: Number of available vehicles at depot d, $\forall d \in D$

V^d: Set of trips that could be assigned to the vehicles (buses) at depot $d \in D$

ψ: A large fixed cost for each vehicle used

ρ^W: A cost per minute for a vehicle waiting in a location outside of a depot

ρ^T: A cost per minute for traveling without passengers

S^d: Set of all feasible schedules for the vehicles in depot d, $\forall d \in D$

$S = \bigcup_{d \in D} S^d$: Set of schedules, $\forall d \in D$

c_s: Cost of schedule s, $\forall d \in D$ and $\forall s \in S^d$

a_{vs}: Binary parameter that is equal to 1 if and only if s covers trip v. $\forall v \in V$

y_s: Binary variable that takes value 1 if and only if schedule s is part of the solution. $\forall s \in S^d$

N_d: Sets of nodes, $\forall d \in D$

A_d: Sets of arcs, $\forall d \in D$

$G_d = (N_d, A_d)$: Network used to generate schedules in S^d, $\forall d \in D$

n_d^0: Source node (start of a schedule), $\forall d \in D$

n_d^1: Sink node (end of a schedule), $\forall d \in D$

n_v^{out}: Departure from the depot to reach the start of trip v, $\forall v \in V$

n_v^{in}: Return to the depot from the end of trip v, $\forall v \in V$

DN: List of depot nodes

DN.first: First items of DN

DN.last: Last items of DN

$T_{vv'}$: Duration of trip v plus the traveling time between the end location of v and the start location of trip v', $\forall$v,v'$\in V$

$A(s)$: Set of arcs in the path representing s, $\forall s \in S^d$

c_{ij}: Cost of each arc (i, j)

$\tilde{T}_{vd}$: Traveling time between the end of trip v to depot d, $\forall d \in D$ and $\forall v \in V$

$\tilde{T}_{vv'}$ is the traveling time between the end location of trip v and the start location of trip v', $\forall d \in D$ and $\forall v,v' \in V$

q_v^+: Non-negative variable for each $v \in V$ that represent the amount of time by which trip v is shifted forward

q_v^-: Non-negative variable for each $v \in V$ that represent the amount of time by which trip v is shifted backward

$x_v = q_v^+ - q_v^-$: The amount of shifting for trip v, $\forall v \in V$

e_{vs}^k: A binary parameter that is equal to 1 if schedule s enters the sub-network of trip v by copy k, and 0 otherwise. $\forall s \in S, \forall v \in V$

o_{vs}^k: A binary parameter that is equal to 1 if schedule s leaves the sub-network of trip v by copy k, and 0 otherwise. $\forall s \in S, \forall v \in V$

α^T: Penalty cost for maintaining the initial timetable

H: Set of all pairs of trips for which it is desirable to keep the initial headway

$h_{vv'}^+$: Non-negative variable to compute the positive deviations from the initial headway between v and v', $\forall (v,v') \in H$

$h_{vv'}^-$: Non-negative variable to compute the negative deviations from the initial headway between v and v', $\forall (v,v') \in H$

α^H: Penalty to keep the initial headway (positive coefficient)

P: Set of pairs of trips for which we would like to ensure a good passenger connection at a location where they intersect

$\tau_{vv'}$: A pre-specified value which is equal to $x_{v'} - x_v$, $\forall (v,v') \in P$

$r_{vv'}^+$: Non-negative variable to compute the positive deviations from the ideal target $\tau_{vv'}$

$r_{vv'}^-$: Non-negative variable to compute the negative deviations from the ideal target $\tau_{vv'}$

$f_{vv'}^P$: Penalty function that is applicable for the pair of trips $(v, v') \in P$

α^P: Penalty for the passenger connection for (v, v') by $f_{vv'}^P(x_{v'} - x_v - \tau_{vv'})$

$[-U^-_{vv'}, U^+_{vv'}]$: Interval for a domain of possible values of $x_{v'} - x_v - \tau_{vv'}$ for function $f^P_{vv'}$

$[-u^-_{vv'}, u^+_{vv'}]$: Interval of the values for which the passenger connection is deemed convenient

$\alpha^P \beta_{vv'}$: Penalty for a deviation yielding an inconvenient passenger connection

$\alpha^P \gamma^-_{vv'}$: A unit penalty for a negative deviation in the interval $[-u^-_{vv'}, u^+_{vv'}]$

$\alpha^P \gamma^+_{vv'}$: A unit penalty for a positive deviation in the interval $[-u^-_{vv'}, u^+_{vv'}]$

$z^+_{vv'}$: Binary variable which is equal to 1 if the deviation leads an inconvenient connection because $x_{v'} - x_v - \tau_{vv'} \in [-U^-_{vv'}, -u^-_{vv'}[$

$z^-_{vv'}$: Binary variable which is equal to 1 if the deviation leads an inconvenient connection because $x_{v'} - x_v - \tau_{vv'} \in [u^+_{vv'}, U^+_{vv'}[$

$w^+_{vv'}$: Non-negative variable to compute the positive deviation inside $[-u^-_{vv'}, u^+_{vv'}]$ which yields a linear penalty

$w^-_{vv'}$: Non-negative variable to compute the negative deviation inside $[-u^-_{vv'}, u^+_{vv'}]$ which yields a linear penalty

$(\lambda_v)_{v \in V}$: Dual variable associated with constraints (3.2) in the MP

$(u_d)_{d \in D}$: Dual variable associated with constraints (3.3) in the MP

$\tilde{c}_s$: Reduced cost of a schedule $s \in S^d$ assigned to a vehicle of depot $d \in D$

$\mathcal{CV}$: Set of all pairs of trips performed consecutively (without returning to the depot)

$\mathcal{PO}$: Set of all trips which are performed immediately after a pull-out trip before a pull-in trip

$\mathcal{PJ}$: Set of all trips which are performed immediately before a pull-in trip

$i(v)$: Index of the copy $k^{i(v)}_v \in K_v$ of the trip v selected in the Phase I solution, $\forall v \in \mathcal{PO} \cup \mathcal{PJ}$