A State Space Reduction Heuristic for Multiple Depot Vehicle Scheduling Problem

A thesis submitted to the
Graduate School of Natural and Applied Sciences

by

İsmail Sevim

in partial fulfillment for the
degree of Master of Science

in

Industrial and Systems Engineering

İSTANBUL
ŞEHİR
UNIVERSITY

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Industrial and Systems Engineering.

APPROVED BY:

Assist. Prof. Hatice Tekiner Moğulkoç (Thesis Advisor)	[illegible]
Assoc. Prof. Mehmet Güray Güler (Thesis Co-advisor)	[illegible]
Assoc. Prof. Ali Fuat Alkaya	[illegible]
Prof. Umut Rıfat Tuzkaya	[illegible]

This is to confirm that this thesis complies with all the standards set by the Graduate School of Natural and Applied Sciences of İstanbul Şehir University:

DATE OF APPROVAL: 01 June 2016

SEAL/SIGNATURE: [illegible]

Declaration of Authorship

I, İsmail Sevim, declare that this thesis titled, 'A State Space Reduction Heuristic for Multiple Depot Vehicle Scheduling Problem ' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: [illegible]

Date: 01.06.2016

"...You're only dancing on this earth for a short while... "

Cat Stevens

A State Space Reduction Heuristic for Multiple Depot Vehicle Scheduling Problem

İsmail Sevim

Abstract

Multiple Depot Vehicle Scheduling Problem (MDVSP) is the problem of assigning timetabled trips of different lines to a limited number of vehicles emanating from multiple depots. It is a component of bus transit planning process of public transportation companies and aims to prepare vehicle schedules covered by minimum number of vehicles with minimum total deadhead kilometers while satisfying trip compatibility relations.

In this thesis, two heuristic solution methodologies are devised to solve MDVSP by following the current literature. Iterative Rescheduling (IR) improves the existing heuristic method Schedule - Cluster - Reschedule (SCR) where Trips Merger (TM) is based on reducing the state space by using the outputs of IR solution.

Vehicle scheduling problem of Metrobus System of Istanbul Electricity, Tramway, and Tunnel (IETT) General Directorate is modelled as MDVSP and solved by IR and TM heuristics. It is shown that preparing vehicle schedules of the system via mathematical optimization instead of manual methods and relaxing the rule of disallowance of line change which is applied in current scheduling methodology leads to less costly vehicle schedules.

Keywords: Bus Transit Planning, Multiple Depot Vehicle Scheduling Problem, 0-1 Integer Programming, Heuristic Optimization

Çok Garajlı Araç Çizelgeleme Problemi için bir Çözüm Uzayı Küçültme Sezgiseli

İsmail Sevim

Öz

Çok Garajlı Araç Çizelgeleme Problemi (ÇGAÇP), sefer tarifelerinde yer alan tarifelerin birden fazla garajda parklayan sınırlı sayıdaki araca atanması problemidir. Seferler arasındaki uyumluluk ilişkilerini göz önüne alarak minimum sayıda araç ve toplam ölü kilometre ile karşılanabilecek araç çizelgelerinin hazırlanmasını amaçlayan problem, toplu ulaşım işletmelerinin hazırlamak durumunda oldukları şehir içi otobüs taşımacılığı planlarından biridir.

Bu tez çalısmasında, güncel literatür takip edilerek ÇGAÇP çözümünde kullanılmak üzere iki adet sezgisel metot geliştirilmiştir. Yinelemeli Çizelgeleme (Iterative Rescheduling) literatürde var olan Çizelgele - Kümele - Tekrar Çizelgele (Schedule - Cluster - Reschedule) sezgiselinin gelişmiş versiyonu iken, Sefer Birleştirici (Trips Merger) algoritmasının çalışma prensibi Yinelemeli Çizelgeleme sezgiselinin sonuçlarının kullanılarak çözüm uzayının küçültülmesine dayanmaktadır.

İstanbul Elektrik, Tramvay ve Tünel (İETT) İsletmeleri Genel Müdürlüğü'nün sorumluluğunda olan Metrobüs Sistemi'ne ait araç çizelgeleme problemi ÇGAÇP olarak modellenmiş ve Yinelemeli Çizelgeleme ve Sefer Birleştirici sezgiselleri yardımıyla çözülmüştür. Araç çizelgelerinin hazırlanmasında manuel metotlar yerine matematiksel optimizasyon tekniklerinin kullanılması ve halihazırdaki çizelgeleme sistemi için geçerli olan hat değişikliğine izin vermeme kuralının göz ardı edilmesi sayesinde daha düşük maliyetli araç çizelgelerinin elde edildiği gösterilmiştir.

Anahtar Sözcükler: Şehir İçi Otobüs Taşımacılığı Planlama, Çok Garajlı Araç Çizelgeleme Problemi, 0-1 Tamsayılı Programlama, Sezgisel Optimizasyon

To my family,

my advisors,

&

Ruken

Acknowledgments

I would like to express my deepest gratitude and appreciation to my advisor Assist. Prof. Hatice Tekiner Moğulkoç for her support and endless patience. I would also like to thank my co-advisor Assoc. Prof. Mehmet Güray Güler, the Boss, for his valuable supports and life-saving ideas. Without their priceless guidance this thesis and a continuing career in academia would not be possible.

Special thanks to my committee members, Assoc. Prof. Ali Fuat Alkaya and Prof. Dr. Umut Rıfat Tuzkaya, for their participation and precious suggestions.

I would also like to thank Zeynep Pınar Mutlu, MSc., for her supports since the time that I was only a junior student and a trainee at IETT General Directorate. She is the one who gave me the topic of the thesis. Unfortunately, she is not responsible for bus transit planning of the company anymore. I wish we had more chances to work together.

I also appreciate my dearest friend and colleague Bünyamin Cansev. He is a mathematician, an industrial engineer, economist and an amateur historian. I wish I could have a mind just like his.

I should note that three babies were born during the birth of this thesis. A boy joined the Moğulkoç family. The Boss is a father now. Lastly, Zeynep Pınar Mutlu is a mother of a pretty daughter. I wish all my little nephews a successful and happy life. I will be glad if one day I have a chance to help them.

I also thank to all my family members: My mother for her supplements, my father for his encouragement, and my sisters for their patience. Your supports during my studies have kept me stronger.

Finally, to Ruken whose name means "a smiling face". Thank you for supporting, encouraging and cheering me whenever I need you. And thank you for teaching me that the distance does not matter.

Contents

List of Figures

List of Tables

Abbreviations

AP	**A**ssignment **P**roblem
ARP	**A**ircraft **R**outing **P**roblem
BRT	**B**us **R**apid **T**ransit
BT	**B**us **T**imetabling
DSP	**D**river **S**cheduling **P**roblem
IETT	**I**stanbul **E**lectricity, **T**ramway, and **T**unnel
IR	**I**terative **R**escheduling
MCNF	**M**ulti **C**ommodity **N**etwork **F**low
MDVSP	**M**ultiple **D**epot **V**ehicle **S**cheduling **P**roblem
RND	**R**oute **N**etwork **D**esign
SCR	**S**chedule - **C**luster - **R**eschedule
SDVSP	**S**ingle **D**epot **V**ehicle **S**cheduling **P**roblem
SFCS	**S**chedule **F**irst - **C**luster **S**econd
SPP	**S**et **P**artitioning **P**roblem
TM	**T**rips **M**erger
TNDP	**T**ransport **N**etwork **D**esign **P**roblem
TSN	**T**ime **S**pace **N**etworks
VSP	**V**ehicle **S**cheduling **P**roblem

Chapter 1

Introduction

The bus planning process of bus transit companies consists of six steps: (1) network design, (2) frequencies setting, (3) timetable development, (4) vehicle scheduling, (5) driver scheduling, and (6) driver rostering. The network design and frequencies setting steps are known to be strategic level plans of such companies, where timetable development is a tactical level plan. The outputs of last three steps are needed for daily operations of bus transit companies. Therefore, they are classified as operational plans. Vehicle scheduling, main topic of this thesis, deals with assigning daily timetabled trips to available vehicles to obtain bus schedules. A Multiple Depot Vehicle Scheduling Problem (MDVSP) is a problem of assigning timetabled trips of different lines to a set of available vehicles emanating from multiple depots. It aims to minimize the number of vehicles and total pull-out, pull-in, and dead-running trip costs, while satisfying trip compatibility concerns and depot capacities. The problem is proven to be NP-Hard. If the number of depots in a MDVSP case is equal to one, then such a problem is called a Single Depot Vehicle Scheduling Problem (SDVSP) and there exist polynomial time algorithms to solve this problem.

A review of the MDVSP literature reveals that the problem is usually modelled as Multi-Commodity Network Flows (MCNF), a Set-Partitioning Problem (SPP), or Time-Space Networks (TSN). The solution methods offered for MDVSP may be grouped into three categories: (1) exact solution approaches, (2) heuristic methods, and (3) metaheuristic methods. Since the problem is NP-Hard, standard optimization software applications are not able to solve large-sized real-world MDVSP instances. Therefore, problem-specific exact solution methodologies are offered by researchers of the field. Also, many heuristic and metaheuristic methods are devised and preferred to exact solution methodologies due to the scalability property of heuristics.

Due to the NP-Hardness of the problem, finding an optimal solution for a large-sized MDVSP instance is a challenging work. In the MDVSP literature, there exist only two studies reporting optimal solutions for large-sized MDVSP instances. The main drawback of the methods offered in both studies is the methods are instance-specific. Therefore, it can be stated there is no best method to solve all large-sized MDVSP instances to proven optimality. Since there exists no exact solution methodology fit in with all MDVSP instances, heuristic methods take action to solve large-sized real-world MDVSP instances in most of the cases.

The vehicle scheduling problem of the Metrobus System, a BRT line governed by Istanbul Electricity, Tramway, and Tunnel (IETT) General Directorate, is modelled as an MDVSP and solved by two newly introduced heuristic methods, namely Iterative Rescheduling (IR) and Trips Merger (TM) in this thesis. Amounts of cost improvements obtained by this solution are reported. Timetabled trips of 7 different lines of 2014-2015 winter timetables of the Metrobus System are used as input to the MDVSP. Since there are 6,254 trips in the problem, it can be classified as a large-sized real-world MDVSP instance.

Schedulers of the Metrobus System use a manual approach to prepare bus schedules and such an approach leads to scheduling solutions far from optimal. Also, there exists a rule applied in the current scheduling approach is thought to cause sub-optimal solutions. According to this rule, a vehicle is not allowed to cover timetabled trips of different lines. Therefore, when a vehicle arrives at a terminal, it has to wait the next timetabled trip of the same line instead of starting the next timetabled trip of any other lines and this causes unnecessary waits of vehicles at terminals. The MDVSP, a mathematical optimization model, prepares bus schedules by relaxing this rule. The solution of MDVSP of the Metrobus System shows that operational costs of the system can be reduced by inserting mathematical optimization techniques into vehicle scheduling phase of Metrobus System planning.

In Chapter 2, steps of public transportation planning and bus transit planning are briefly described. The definition of MDVSP and related literature review are also topics of the chapter. Two new MDVSP heuristics are introduced in Chapter 3 and a comparison of these heuristics with two existing heuristic methods concludes the aforementioned chapter. The case study of this thesis is given in Chapter 4. Finally, Chapter 5 contains concluding remarks and future research directions.

Chapter 2

Background and Multiple Depot Vehicle Scheduling Problem

2.1 Overview

In this chapter, Multiple Depot Vehicle Scheduling Problem (MDVSP) is introduced. Related mathematical models and literature review are given for MDVSP. Steps of public transportation planning and bus transit planning processes are also discussed briefly to show where MDVSP is positioned among planning activities.

Steps of public transportation planning and corresponding definitions are listed at the beginning of Section 2. It follows with brief discussions of bus planning process: (1) transport network design problem (TNDP), (2) bus timetabling (BT), (3) vehicle scheduling problem (VSP), (4) driver scheduling and rostering. In Section 3, definition of MDVSP and related terminology are given. Aforementioned section ends with two different mathematical models for MDVSP, namely Multi-Commodity Network Flow (MCNF) and Set Partitioning Problem (SPP). Last section of this chapter is a literature survey on MDVSP.

2.2 Bus Planning Process of Public Transport Companies

Crowded cities have to plan their transportation systems cleverly and efficiently to avoid traffic congestion and to increase the prosperity and mobility of their habitants. Steps

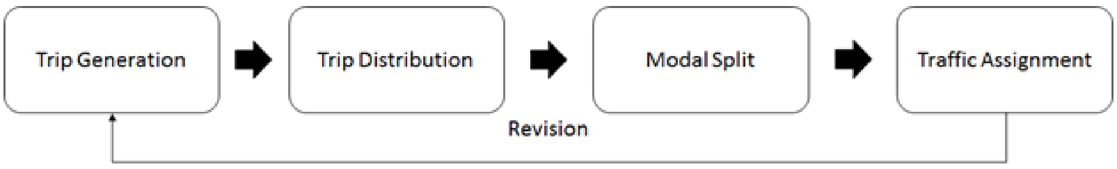

FIGURE 2.1: Steps of transportation planning

in transportation planning are: (1) trip generation, (2) trip distribution, (3) modal split, and (4) traffic assignment. Brief descriptions of each step are given below:

Trip Generation: The city is separated into zones. Planners may use the existing administrative structure such as counties or their own zoning system. Trip generation and attraction values of each zone are found by conducting polls, doing observations etc.

Trip Distribution: The ending zones of all trips generated from each zone are planned in a way satisfying number of generations and attractions of each zone defined in the trip generation phase.

Modal Split: The transport modes of trips between each zone are determined. For instance, 10% of trips between Zone-A and Zone-B are covered by private cars.

Traffic Assignment: This is the last step of transportation planning process. Each trip must be assigned to one of the existing roads. Planners usually follow Wardrop's User Equilibrium principle [?] while assigning trips to available routes. According to the principle, in a balanced traffic assignment there are no better route assignments than the assigned ones for all of the agents moving in the traffic flow.

Steps of transportation planning is given in Figure **??** [?]. Please note that, transportation planning is an iterative process.

The above planning process covers all transportation modes and activities including public or private agents and institutions. In this thesis, one of the activities of bus planning process of public transportation companies is studied. Although there are different classification for activities of bus planning process, framework of [?] is employed here. Planning activities and outputs-inputs of each activity are given in Table **??** [?].

TABLE 2.1: Bus planning activities of public transportation companies

Independent Inputs	**Planning Activity**	**Outputs**
Demand Data Supply Data Route Performance Indicators	Network Design	Route Changes New Routes Operating Strategies
Subsidy Available Buses Available Service Policies Current Patronage	Frequencies Setting	Service Frequencies
Demand by Time of Day Times for First and Last Trips Running Times	Timetable Development	Trip Departure Times Trip Arrival Times
Deadhead Times Recovery Times Schedule Constraints Cost Structure	Bus Scheduling	Bus Schedules
Driver Work Rules Run Cost Structure	Driver Scheduling	Driver Schedules

2.2.1 Transport Network Design Problem

TNDP deals with setting lines and their frequencies. The former one is also called Route Network Design (RND) [?]. In RND, planners aim to design an optimal set of stops and lines in terms of costs and passenger satisfaction, where frequency setting determines number of trips for time intervals. Public transportation companies want to spend least possible amount of resources to satisfy passenger needs where passengers demand cheap and through journeys. Minimizing number of vehicles, number of links, equivalent pollution index, or maximizing number of users, ratio of number of stops to number of links or combinations of these indicators may be chosen as performance criteria of TNDP. For a wider list of indicators readers are referred to [?].

Besides aiming suitable purposes, planners have to face with some restrictions such as (a) minimum values of frequencies for each route, (b) maximum allowed load factor, a measure of the capacity utilization, for each route, and (c) maximum number of buses available [?]. Note that restrictions of a real world case are not limited to these. A comprehensive literature review of TNDP may be found in [?].

2.2.2 Bus Timetabling

BT aims to decide the departure times of each bus trip. During this decision process, the objective is to obtain an optimal list of departure times, i.e., timetables. There may exist many factors affecting the optimality definition. For example, three different factors are discussed in [**?**]. These factors are listed as: (1) passengers may transfer to next line with short waiting times, (2) bus bunching, a situation more than one bus belong to same line arrives same station simultaneously, must be avoided, and (3) departures should be aligned evenly.

2.2.3 Vehicle Scheduling Problem

The purpose of VSP is assigning timetabled trips to an available number of vehicles in a fleet by considering compatibility relations between each trip. Assignment activity may have many restrictions. Since, a VSP variant, MDVSP is the main topic of this thesis, related literature and further discussions are given in next sections. Please, note that VSP is analogous to Aircraft Routing Problem (ARP) of aviation systems.

2.2.4 Driver Scheduling and Rostering

Driver Scheduling Problem (DSP) deals with assigning trips to anonymous drivers of a transportation company to generate daily duties. It must be ensured that each vehicle has a driver, unless it is in a depot where vehicles park after their duties.

Public transportation companies have to face challenging restrictions while scheduling its drivers. Strict government rules such as maximum amount of work without a break or starting and ending times of a workday must be satisfied [**?**]. The main purpose of driver scheduling is to cover all timetabled trips with minimum number of duties. A comprehensive overview on DSP can be found in [**?**].

Preparing daily rosters is also a compelling task for public transportation companies. The main purpose of this planning activity is to cover all generated duties with a minimum number of drivers. Minimizing total overwork is also a purpose of rostering. During rostering, planners have to follow some rules such as vacation days for each driver and maximum allowed number of working days in a row to distribute duties evenly among all drivers. A real world application of rostering can be found in [**?**].

2.2.5 A Brief Discussion on Optimization in Bus Planning Process

Public transportation companies use mathematical optimization methods as well as practical methods to plan their activities. Although planning all activities simultaneously is theoretically known to produce the most efficient plans, public transportation companies follow sequential approaches in practice. Actually, they are obliged to employ such an approach due to the computational complexity of each planning activity. For example, it is shown that RND, VSP, and DSP are NP-Hard [**? ? ?**].

Planning even a single step of a large-sized real-world public transportation case through mathematical optimization techniques is not always possible in a tolerable amount of time. Therefore, researchers decide to devise efficient methods for solving separate steps of the bus planning process. For instance, a heuristic method is devised to solve MDVSP in this thesis. However, there exist simultaneous solution approaches in the transportation literature. It should be noted none of these studies deal with all steps of large-size real-world bus planning cases.

Genetic algorithms to solve RND and frequency setting simultaneously are given in [**? ?**]. There are also solution approaches for concurrent optimization of vehicle and driver scheduling [**? ?**]. On the other hand, timetabling, vehicle scheduling, and driver scheduling problems are solved simultaneously with heuristic methods in [**?**], where a solution methodology to solve vehicle scheduling, driver scheduling, and driver rostering problems concurrently is discussed in [**?**].

2.3 Multiple Depot Vehicle Scheduling Problem

Two major components of a public transport company's resources are labor and energy. Therefore, it is possible to obtain significant cost reductions through efficient usage of these cost components for most of bus transit planning cases. Planners have to carefully prepare its driver and vehicle schedules to exploit these cost reduction opportunities.

In this thesis, the preparation of vehicle schedules via mathematical programming is discussed.

Before going into the details of MDVSP, related terminology is discussed firstly. A bus transit company has a **fleet** of **vehicles** to serve its customers. Different fleets may contain different types of vehicles. For example, a fleet may have standard buses, articulated buses, and double-decker buses where a fleet belongs to another company may consist of minibuses only. Vehicles of a fleet park and get serviced in single or multiple **depots**. Whether single or multiple, depots have maximum capacities. Let D be the set of depots, then each $d \in D$ has at most r_d vehicles within.

Each vehicle is supposed to cover a portion of **timetabled trips** of same or different **lines** that is a sequential set of **stations**. A timetabled trip is a single journey on a route of a line. Starting-ending stations, starting-ending (planned) times, and duration (planned) of a timetabled trip is defined and shared with passengers in form of **timetable** in advance. Vehicles may have to cover some **deadhead trips** that is a trip without passengers. A deadhead trip may be one of these three: (1) a **pull-out trip** between a depot and starting station of a timetabled trip, (2) **pull-in trip** between ending station of a trip and a depot, and (3) **dead-running trip** between ending station of a trip and starting station of a **compatible trip**. Let T be the set of timetabled trips, e_t be the ending time of trip $t \in T$, $s_{t'}$ be the starting time of a trip $t' \in T \backslash \{t\}$ and $\Delta_{t,t'}$ be the duration of a dead-running trip between ending station of trip t and starting station of trip t'. If $e_t + \Delta_{t,t'} \leq s_{t'}$, the it is said that trips t and t' are compatible trips and a vehicle can cover these trips in a sequence.

MDVSP is the problem of assigning $|T|$ number of timetabled trips to vehicles emanating from $|D|$ number of depots. It is solved to obtain **bus schedules**. A bus schedule is a feasible sequence of journeys starts with a pull-out trip followed by a few timetabled trips with possible dead-running trips between each consequent timetabled trip pairs and ends with a pull-in trip. Note that MDVSP is called Single Depot Vehicle Scheduling Problem (SDVSP) if $|D| = 1$.

2.3.1 Objective Function of Multiple Depot Vehicle Scheduling Problem

An optimal solution for a MDVSP case covers all timetabled trips with minimum number of vehicles and minimum total deadhead trip kilometers. However, these two objectives may conflict with each other and one of them must be prioritized. In MDVSP, minimizing number of vehicles is always the primal objective due to the high procurement, repair, and maintenance costs of vehicles.

Objective function of MDVSP is summation of deadhead trip distances which are measured in terms of kilometers in this thesis. By adding an adequately large penalty P_V to pull-out and pull-in trip distances, it is ensured that minimizing number of vehicles is prioritized.

2.3.2 Constraints of Multiple Depot Vehicle Scheduling Problem

Although different set of constraints may be incorporated into different MDVSP cases, there are four basic constraints all MDVSP cases must have: (1) Each trip must be assigned to only one vehicle, (2) each consequent trip pairs in any bus schedule must be compatible, (3) each bus schedule must start and end at same depot, and (4) it is not allowed to exceed depot capacities.

In practice, there may exist a boundary on maximum kilometers a bus can ride or maximum duration a bus can be on the road in a day. Such restrictions are called route time constraints and incorporated into several MDVSP cases [**?** **?** **?**]. Moreover, if there are more than one type of vehicle in a fleet and each timetabled trip can only be assigned to a subset of these vehicle types, then an additional set of constraints called multi-vehicle type must be added to MDVSP. In MDVSP literature, there are a number of studies take into account such constraints [**?** **?**].

2.3.3 Mathematical Modelling Approaches

In this section the MDVSP literature is discussed in terms of modelling approaches. It is found that MDVSP is modelled as Multi Commodity Network Flows, Set Partitioning Problem, and Time-Space Networks through a review of MDVSP literature. Network

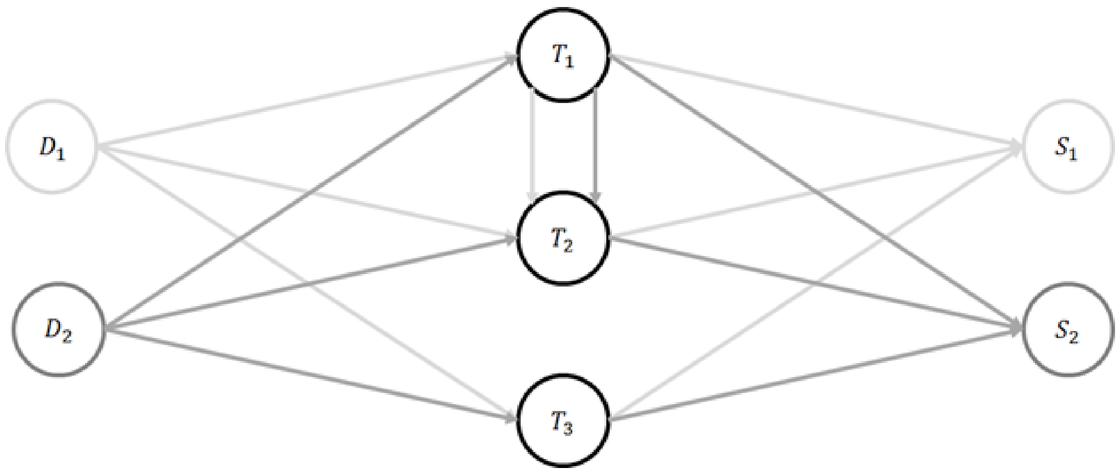

FIGURE 2.2: Network for Multi-Commodity Network Flow Model

structures and mathematical models of two studies are given as examples of modelling -one for MCFN and one for SPP. Interested readers are referred to [**?**] for details of TSN approach.

2.3.3.1 Multi-Commodity Network Flow

In [**? ? ?**] MDVSP is modelled as MCNF. Network structure and mathematical model of [**?**] are given in this section as an example of MCNF modelling.

The given mathematical model is based on flows over directed multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The set of nodes of $\mathcal{G}$ is union of three subsets: (1) set of trip nodes $\mathbf{T} = \{T_1, T_2, \ldots, T_n\}$ where n is equal to number of trips in the MDVSP case, (2) set of source nodes $\mathbf{D} = \{D_1, D_2, \ldots, D_d\}$ where d is equal to number of depots in the problem, and (3) set of sink nodes $\mathbf{S} = \{S_1, S_2, \ldots, S_d\}$. There are three different arc sets that union of them is $\mathcal{E}$: (1) pull-out arc set L that contains arcs (D_d, T_i) for $d = 1, 2, \ldots, |\mathbf{D}|$ and $i = 1, 2, \ldots, |\mathbf{T}|$, (2) deadhead arc set $\mathbf{C}$ that contains $|\mathbf{D}|$ replications of arcs (T_i, T_j) where trip$i$ and trip j are compatible trips, and (3) pull-in arc set A that contains arcs (T_i, S_d) for $i = 1, 2, \ldots, |\mathbf{T}|$ and $d = 1, 2, \ldots, |\mathbf{S}|$. Note pull-out arcs and pull-in arcs correspond to pull-out trips and pull-in trips respectively where deadhead arcs correspond to dead-running trips.

The network representation of an MDVSP case with 2 depots and 3 trips can be found in Figure **??**. It is also given that $\mathbf{C} = \{(T_1, T_2)\}$, i.e., only first and second trips are compatible. Note that the arcs of Depot 1 are drawn in color of light grey where arc of Depot 2 have color of dark grey.

A MCNF formulation based on multigraph $\mathcal{G}$ is given below. From now on, the model will be called (P). In (P), **T** takes part as transsshipment nodes where **D** and **S** are defined as source and sink nodes respectively. Production capacities of source nodes that are typical for MCNF are actually depot capacities. Since all arc capacities are equal to 1, it is ensured that at most one vehicle can cover a deadhead trip whether it is a pull-out, dead-running or pull-in trip. Therefore, aim of (P) is to obtain minimum number of chains visit all elements of **T** with the lowest cost. Model (P) is given below:

$$\min \quad \sum_d \sum_i l_{d,i} L_{d,i} + \sum_i \sum_j \sum_d c_{i,j,d} x_{i,j,d} + \sum_d \sum_i a_{i,d} A_{i,d} \tag{2.1}$$

$$s.t. \quad \sum_j L_{d,i} \leq r_d \qquad \forall d, \tag{2.2}$$

$$L_{d,i} + \sum_j x_{j,i,d} - y_{i,d} = 0 \qquad (j,i) \in C \qquad \forall i, d, \tag{2.3}$$

$$A_{i,d} + \sum_j x_{i,j,d} - y_{i,d} = 0 \qquad (i,j) \in C \qquad \forall i, d, \tag{2.4}$$

$$\sum_j A_{i,d} \leq r_d \qquad \forall d, \tag{2.5}$$

$$\sum_d y_{i,d} = 1 \qquad \forall i, \tag{2.6}$$

$$\textit{All variables are binary.} \tag{2.7}$$

Variables of model (P) are defined below:

- $L_{d,i}$: Binary variable takes a value of 1 if trip i is the first trip of a bus schedule and covered by a vehicle emanating from depot d and 0 elsewhere.
- $x_{i,j,d}$: Binary variable takes a value of 1 if a dead-running trip between two compatible trips i and j is covered by a vehicle emanating from depot d and 0 elsewhere.
- $A_{i,d}$: Binary variable takes a value of 1 if trip i is the last trip of a bus schedule and covered by a vehicle emanating from depot d and 0 elsewhere.
- $y_{i,d}$: Binary variable takes a value of 1 if trip i is covered by a vehicle emanating from depot d and 0 elsewhere.

Parameters of model (P) are as follows:

- $l_{d,i}$: Cost of a pull-out trip from depot d to the starting station of trip i.
- $c_{i,j,d}$: Cost of a dead-running trip from ending station of trip i to the starting station of trip j. Please note that the value of the parameter is independent from source depot of the vehicle covers trip j immediately after trip i.
- $a_{i,d}$: Cost of pull-in trip from ending station of trip i to depot d.
- r_d: Maximum number of vehicles emanating from depot d (depot capacities).

Objective function is a combination of all deadhead trip costs in terms of kilometers, i.e., pull-out trip distances plus dead-running trip distances plus pull-in trip distances. The objective is to minimize the total cost. In order to enforce the model to choose the vehicle schedule with minimum number of buses, the pull-out and pull-in trip distances are added adequately large penalty P_V. Constraints **??** and **??** are capacity constraints of depots where **??**-**??** are flow conservation constraints common for MCNF formulations. Constraints **??** assigns each trip to only one depot. Constraints **??** are binary constraints and assure all variables of the mathematical model take a value of 0 or 1.

2.3.3.2 Set Partitioning Problem

Once an MDVSP is modelled as MCNF, it is possible to obtain SPP equivalent of the formulation and to solve the problem with a standard [**?**] or modified [**?**] column generation approaches. Aforementioned solution strategy is shown to be able to solve different sized MDVSP cases [**?** **?** **?**]. SPP formulation of [**?**] is given in this section as an example of SPP approach.

Before giving SPP formulation, the directed multigraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined. Vertices of directed multigraph $\mathcal{V}$ is the union of two sets $\mathbf{T}$ and $\mathbf{D}$ where $\mathbf{T} = \{T_1, T_2, \ldots, T_n\}$ and $\mathbf{D} = \{D_1, D_2, \ldots, D_n\}$ where n is the number of trips and d is the number of depots in an MDVSP case. Hence $\mathcal{V} = \{T_1, T_2, \ldots, T_n, D_1, D_2, \ldots, D_n\}$. There are three arc sets in the network that union of them is $\mathbf{E}$: (1) pull-out trip arcs set $\mathbf{L}$ that have arc type of (D_k, T_i) where $k = 1, 2, \ldots, d$ and $i = 1, 2, \ldots, n$, (2) dead-running trip arcs set $\mathbf{L}$ that contains $|\mathbf{E}|$ replications of (T_i, T_j) where $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, n$, $i \neq j$ and trips i and j are compatible trips, and (3) pull-in trip arcs set $\mathbf{A}$ that have arcs in form of (T_i, D_k) for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, d$.

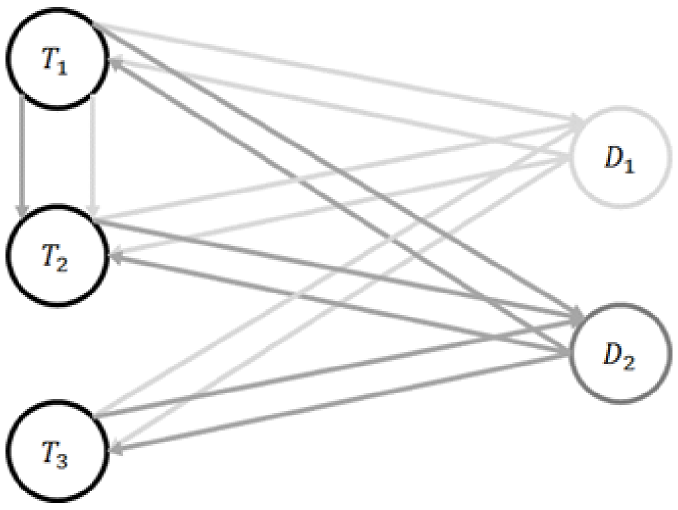

FIGURE 2.3: Network for Set Partitioning Problem

Let I be an MDVSP case with 3 timetabled trips and 2 depots where only compatibility relation exists between T_1 and T_2. Illustration of the directed multigraph $\mathcal{G}$ for I can be found in Figure **??**. Note that arcs whose tail or head node is D_1 are drawn in color of light grey where remaining arcs have color of dark grey.

SPP formulation of MDVSP based on directed multigraph $\mathcal{G}$ is given below. Let Ω be the set of all bus schedules. From now on, the formulation will be called (M). Model (M) is given below.

$$\min \quad \sum_{p\in\Omega} c_p\theta_p \tag{2.8}$$

$$s.t. \quad \sum_{p\in\Omega} a_{ip}\theta p = 1 \qquad i = 1, 2, \ldots, |\mathbf{T}| \tag{2.9}$$

$$\sum_{p\in\Omega} b_p^k\theta p \leq v^k \qquad k = 1, 2, \ldots, |\mathbf{D}| \tag{2.10}$$

$$\theta_p \in \{0, 1\} \qquad \forall p \in \Omega \tag{2.11}$$

Variables of model (M) are given below.

- θ_p: Binary variable takes a value of 1 if $p \in \Omega$ is an element of bus schedules and 0 elsewhere.

Parameters of model (M) are defined below.

- c_p: Total deadhead trip cost of a bus schedule $p \in \Omega$ i.e., cost of the pull-out trip plus cost of possible dead-running trips plus cost of the pull-in trip that $p \in \Omega$ incurs.
- a_{ip}: Binary constant takes a value of 1 if trip i is covered by $p \in \Omega$ and 0 elsewhere.
- b_p^k: Binary constant takes a value of 1 if $p \in \Omega$ is covered by a vehicle emanating from depot k.
- v^k: Maximum number of vehicles emanating from depot k (depot capacities).

Objective function of model (M) is total of costs of all bus schedules $p \in \Omega$. Constraints **??** assure each trip is covered only one bus where constraints **??** are depot capacity constraints. Finally, constraints **??** ensure variables θ_p where $p \in \Omega$ can only take a value of 0 or 1. The model aims to obtain bus schedules with minimum cost by choosing an appropriate combination of bus schedules from Ω to cover all timetabled trips $\mathbf{T}$.

Since number of all bus schedules may be enormous in practice, sometimes it is even impossible to keep all bus schedules in memory. Instead of storing all bus schedules explicitly, it is more efficient to produce them when they are necessary. Therefore, a sub-problem to obtain most beneficial bus schedules is solved and these bus schedules are added to Ω that contains a couple of bus schedules initially. Then model (M) is solved to optimality with new Ω. This iterative scheme is applied until obtaining an evidence indicates that objective value of last solved model (M) cannot be improved anymore. This framework is known as column generation and widely used in scheduling field.

In such a framework given above model (M) is called master problem. As noted before, a sub-problem is needed to add new bus schedules to Ω. Readers may find a mathematical model based on multigraph $\mathcal{G}$ prepared as a sub-problem formulation (S). This formulation is used to obtain appropriate bus schedules to add Ω. Model S is given below.

$$\min \quad \sum_{k\in\mathbf{D}}\sum_{i\in\mathbf{T}}\bar{c}_{k,i}L_{k,i} + \sum_{(i,j)\in\mathbf{C}}\sum_{k\in\mathbf{D}}\bar{c}^k_{i,j}x^k_{i,j} + \sum_{i\in\mathbf{T}}\sum_{k\in\mathbf{D}}\bar{c}_{i,k}A_{i,k} \tag{2.12}$$

$$s.t. \quad \sum_{i\in\mathbf{T}}A_{i,k} - \sum_{i\in\mathbf{T}}L_{k,i} \qquad \forall k = 1,2,\ldots,|\mathbf{D}| \tag{2.13}$$

$$\sum_{k\in\mathbf{D}}L_{k,i} + \sum_{j:(j,i)\in\mathbf{C}}x^k_{j,i} - \sum_{j:(i,j)\in\mathbf{C}}x^k_{i,j} - \sum_{k\in\mathbf{D}}A_{i,k} \tag{2.14}$$

$$\forall i = 1,2,\ldots,|\mathbf{T}|$$

$$0 \le L_{k,i} \le 1 \qquad \forall k = 1,2,\ldots,|\mathbf{D}|, \quad \forall i = 1,2,\ldots,|\mathbf{T}| \tag{2.15}$$

$$0 \le A_{i,k} \le 1 \qquad \forall k = 1,2,\ldots,|\mathbf{D}|, \quad \forall i = 1,2,\ldots,|\mathbf{T}| \tag{2.16}$$

$$0 \le x^k_{i,j} \le 1 \qquad \forall (i,j) \in \mathbf{C} \tag{2.17}$$

Variables of model (S) are given below:

- $L_{k,i}$: Amount of flow on a pull-out arc which connects depot k to trip i where $k \in \mathbf{D}$ and $i \in \mathbf{T}$.
- $x^k_{i,j}$: Amount of flow on a k^{th} replica of dead-running trip arc between trip i and trip j where $k \in \mathbf{D}$ and $(i,j) \in \mathbf{C}$.
- $A_{i,k}$: Amount of flow on a pull-in arc which connects trip i and depot k where $i \in \mathbf{T}$ and $k \in \mathbf{D}$.

Parameters of model (S) are given below.

- $\bar{c}_{k,i}$: Let β_k be the dual variable associated with constraint **??** of model (M) where $k = 1,2,\ldots,|\mathbf{D}|$ and β^*_k denote the value β_k in an optimal solution. Then, $\bar{c}_{k,i} = l_{k,i} - \beta^*_k$ where $l_{k,i}$ is pull-out trip cost of a journey connects depot k and starting station of trip i.
- $\bar{c}^k_{i,j}$: Let α_i be the dual variable associated with Constraint **??** of model (M) where $i = 1,2,\ldots,|\mathbf{T}|$ and α^*_i denote the value α_i in an optimal solution. Then, $\bar{c}^k_{i,j} = c^k_{i,j} - \alpha^*_i$ where $c^k_{i,j}$ is cost of dead-running trip between trip i and trip j.
- $\bar{c}_{i,k}$: Let β_k be the dual variable associated with constraint **??** of model (M) where $k = 1,2,\ldots,|\mathbf{D}|$ and β^*_k denote the value of β_k in an optimal solution. Then,

$\bar{c}_{i,k} = a_{i,k} - \beta_k^*$ where $a_{i,k}$ is pull-in trip cost of a journey connects ending station of trip i and depot k.

Objective function of model (S) is total of reduced costs associated with pull-out, dead-running, and pull-in trip costs. Constraints **??**-**??** are flow conservation constraints for depot nodes **D** and trip nodes **T** respectively where constraints **??**-**??** ensure flows on all arcs cannot exceed 1 unit of flow. Model (S) aims to obtain a bus schedule with minimum negative reduced cost by solving a shortest-path problem. Note, the model can be solved for each depot separately.

2.4 Literature Review

In this section, it is aimed to review solution methodologies of MDVSP. The taxonomy of methodologies is based on solution approaches. Firstly, exact solution approaches are discussed and then heuristic and metaheuristics methods are reviewed. For a taxonomy study based on modelling approaches for SDVSP and MDVSP, readers are referred to [**?**].

2.4.1 Exact Solution Approaches

The first exact solution methodology is offered in [**?**]. MDVSP is modelled as a single-commodity network model with additional subtour breaking constraints. The authors derive a lower bound by using an "additive lower bounding" procedure [**?**] and then use a branch-and-bound technique to obtain integer solutions. Another method incorporating a branch-and-bound technique is given in [**?**]. Here the authors formulate an equivalent set-partitioning model of a MCNF and solve instances with 6 depots and 300 trips to optimality with Dantzig-Wolfe decomposition. However, a further study of Dantzig-Wolfe decomposition for MDVSP [**?**] indicates such an approach is unable to solve instances with more than a thousand trips. Instead, the author discusses a delayed column generation method is able to solve real-world instances up to 2,283 timetabled trips. A set-partitioning approach to find an optimal solutions is discussed in [**?**]. A heuristic solution to the dual of the linear relaxation of set-partitioning problem is found, then this solution is used to reduce the number of variables of set-partitioning formulation. Finally, this reduced problem is solved to optimality with a general branch-and-bound algorithm.

A MCNF formulation of MDVSP, where depot assignment and scheduling are done by different set of variables, is used to obtain optimal solutions to MDVSP instances [**?**]. An equivalent SDVSP is solved, then a solution for this problem is used as a dual feasible basis for the linear relaxation of MDVSP. Subsequently, a branch-and-bound method is applied to obtain integer solutions. A MDVSP variant, MDVSP with time windows, is solved to optimality with a branch-and-price method [**?**]. At each branching node, a SPP is solved with column-generation. A best-first procedure is applied during branching to reduce the number of nodes to explore. Optimal solutions are reported for instances with 5 depots and 250 trips. A heuristic approach is also given in same study. A branch-and-bound algorithm that combines column generation, variable fixing, and cutting planes is offered in [**?**]. Method is able to solve instances with 4 depots and 500 trips. The authors also denote when an average number of trips in a column is large, then column generation works less efficiently.

In [**?**], the idea of separating trips into categories is inserted to MDVSP literature: (1) morning trips, (2) midday trips, and (3) afternoon trips. Trip compatibility constraint of MDVSP is also separated into two different categories: (1) depot compatibility and (2) street compatibility. This separation leads the authors to set a new mathematical model with a fewer number of variables compared to general multi-commodity flow network models [**?**]. However, since an additional set of constraint, route time restrictions, is added to pure MDVSP, offered method is able to solve in most instances with 400 trips. In another study, it is reported MDVSP with route time constraints with up to 600 trips are solved to optimality with a branch-and-cut algorithm incorporating heuristic procedures [**?**].

To the best of our knowledge, there are only two studies that solve real-world and large sized MDVSP instances to optimality. The authors of [**?**] use time-space networks instead of connection-based networks to model MDVSP. It is shown that time-space network models need up to 99% fewer number of variables comparing to connection-based network models when there are a few of terminals in a MDVSP case. Optimal solutions are obtained by direct usage of standard optimization software applications for instances with 5 depots and 7,068 trips.

A MCNF model may be transformed into an equivalent set-partitioning model. Then, this set-partitioning problem may be solved with a column generation procedure to obtain optimal vehicle schedules. During column generation iterations inactive columns with negative reduced costs become active. However, this activation method is not useful for large instances of MDVSP. In [?], it is offered to also activate some inactive variables with positive reduced costs. This technique is called "Lagrangean pricing" and solves real-world large-sized MDVSP instances with up to 49 depots and 24,906 trips. It should be noted that integrality gaps for these instances are almost 0%, thus a simple rounding procedure is enough to obtain integer solutions and no branch-and-bound procedure is needed.

2.4.2 Heuristic Solution Approaches

MDVSP is shown to be a NP-Hard problem [?]. This means there exists no polynomial time algorithm that finds an optimal solution for MDVSP cases. Although there are standard optimization software applications that have an ability to solve small or medium-sized MDVSP cases to optimality, many large-sized real-world cases of MDVSP are still unsolvable. Therefore, many authors prefer to use heuristic methods to overcome this problem. Even though heuristic methods do not guarantee finding optimal solutions, it is desirable to have at least a feasible schedule for most MDVSP cases. Moreover, if a heuristic finds a sub-optimal solution in short amount of time, this solution may be preferred to optimal solutions obtained in longer times.

Heuristic solution methodologies have been used to solve MDVSP since the early 1980s. A list of primal heuristics are given in [?]. One of them is called "Cluster First - Schedule Second". First, total pull-out and pull-in arc costs are calculated for each trip-depot pair. Then, each trip is assigned to a depot where total pull-in and pull-out cost is minimum. After this assignment, each SDVSP is solved to optimality. If there are infeasibilities in terms of depot capacities, then these infeasibilities are resolved heuristically. Another heuristic is called "Schedule - Cluster - Reschedule". A MDVSP is transformed into a SDVSP by choosing minimum pull-out and pull-in costs among all depots. Then, this SDVSP is solved to optimality. Generated vehicle schedules are assigned to depots by using actual pull-out and pull-in costs. After this assignment, "Cluster First - Schedule Second" heuristic is used to realize the rescheduling part.

In [?], MDVSP is modelled as capacitated multi-commodity matching problem. The authors use Lagrangian relaxation method where depot assignment constraints are relaxed. This relaxation decomposes MDVSP into separate SDVSPs. Since, SDVSP can be solved in polynomial time it is possible to obtain optimal solutions for these separate SDVSPs. There may exist some infeasibilities such as unassigned trips or trips assigned to more than one depot in solution due to the relaxed constraint set. A greedy heuristic is also given in the study to remove these infeasibilities. An MDVSP variant, MDVSP with time windows, is solved heuristically by an algorithm offered in [?]. Authors of the paper uses a column generation approach embedded in a branch-and-bound procedure. Depth-first search is used without back-tracking during branching.

In [?], two different heuristic procedure is given to solve MDVSP with route time constraints. First, pure MDVSP is solved to optimality. If there are infeasibilities in the solution in terms of route time constraints, then some variables are fixed to avoid these infeasibilities and the problem is solved again. Authors also offer a heuristic method to reduce number of variables for a given instance. It is reported some compatibility relations may be relaxed to reduce problem size. However, this relaxation procedure is not defined clearly. Another size reduction heuristic is given in [?]. Instead of solving MDVSP directly, separate SDVSPs for each depot in the problem are solved to optimality. If a trip-compatibility arc is activated in all SDVSP solutions, then this arc is fixed. Thus, problem size is reduced at amount of one. After fixing all compatibility arcs satisfying this condition, reduced problem is modelled as time-space network and solved to optimality by direct usage of standard optimization software applications. A similar size reduction methodology is discussed in [?]. Instead of fixing active compatible arcs as done in [?], authors offer to fix an inactive compatible arc if that arc is inactive in each SDVSP. Once the problem size is reduced, it is solved heuristically by using truncated column generation method which is given in [?]. Another method incorporates truncated column generation and size reduction is offered in [?].

Since, many compatibility arcs of MDVSPs are inactive in the solution, a degeneracy problem exist in many MDVSP cases. A method to resolve this problem is discussed in [?]. A variable fixing strategy based on "Schedule First - Cluster Second" heuristic [?] also given in the study. Performance of different heuristics are compared in [?]. A comparison between, a branch-and-cut, a Lagrangian, and truncated column generation heuristics is done by solving cases[1] randomly generated in a way given in [?]. It is reported the truncated column generation method is best in terms of solution quality if longer solution times are tolerable. For other heuristics methods readers are referred to [? ? ? ? ? ?].

2.4.3 Metaheuristic Solution Approaches

Finding optimal solutions for combinatorial optimization problems is mostly intractable, since this class of problems are known to be NP-Hard [?]. In practice, scientists are satisfied with sub-optimal or at least feasible solutions in some cases. If sub-optimal solutions are satisfying, metaheuristic methods can be used to solve combinatorial optimization problems. Metaheuristic methods are classified into two groups. Local search methods try to find a local optimum point in a neighborhood by searching the neighborhood starting from a solution belonging to that neighborhood, where population-based methods explore solution space to avoid getting stuck at a local optima.

In recent years, metaheuristic methods are also offered to solve MDVSP. Both local search and population-based methods are used in the literature. In [?], "block-moves" neighborhood is compared to two neighborhood structures of MDVSP literature: (1) shift, and (2) swap [?]. It is reported "block-moves" neighborhood outperforms both existing neighborhood structures. Iterated local search method incorporates "block-moves" is offered to solve MDVSP [?]. The authors asses performance of their method by solving instances used in [?].

Max-min ant system [?] is used to solve MDVSP with route time constraints [?]. Best-worst ant system [?], ant colony algorithm [?] are shown to be suitable population-based methods to solve MDVSP. On the other hand, local search methods such as large neighborhood search and tabu search [?], and variable depth local search [?] are also offered to solve MDVSP.

Although a wide range of metaheuristics solutions of MDVSPs have been studied in the literature, none of these studies deal with solving large-sized real-world instances. Therefore, the applicability of metaheuristics in practice is still questionable.

[1]Available for download at http://people.few.eur.nl/huisman/instances.htm

Chapter 3

Heuristic Algorithms for Multiple Depot Vehicle Scheduling Problem

3.1 Overview

In this chapter, four different heuristic methods for MDVSP are discussed and a comparison of these heuristics is given. Two of these heuristics, namely Iterative Rescheduling (IR) and Trips Merger (TM), are introduced first in this thesis, where the other two heuristic methods have been already studied in MDVSP literature. A comparison of solution times of TM heuristic and a standard optimization software application solution is also given to show the efficiency of TM heuristic.

Sections **??**, **??**, **??**, and **??** give details of heuristics Schedule First-Cluster Second (SFCS), Schedule-Cluster-Reschedule (SCR), IR, and TM consecutively. Comparison of heuristic methods is given in Section 6. The chapter ends with a section studying the efficiency of TM heuristic.

3.2 Schedule First - Cluster Second

The heuristic solution methodology is based on the idea of solving a SDVSP derived from a base MDVSP [**?**]. Since SDVSP can be solved in polynomial time [**?**], it is computationally efficient to obtain bus schedules by solving a SDVSP derivation instead of base MDVSP. From now on SDVSP derivation of a base MDVSP is called $SDVSP_{min}$. The derivation method of $SDVSP_{min}$ will be discussed later in this section.

The heuristic consists of two steps: (1) Solving $SDVSP_{min}$ to optimality to obtain a set of bus schedules Ω. (2) Assigning each bus schedule $p \in \Omega$ to one of the depots by considering depot capacities. The final assignments are bus schedules. These assignments are also a heuristic solution to the base MDVSP. Since the method has a heuristic nature, it is not always guaranteed the solution is optimal. In practice, there may exist time boundaries on obtaining the set of all bus schedules for each depot. Therefore, a suboptimal solution may be preferred to optimal solution which could require more time to be found.

The first step of SFCS heuristic is solving an SDVSP derived from a base MDVSP. Here, a method given in [**?**] is discussed to derive an appropriate SDVSP. For an MDVSP, let D be the set of depots, T be the set of timetabled trips, $l_{k,i}$ be the pull-out trip cost between depot k and starting station of trip i and $a_{i,k}$ be the pull-in trip cost between ending station of trip i and depot k where $k = 1, 2, \ldots, |D|$ and $i = 1, 2, \ldots, |T|$ and let $c_{i,j}$ be the cost of a dead-running trip between ending and starting stations of compatible trips i and j where $i = 1, 2, \ldots, |T|$, $j = 1, 2, \ldots, |T|$ and $i \neq j$. Note that $c_{i,j}$ values are independent from depots. Also let $\mathbf{l}_i$ be the set of all $l_{k,i}$ and $\mathbf{a}_i$ be the set of all $a_{i,k}$ for $i = 1, 2, \ldots, |D|$ and $k = 1, 2, \ldots, |D|$.

Assume there is a virtual depot D_v that has a depot capacity r equal to sum of capacities of all depots in a MDVSP case. Vehicles of the depot are supposed to cover timetabled trips of T defined above. Pull-out trip cost of a journey between D_v and starting station of trip i is $l_i = \min(\mathbf{l}_i)$ and pull-in trip cost of a journey between ending station of trip i and D_v is $a_i = \min(\mathbf{a}_i)$. These parameters r, l_i, a_i, and $c_{i,j}$ are used to construct a SDVSP model by using formulation of model (P) and this model is called $SDVSP_{min}$, i.e., SDVSP derivation of a MDVSP case.

Let Ω be the set of bus schedules obtained by solving $SDVSP_{min}$ and D be the set of all depots. Then a formulation of an assignment problem (AP) for clustering is given below. From now on the model is called model (A).

$$\min \quad \sum_{p\in\Omega}\sum_{k\in D} c_{p,k}x_{p,k} \tag{3.1}$$

$$s.t. \quad \sum_{k\in D} x_{p,k} = 1 \qquad \forall p = 1,2,\ldots,|\Omega| \tag{3.2}$$

$$\sum_{p\in\Omega} x_{p,k} \leq r_k \qquad \forall k = 1,2,\ldots,|D| \tag{3.3}$$

$$x_{p,k} \in 0,1 \qquad \forall p = 1,2,\ldots,|\Omega|, \qquad \forall k = 1,2,\ldots,|D| \tag{3.4}$$

Let Ω_k be the set of bus schedules assigned to depot k where $k = 1,2,\ldots,|D|$. Then, Schedule First - Cluster Second heuristic is outlined by Algorithm **??**. Note that $solve(m,d)$ is an operator solves a mathematical model m with input d.

Algorithm 1 Schedule First - Cluster Second

```
1: procedure
2:     $\Omega \leftarrow solve(SDVSP_{min}, T)$
3:     $\forall\Omega_k \leftarrow solve(A, \Omega)$
4:     return $\forall\Omega_k$
5: end procedure
```

3.3 Schedule - Cluster - Reschedule

The heuristic method is an improved version of SFCS heuristic. The basic idea behind the SCR heuristic is rescheduling clustered timetabled trips by solving separate SDVSPs for each depot. Since SCR heuristic uses outputs of SFCS heuristic as inputs, objective value of a SCR solution is always less than or equal to objective value of SFCS solution to same MDVSP case.

Assume that SFCS heuristic is used to solve a MDVSP case and bus schedules for each depot Ω_k are obtained. Let T_k be a subset of timetabled trips set T and contain timetabled trips covered by bus schedules Ω_k where $k = 1,2,\ldots,|D|$. Seperate SDVSPs for each depot are solved by considering subsets T_k to obtain rescheduled bus schedules.

Let $SDVSP_k$ be the SDVSP model to reschedule bus schedules $p \in \Omega_k$. If *chainbreaker* operator assigns each timetabled trip covered by bus schedules $p \in \Omega$ to a set T, then Algorithm **??** outlines SCR heuristic.

Algorithm 2 Schedule - Cluster - Reshedule

```
1: procedure
2:     $\forall\Omega_k \leftarrow$ Algorithm ??
3:     for $k \in D$ do
4:         $T_k \leftarrow chainbreaker(\Omega_k)$
5:     end for
6:     return $\forall\Omega_k$
7: end procedure
```

Interested readers are referred to [?] for another implementation of SCR.

3.4 Iterative Rescheduling

The heuristic is inspired by the well-known clustering algorithm K-means and based on the idea of using "Cluster" and "Reschedule" parts of the SCR algorithm in an iterative scheme. Before going into the details of IR heuristic, it is useful to share the steps of K-means algorithm.

Let P be a set of data points to be clustered into k number of clusters. Then, K-means algorithm aims to assign all elements of P to the clusters such that sum of squared distances between each data point and centroid of the cluster that a data point is possessed are minimized. Steps of K-means algorithm are as follows [?]:

1. Initiate with an opening partition. Repeat steps 2 and 3 until convergence, i.e., the membership of clusters is fixed.
2. Change the last partition by assigning each data point to closest cluster.
3. Find centroids of newly generated clusters.

Assume that a MDVSP case is solved by using SCR heuristic. Sets of bus schedules Ω_k is obtained for each depot k where $k = 1, 2, \ldots, |D|$. Then, define a set $\Omega = \cup_k \Omega_k$ i.e. a set whose members are all bus schedules obtained by the SCR heuristic. After this step, assign all bus schedules $p \in \Omega$ to depots by solving an AP in such a way used in the SFCS heuristic. The IR heuristic is applying these two steps iteratively over sets of bus schedules obtained by the SCR heuristic until convergence, i.e., objective function values of consequent iterations are equal to each other. Since the IR heuristic uses outputs of the SCR heuristic as inputs, objective value of an IR solution is always less than or equal to objective value of the SCR solution to the same MDVSP case.

Let *objective* be an operator returns objective value of given sets of bus schedules. Then, pseudo-code of the IR heuristic can be found in Algorithm **??**.

Algorithm 3 Iterative Rescheduling

```
1: procedure
2:     $i \leftarrow 2$
3:     $\forall\Omega_k \leftarrow Algorithm$ ??
4:     $\Phi_1 \leftarrow \cup_k\Omega_k$
5:     $r_1 \leftarrow \sum_{k\in D} objective(\Omega_k)$
6:     $\Omega \leftarrow \cup_k\Omega_k$
7:     $\forall\Omega_k \leftarrow solve((A), \Omega)$
8:     $c_1 \leftarrow \sum_{k\in D} objective(\Omega_k)$
9:     for $k \in D$ do
10:        $T_k \leftarrow chainbreaker(\Omega_k)$
11:        $\Omega_k \leftarrow solve(SDVSP_k, T_k)$
12:    end for
13:    $\Phi_i \leftarrow \cup_k\Omega_k$
14:    $r_i \leftarrow \sum_{k\in D} objective(\Omega_k)$
15:    while $r_i < r_{i-1}$ do
16:        $i \leftarrow i + 1$
17:        $\Omega \leftarrow \cup_k\Omega_k$
18:        $\forall\Omega_k \leftarrow solve((A), \Omega)$
19:        $c_i \leftarrow \sum_{k\in D} objective(\Omega_k$
20:        if $c_i = r_{i-1}$ then
21:            breakwhile
22:        else
23:            for $k \in D$ do
24:                $T_k \leftarrow chainbreaker(\Omega_k)$
25:                $\Omega_k \leftarrow solve(SDVSP_k, T_k)$
26:            end for
27:            $\Phi_i \leftarrow \cup_k\Omega_k$
28:            $r_i \leftarrow \sum_{k\in D} objective(\Omega_k)$
29:        end if
30:    end while
31:    return $\forall\Omega_k$
32:    return $\Phi$
33: end procedure
```

3.5 Trips Merger

Since MDVSP is NP-Hard and real world MDVSP cases are large sized, it is not always possible to obtain optimal solutions to most of real world instances. Different heuristic and metaheuristic methods are reported to be useful for solving such cases. There exist a couple of heuristic methods based on the idea of reducing state space heuristically and solving this reduced problem with direct usage of standard optimization software applications. Interested readers are referred to [**?** **?**] for such methods. The TM heuristic is also based on reducing the state space.

Assume the IR algorithm is run for an MDVSP case. Set of reschedules, prepared at each iteration of IR algorithm, Φ is obtained. If two timetabled trips are covered in a sequence in all of the reschedules, then this trip pair is merged, i.e. it is predetermined that two trips will be covered in a sequence. If there are more than two trips in a sequence possesses given property, then all of these trips are merged. After merging operation, obtained reduced MDVSP is solved to optimality by direct usage of standard optimization software applications.

After solving a MDVSP case with IR algorithm, underlying network of MDVSP, which will be called **based network** from now on, must be transformed into a **reduced network** in the TM framework, i.e., some nodes and arcs must be eliminated from the base network to obtain the reduced network. Elimination may get different forms with respect to trip features. There may exist three different trip features in terms of trips merging: (1) a trip may not be merged with another trip or trips, (2) a trip may be merged with another trip, and (3) a trip may be merged with more than two trips. First, trips are sorted in some manner. Then, elimination decisions are taken one by one for each trip. Let **chain** be a set of sequential trips to be merged, **first trip** be the starting trip of the chain, **last trip** be the ending trip of the chain, and **intermediate trip** be any trip between first and last trips. Following is a framework for elimination in case of each of the three forms given above.

- If a trip does not merge with any other trip or trips, then there is no need for any elimination, i.e., nodes and arcs related to the trip node stay same.
- If two trips are merged, nodes and arcs related to these two trips are eliminated from the base network. Instead, a single node has incoming arcs of first trip as its incoming arcs and outgoing arcs of last trip as its outgoing arcs is inserted to the network.

- If a chain has more than two trips, nodes and arcs related to first, last, and all intermediate trips of this chain are eliminated from the base network. Instead, a single node has incoming arcs of first trip as its incoming arcs and outgoing arcs of last trip as its outgoing arcs is inserted to the network.

Note, since each inserted arc comes from the base network, costs of these arcs also come from the base network, i.e., if cost of a pull-out arc connects depot k with starting station of trip i is equal to c in the base network, then cost of a pull-out arc of a node represents a chain whose first trip is i is also equal to c. However, since all dead-running trip arcs are eliminated from the base network if two or more trips are merged and none of these arcs are inserted to the reduced network, cost of these dead-running trips are not transferred to the reduced network. Thus, sum of costs of such dead-running trips are added to objective function of the mathematical model of the reduced network as a constant to ensure that costs of same bus schedules obtained by solving MCNF formulations based on base and reduced networks are equal.

Let *clear* be an operator deletes a variable, $set(m, c)$ be an operator prepares an appropriate SDVSP formulation m where all pull-out and pull-in trip costs are equal to 1 and all dead-running trip costs are equal to 0 with respect to compatibility matrix c, and *lowerbound* be a user defined positive integer whose maximum value is equal to $|\Omega|$[1]. Then Algorithm **??** outlines TM heuristic.

3.6 Comparison of Heuristics

In this section, comparison of heuristics SFCS, SCR, IR, and TM is given. Solutions of each heuristic method to a set of benchmark instances[2] are compared to optimal solutions obtained by GUROBI®solver. Given benchmark instances firstly used in [**?**] and generated by using a technique given in [**?**]. Properties of benchmark instances are given in Table **??**. *Code* is the code of benchmark instance given in source website. *#Depots* and *#Trips* are number of depots and number of trips in an instance respectively.*Total Vehicle Capacity* gives sum of depot capacities for an instance where *#Variables* and

[1]Default value of *lowerbound* is equal to $|\Omega|$.
[2]Instances are available at http://people.few.eur.nl/huisman/instances.htm

Algorithm 4 Trips Merger

```
1: procedure
2:     $\forall\Omega_k, \Phi \leftarrow$ Algorithm ??
3:     $clear(\forall\Omega_k)$
4:     for $k = 1, 2, \ldots, |\Phi|$ do
5:         for $(i, j) \in \Phi_i$ do
6:             if $x_{i,j} = 1$ then
7:                 $drm_k(i, j) \leftarrow 1$
8:             else
9:                 $drm_k(i, j) \leftarrow 0$
10:            end if
11:        end for
12:    end for
13:    $checkmatrix \leftarrow \sum_{k=1}^{|\Phi|} drm_k$
14:    for $(i, j) \in checkmatrix$ do
15:        if $checkmatrix(i, j) \geq lowerbound$ then
16:            $mergecandidate(i, j) \leftarrow 1$
17:        else
18:            $mergecandidate(i, j) \leftarrow 0$
19:        end if
20:    end for
21:    $SDVSP_{mc} \leftarrow set((P), mergecandidate)$
22:    $T_M \leftarrow solve(SDVSP_{mc}, T)$
23:    $\forall\Omega_k \leftarrow solve(MDVSP, T_M)$
24:    return $\forall\Omega_k$
25: end procedure
```

#Constraints are number of variables and number of constraints for model (P) prepared according to an instance.

Table 3.1: Benchmark instances

#	Code	#Depots	#Trips	Total Vehicle Capacity	#Variables	#Constraints
1	s0	4	500	233	306,620	4,508
2	s1	4	500	228	309,728	4,508
3	s2	4	500	217	313,772	4,508
4	s3	4	500	230	309,676	4,508
5	s4	4	500	216	303,868	4,508
6	s0	4	1000	447	1,225,640	9,008
7	s1	4	1000	438	1,228,572	9,008
8	s2	4	1000	407	1,222,576	9,008
9	s3	4	1000	429	1,198,052	9,008
10	s4	4	1000	429	1,211,288	9,008
11	s0	4	1500	675	2,690,272	13,508
12	s1	4	1500	656	2,769,400	13,508
13	s2	4	1500	676	2,750,656	13,508
14	s3	4	1500	648	2,763,136	13,508
15	s4	4	1500	593	2,756,936	13,508

All heuristics are coded in MATLAB®Release 2013a where GUROBI®Solver 6.5.0 takes action whenever a linear programming solution is needed. All calculations are done on a laptop with an Intel®Core™i7-4510U CPU @ 2.00 GHz processor with 6.00 GB RAM on a Microsoft®Windows®64 bit operating system. All linear programs prepared for MDVSP instances are based on MCNF formulation, i.e., each instance are modelled as (P).

All instances are solved by SFCS, SCR, IR, and TM algorithms heuristically. Instances are also solved to optimality by GUROBI®Solver where branching terminates when integrality gap is 0.01%. Objective function values of solutions for all instances obtained by each heuristic method and GUROBI®Solver are given in Table **??**. TM solutions are superior to IR solutions where IR solutions are superior to solutions of SCR and SFCS heuristics as expected in all MDVSP instances.

TABLE 3.2: Objective function values of each instance obtained by given heuristics and GUROBI®

#	**SFCS**	**SCR**	**IR**	**TM**	**GUROBI®**
1	1,298,849	1,298,303	1,297,921	1,297,101	1,289,158
2	1,251,543	1,251,282	1,250,632	1,250,181	1,241,687
3	1,297,518	1,297,188	1,294,325	1,293,523	1,283,812
4	1,267,631	1,267,116	1,266,211	1,264,910	1,258,686
5	1,325,449	1,325,389	1,325,388	1,325,017	1,317,153
6	2,545,407	2,544,500	2,542,442	2,536,601	2,516,095
7	2,432,147	2,431,306	2,429,876	2,428,467	2,413,375
8	2,463,745	2,463,362	2,462,852	2,462,265	2,452,982
9	2,502,842	2,502,711	2,501,631	2,501,014	2,490,780
10	2,526,349	2,525,541	2,524,541	2,524,285	2,519,307
11	3,859,269	3,857,582	3,852,859	3,848,377	3,830,716
12	3,568,324	3,568,001	3,567,498	3,567,179	3,559,193
13	3,672,906	3,671,434	3,665,827	3,664,764	3,649,628
14	3,432,750	3,432,089	3,430,154	3,428,791	3,406,826
15	3,597,976	3,596,176	3,592,931	3,587,251	3,567,124

CPU times of each heuristic method are given in Table **??** for each benchmark instance.

Gaps between optimum solution and heuristic solutions are calculated for each instance to show the efficiencies of each heuristic method. Aforementioned gaps are given in Table **??**[3].

It is already mentioned all pull-out and pull-in trip costs are added an adequately large penalty P_V to ensure that aim of minimizing number of vehicles is prioritized. Vehicle penalty P_V, is decided to be equal to 5,000 for all instances. Please note, since P_V is

[3]$gap_i = (heuristic\ solution_i - optimal\ solution_i)/heuristic\ solution_i$

Table 3.3: CPU times of each heuristic method for each benchmark instance

SFCS	**SCR**	**IR**	**TM**
0.58	0.70	0.89	0.98
0.40	0.51	0.61	0.75
0.44	0.56	1.00	1.43
0.47	0.58	0.80	0.89
0.50	0.55	0.59	0.69
3.94	4.39	5.25	5.75
2.58	3.23	5.34	5.56
2.55	3.14	3.66	3.67
2.41	2.93	3.79	3.91
2.56	3.20	5.06	5.76
5.08	6.93	16.92	18.81
6.17	8.46	10.81	10.58
6.36	7.83	15.73	16.14
7.71	9.42	14.19	13.5
9.22	10.44	14.89	12.74

Table 3.4: Gaps between heuristic and optimal solutions for each instance

#	**SFCS**	**SCR**	**IR**	**TM**
1	0.75%	0.70%	0.68%	0.61%
2	0.79%	0.77%	0.72%	0.68%
3	1.06%	1.03%	0.81%	0.75%
4	0.71%	0.67%	0.59%	0.49%
5	0.63%	0.62%	0.62%	0.59%
6	1.15%	1.12%	1.04%	0.81%
7	0.77%	0.74%	0.68%	0.62%
8	0.44%	0.42%	0.40%	0.38%
9	0.48%	0.48%	0.43%	0.41%
10	0.28%	0.25%	0.21%	0.20%
11	0.74%	0.70%	0.57%	0.46%
12	0.26%	0.25%	0.23%	0.22%
13	0.63%	0.59%	0.44%	0.41%
14	0.76%	0.74%	0.68%	0.64%
15	0.86%	0.81%	0.72%	0.56%

added to pull-out and pull-in trip costs for each depot, each additional vehicle increases the objective function at amount of 10,000 units. Therefore, objective function values given in Table **??** do not represent actual cost of all deadhead trips. An amount equal to number of vehicles used in a solution times 10,000 is subtracted from each objective function value to obtain actual cost of all deadhead trips. Number of vehicles used for each instance are given in Table **??** where actual deadhead trip costs for each instance are given in Table **??**.

Table 3.5: Number of vehicles used in solutions for each instance

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
#Vehicles	123	118	123	120	126	241	229	233	237	238	368	338	350	326	343

Table 3.6: Actual deadhead trip costs of each instance found by given heuristics and GUROBI®

#	**SFCS**	**SCR**	**IR**	**TM**	**GUROBI®**
1	68,849	68,303	67,921	67,101	59,158
2	71,543	71,282	70,632	70,181	61,687
3	67,518	67,188	64,325	63,523	53,812
4	67,631	67,116	66,211	64,910	58,686
5	65,449	65,389	65,388	65,017	57,153
6	135,407	134,500	132,442	126,601	106,095
7	142,147	141,306	139,876	138,467	123,375
8	133,745	133,362	132,852	132,265	122,982
9	132,842	132,711	131,631	131,014	120,780
10	146,349	145,541	144,541	144,285	139,307
11	179,269	177,582	172,859	168,377	150,716
12	188,324	188,001	187,498	187,179	179,193
13	172,906	171,434	165,827	164,764	149,628
14	172,750	172,089	170,154	168,791	146,826
15	167,976	166,176	162,931	157,251	137,124

Gaps between optimum solution and heuristic solutions for deadhead trip costs are calculated in such a way used in Table **??**. Calculated gaps for each heuristic is given in Table **??**.

Table 3.7: Gaps between optimum solution and heuristic solutions for deadhead trip costs

#	**SFCS**	**SCR**	**IR**	**TM**
1	14.08%	13.39%	12.90%	11.84%
2	13.78%	13.46%	12.66%	12.10%
3	20.30%	19.91%	16.34%	15.29%
4	13.23%	12.56%	11.37%	9.59%
5	12.68%	12.60%	12.59%	12.10%
6	21.65%	21.12%	19.89%	16.20%
7	13.21%	12.69%	11.80%	10.90%
8	8.05%	7.78%	7.43%	7.02%
9	9.08%	8.99%	8.24%	7.81%
10	4.81%	4.28%	3.62%	3.45%
11	15.93%	15.13%	12.81%	10.49%
12	4.85%	4.69%	4.43%	4.27%
13	13.46%	12.72%	9.77%	9.19%
14	15.01%	14.68%	13.71%	13.01%
15	18.37%	17.48%	15.84%	12.80%

It has already been discussed SCR improves the solution of SFCS, IR improves the solution of SCR, and TM improves the solution of IR. Therefore amounts of improvement on SFCS solutions obtained by each heuristic are given in Table **??** to compare given heuristics.

TABLE 3.8: Amounts of improvement on SFCS solutions obtained by each heuristic

#	**SCR**	**IR**	**TM**
1	0.79%	1.35%	2.54%
2	0.36%	1.27%	1.90%
3	0.49%	4.73%	5.92%
4	0.76%	2.10%	4.02%
5	0.09%	0.09%	0.66%
6	0.67%	2.19%	6.50%
7	0.59%	1.60%	2.59%
8	0.29%	0.67%	1.11%
9	0.10%	0.91%	1.38%
10	0.55%	1.24%	1.41%
11	0.94%	3.58%	6.08%
12	0.17%	0.44%	0.61%
13	0.85%	4.09%	4.71%
14	0.38%	1.50%	2.29%
15	1.07%	3.00%	6.38%

It is indicated by Table **??**, the amount of improvement on SFCS solutions are at a range of 0.61%-6.50%. On average, the SCR method improves the solution of the SFCS heuristic in amount of 0.54%, where values of same indicator for IR and TM are 1.92% and 3.21%, respectively.

As a last note to this section, it can be stated that solution quality of the TM heuristic strongly depends on the solution quality of opening heuristic, namely SFCS method. Actually, it is straightforward because TM is an improved version of SFCS. A scatter chart for solution quality of TM versus solution quality of SFCS heuristic and corresponding linear regression line is given in Figure **??**. Since R^2 is 0.905 it can be concluded the gap between the optimal solution and the TM solution strongly depends on the SFCS solution.

3.7 Efficiency of TM Heuristic

Since MDVSP is a NP-Hard problem, standard optimization software applications suffer from a lack of sufficiency to solve large real-world cases. Therefore, heuristic and

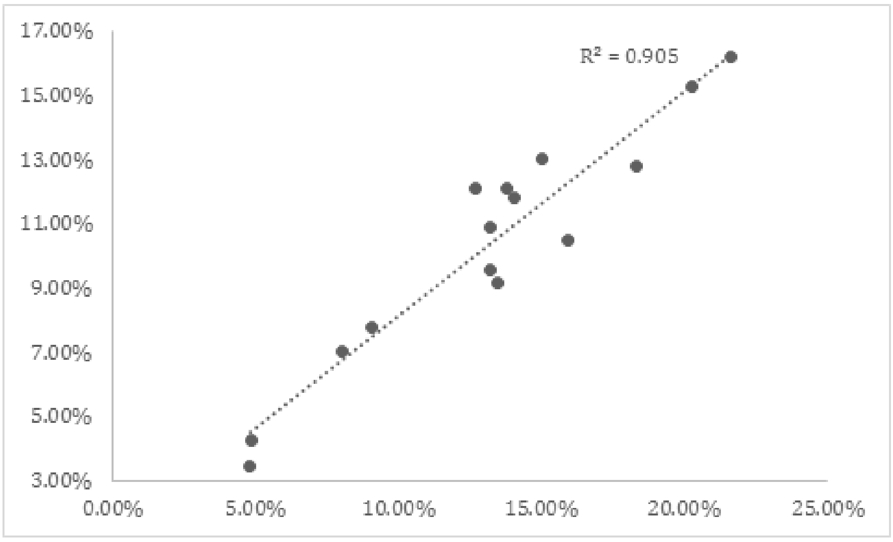

FIGURE 3.1: Solution quality of TM heuristic versus SFCS method

TABLE 3.9: Solution times of TM heuristic and GUROBI®Solver (in sec)

#	**Trips Merger**	**GUROBI®Solver**
1	0.98	39.57
2	0.75	41.55
3	1.43	30.17
4	0.89	36.8
5	0.69	274.76
6	5.75	6944.42
7	5.56	333.96
8	3.67	297.9
9	3.91	377.62
10	5.76	355.74
11	18.81	1720.29
12	10.58	1616.43
13	16.14	26787.5
14	13.5	1683.35
15	12.74	2043.72

metaheuristic methods are devised to solve this problem. Heuristic and metaheuristic methods are known to be scalable and efficiencies of such methods are independent from problem size. This fact applies for the TM heuristic too. Solution times of the TM heuristic and GUROBI®Solver are given in Table **??**.

A ratio of solution time of TM heuristic to GUROBI®Solver can be used for comparing solution times of both approaches. Values of this ratio for each instance are given in Table **??**.

On average, solution times ratio is equal to 2.34% for 500 trips instances (instances 1-5),

TABLE 3.10: Ratios of solution times of TM to GUROBI®Solver

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Ratio (%)	2.5	1.8	4.7	2.4	0.3	0.1	1.7	1.2	1	1.6	1.1	0.7	0.1	0.8	0.6

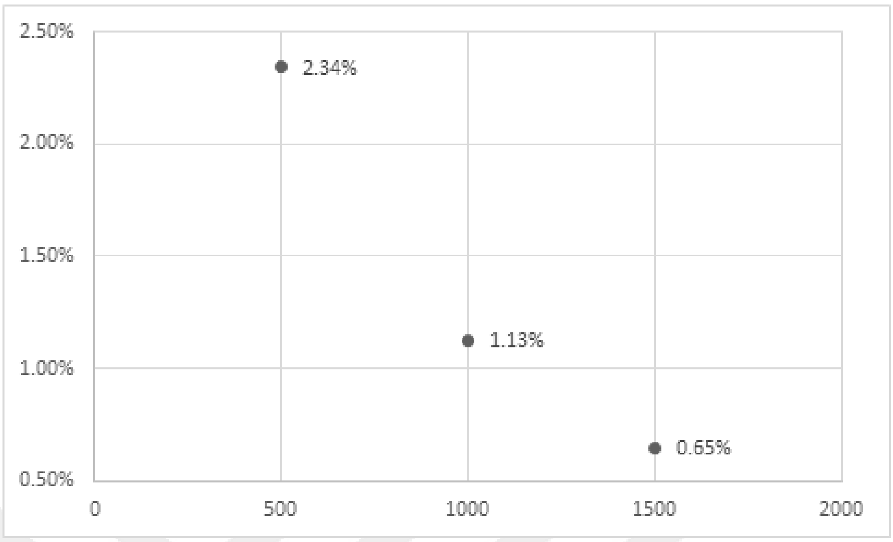

FIGURE 3.2: Average solution times ratio for each subset based on number of trips

1.13% for 1000 trip instances (instances 6-10) and 0.65% for 1500 trip instances (instances 11-15). A scatter chart for these ratios is given in Figure **??** to show scalability of TM.

Figure **??** indicates solution times ratio follows a negative exponential function. It is true because TM solves a couple of SDVSPs and GUROBI® solves a single MDVSP. Please note, there exist polynomial time algorithms to solve SDVSP where no polynomial time algorithm exists for MDVSP.

The TM heuristic has one more property makes it preferable to direct usage of standard optimization software applications such as GUROBI®. Instance 5 needs much longer CPU time than other 500-trip instances to be solved by GUROBI®. However, solution time of TM heuristic for this ill-conditioned instance does not variate significantly from average solution time of 500-trip instances. The same fact also applies for instances 6 and 13 for 1000-trip instances and 1500-trip instances respectively. This analysis indicates that solution time of the TM heuristic is not affected by compatibility relations of timetabled trips of a MDVSP case, where GUROBI® Solver is sensitive to these relations.

Chapter 4

Case Study

4.1 Overview

In this chapter the 2014-2015 winter timetables of the Metrobus System of the IETT General Directorate are assigned to vehicles of the system by solving MDVSP by the TM heuristic. Section **??** introduces the Metrobus System and the data to be used in the case study where SFCS, IR, and TM solutions to the vehicle scheduling problem of the Metrobus System are given in Section **??**. Solution times and qualities of each method are also discussed in Section **??**.

4.2 Metrobus System

The Metrobus System is a Bus Rapid Transit (BRT) line governed by the IETT General Directorate and serves the people of Istanbul. In an average day 800,000 passengers are served by the system. It covers 8.27% of daily trips of commuters among all public transportation modes of the city, e.g. minibuses, taxis, ferries, etc.

Istanbul is known to be have the third-most congested traffic after Mexico City and Bangkok and commuters suffer from significant amounts of lost time in traffic jams. The Metrobus System, as a BRT line, helping the people of Istanbul to overcome this problem by doing daily trips on 7 different lines. List of these lines is given in Table **??**.

TABLE 4.1: List of lines of Metrobus System

Line	**Terminal 1**	**Terminal 2**
34AS	Avcilar	Sogutlucesme
34	Avcilar	Zincirlikuyu
34BZ	Beylikduzu	Zincirlikuyu
34C	Beylikduzu	Cevizlibag
34G	Beylikduzu	Sogutlucesme
34Z	Zincirlikuyu	Sogutlucesme
34U	Uzuncayir	Zincirlikuyu

TABLE 4.2: Depots serve Metrobus System and their properties

Depot	**Area (m2)**	**Closed Area (m2)**	**Capacity (#Vehicles)**
Ikitelli	192,000	28,000	74
Edirnekapi	60,000	6,720	237
Hasanpasa	37,000	4,000	120
Anadolu	58,200	10,000	65
Total			**496**

TABLE 4.3: Number of daily trips of all lines of Metrobus System

Line	**Total Number of Trips** $(1 \rightarrow 2)$	**Total Number of Trips** $(2 \rightarrow 1)$
34AS	778	779
34	322	328
34BZ	887	896
34C	359	350
34G	38	37
34Z	738	583
34U	159	-
Total	**3,281**	**2,973**

Timetabled trips of the Metrobus System are covered by vehicles which are allowed to cover any of the trips, park and get repair-maintenance in four different depots spread over Istanbul. These depots and depot properties are given in Table **??**.

In this study, the 2014-2015 winter timetables which have maximum number of trips over all timetables of Metrobus System are used. The number of daily trips of all lines are given in Table **??**. In Table **??**, $(1 \rightarrow 2)$ shows trips starting from terminal 1 and ends at terminal 2 where $(2 \rightarrow 1)$ represents trips starting from terminal 1 and ends at terminal 2. Trip frequencies of all lines are given in Appendix A.

Number of daily trips of all lines add up to 6,254. In current practice, these timetabled trips are covered by 496 vehicles emanating from 4 different depots spread over Istanbul. This study aims to reduce the number of vehicles to cover such a timetable. Also, the total deadhead kilometers covered by these vehicles is aimed to be minimized by mathematical optimization techniques.

In the current scheduling approach of the Metrobus System, schedulers use a spreadsheet for scheduling purposes and they do the scheduling manually. Therefore, there is a strong suspicion that the vehicle schedules prepared by using such a manual method are far from optimum. This fact indicates there are cost improvement opportunities on scheduling via mathematical optimization. Also, there exists a rule applied on vehicle scheduling that decreases the degree of efficient usage of the fleet. According to this rule, a single vehicle cannot cover timetabled trips of different lines on the same day, i.e., when a vehicle comes to a terminal, it must wait for the starting time of the nearest timetabled trip of the same line instead of starting the nearest timetabled trip of any of the other lines. This fact causes unnecessary waiting of vehicles in terminals. In MDVSP, there is no such rule. Actually this rule may be treated as a side constraint added to the MDVSP. Therefore, it is supposed to have further cost improvements by solving the problem by modelling it as a MDVSP.

4.3 Computational Results

In this study, vehicle scheduling problem of the Metrobus System is modelled as a MD-VSP and solved by the TM heuristic introduced earlier in this thesis. A set of timetabled trips T contains timetabled trips of the 2014-2015 winter timetables. From now on, **the instance** refers to the aforementioned timetables. Please note all computations are done in a laptop with an Intel® Core™ i7-4510U CPU @ 2.00 GHz processor with 6.00 GB RAM on a Microsoft® Windows® 64 bit operating system except single-depot problem whose properties are given in Table **??**. A laptop with an Intel® Core™ i5-3210M CPU @ 2.50 GHz processor with 8.00 GB RAM on a Microsoft® Windows® 64 bit operating system is used when the exception occurs. All heuristic methods are coded in MATLAB R2013a environment and GUROBI® Solver is used whenever a linear programming solution is needed.

The instance is modelled as MDVSP with MCNF formulation, i.e., model (P). Properties of the model are given in Table **??**.

Table 4.4: Model size of the multiple-depot formulation

Model	#Trips	#Depots	#Compatibility Arcs	#Variables	#Constraints
Multi-Depot	6254	4	16,107,698	64,505,840	75,056

GUROBI® Solver is unable to solve the multi-depot model and returns the error *out of memory.* This is expected because of NP-Hardness and size of the problem. Since the direct solution approach is unsuccessful, it is concluded that a MDVSP-specific method of the existing literature or a newly introduced method is necessary. First, the instance is solved by the SFCS heuristic to show the feasibility of the problem, i.e., are there adequate number of vehicles to cover timetabled trips of the instance? The SFCS heuristic needs a SDVSP formulation of the instance. Problem size of such a formulation is given in Table **??**.

Table 4.5: Model size of the single-depot formulation

Model	#Trips	#Depots	#Compatibility Arcs	#Variables	#Constraints
Single-Depot	6254	1	16,107,698	16,126,460	18,764

GUROBI® Solver is able to solve such a formulation to optimality in 5,070.72 seconds. Solution indicates that timetabled trips of the 2014-2015 winter timetables may be covered by a fleet consisting of 476 vehicles, instead of the current fleet which has 496 vehicles. After solving the single-depot formulation, the SFCS heuristic assigns each bus schedule to available vehicles emanating from multiple-depots. The assignment is obtained by solving model (A). Problem size of the aforementioned model is given in Table **??**.

The problem is solved to optimality in less than 2 seconds. Result of this assignment actually is a solution to the instance. It is already found by the single-depot model that all timetabled trips may be covered by 476 vehicles. According to the solution of the assignment model, it is found the total deadhead kilometers incur in prepared bus schedules is equal to 16,618.

Once the feasibility of the problem is shown by solving the instance with the SFCS heuristic, it is the IR heuristic's turn. A cluster and a reschedule are generated in each iteration of the IR heuristic and the method terminates if objective value is same for two

Table 4.6: Model size of the assignment formulation

Model	#Bus Schedules	#Depots	#Variables	#Constraints
Assignment	476	4	1,904	480

Table 4.7: IR iterations

Iteration	**Phase**	**Objective Function Value**
1	Cluster	9,536,617.50
1	Reschedule	9,536,500.60
2	Cluster	9,535,909.90
2	Reschedule	9,535,887.20
3	Cluster	9,535,511.70
3	Reschedule	9,535,452.50
4	Cluster	9,535,215.70
4	Reschedule	9,535,169.30
5	Cluster	9,535,069.70
5	Reschedule	9,535,069.70

Table 4.8: IR iterations without vehicle penalties

Iteration	**Phase**	**Total Deadhead Kilometers**
1	Cluster	16,617.50
1	Reschedule	16,500.60
2	Cluster	15,909.90
2	Reschedule	15,887.20
3	Cluster	15,511.70
3	Reschedule	15,452.50
4	Cluster	15,215.70
4	Reschedule	15,169.30
5	Cluster	15,069.70
5	Reschedule	15,069.70

consecutive reschedule and cluster or vice versa. The IR heuristic iterates 5 times until the termination criteria is satisfied when solving the instance. The method needs 12,804 seconds to terminate. Changes in objective function value through iterations of the IR method are given in Table **??**. Please note, objective function value of cluster phase of the first iteration is actually the solution of SFCS.

Since an adequate penalty $P_V = 10,000$ is added to each pull-out and pull-in arc to prioritize the aim of minimizing number of vehicles, objective function values given in Table **??** does not represent actual deadhead kilometers. Actual deadhead kilometers found at each iteration of IR are given in Table **??**.

Improvements through IR iterations are given in Figure **??**.

Vehicle schedules obtained at each iteration of IR heuristic are used for reducing the problem size of the instance in TM framework. Since the IR heuristic terminates at the fifth iteration there are five different vehicle schedules each obtained at one iteration of IR.

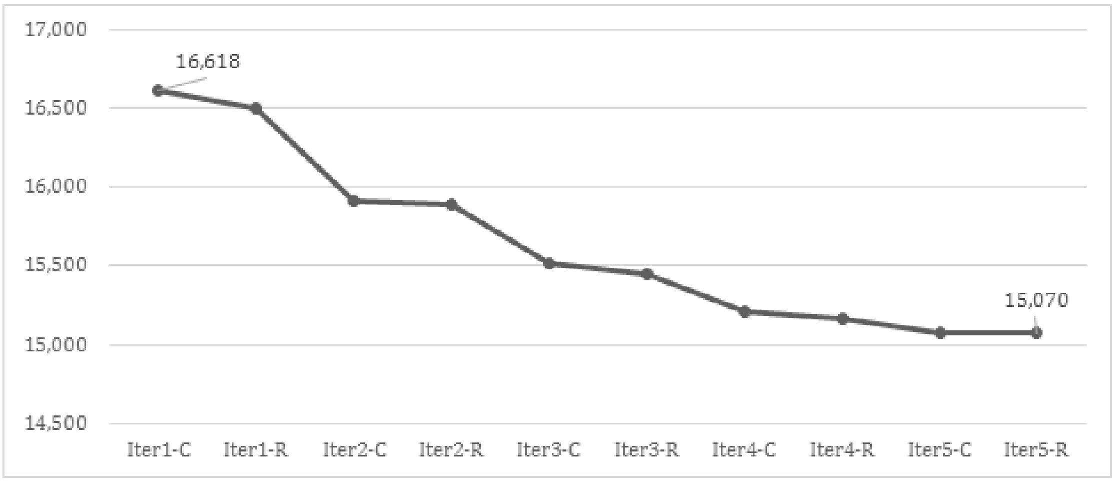

FIGURE 4.1: IR iterations without vehicle penalties

If TM uses these vehicle schedules to reduce the problem size with different *lowerbound* values, five different reduced problems are obtained. Properties of these reduced problems are given in Table **??**.

TABLE 4.9: Properties of the reduced problems

lowerbound	#Compatibility Arcs	#Variables	#Constraints	#Merged Trips	Amount of Reduction
1	0	0	0	476	92.39%
2	1,611,189	6,471,132	26,384	2198	64.85%
3	5,460,921	21,888,816	45,140	3761	39.86%
4	9,883,419	39,593,172	59,504	4958	20.72%
5	12,958,014	51,899,676	67,628	5635	9.90%

Columns *#Variables* and *#Constraints* gives the number of variables and number of constraints in a MDVSP formulation based on model (P) with 4 depots, respectively. Please note, when *lowerbound* $= 1$, solving the reduced problem as a MDVSP is equivalent to solving an assignment problem. Since there are no compatibility arcs in such a reduced problem, none of the merged trips is an element of a chain at the final solution of MDVSP. Therefore, this reduction is trivial.

Each of the reduced problems are modelled as MDVSP and tried to be solved to optimality by direct usage of GUROBI® Solver as prescribed in TM framework. If *lowerbound* parameter of TM heuristic is larger than 2, GUROBI® returns the error *out of memory.* Since TM solution is equal to IR solution when *lowerbound* is equal to 1, it is concluded that the only optimal solution is obtained for the problem where *lowerbound* is equal to 2.

Out of all reduced problems only the problem corresponding to *lowerbound* $= 2$ is solved to optimality. GUROBI® Solver needs 7,160.52 seconds to solve the aforementioned

Table 4.10: Depot assignments

Depot	**Depot Capacity**	**#Vehicles Used**	**Utilization Ratio**
Ikitelli	74	74	100%
Edirnekapi	237	237	100%
Anadolu	65	45	69.23%
Hasanpasa	80	80	100%

problem. Since first step of TM, the IR heuristic needs 12,804.04 seconds, total time to obtain a solution for the instance by the TM method is equal to 19,964.56 seconds.

Objective function value of the MDVSP prepared for the instance and solved by the TM heuristic is equal to 9,533,136. Therefore, obtained vehicle schedules incur 13,136 kilometers of deadhead trips where 13.08% of this amount is caused by dead-running trips, i.e., kilometers of dead-running trips add up to 1,718.40. Please note that, number of vehicles found by TM is also equal to 476. Depot assignments of these vehicles are summarized in Table **??**.

Summary stats of vehicle schedules by depots are given in Table **??**. *#Trips (Avg.)* gives average number of trips covered by the vehicle schedules where *#Dead-Running Trips (Avg.)* gives average number of dead-running trips whose costs are nonzero and covered by these schedules. *#Line Change (Avg.)* column gives average number of line change occurrence in vehicle schedules. Please note that the value of the stat *#Line Change (Avg.)* is equal to zero in bus schedules prepared by current scheduling approach. Finally, *Total Dead-Running (km)* column shows the total dead-running trip kilometers covered by the vehicle schedules.

Table 4.11: Summary stats of vehicle schedules by depots

Depot	**#Trips (Avg.)**	**#Dead-Running Trips (Avg.)**	**#Line Change (Avg.)**	**Total Dead-Running (km)**
Ikitelli	16.16	0.61	8.07	641.5
Edirnekapi	12.08	0.29	5.62	343.4
Anadolu	12.69	0.47	6.6	253.4
Hasanpasa	13.54	0.46	6.28	480.1

Chapter 5

Conclusion and Future Research

In this study, the vehicle scheduling problem of the Metrobus System is modelled as MDVSP and the model is solved separately by two newly introduced heuristic methods. It is reported that timetabled trips of 7 different lines may be covered by a fleet consists of 476 vehicles instead of the current fleet which has 496 vehicles. This result indicates that the current size of the fleet may be reduced by 4.03% through solving the vehicle scheduling problem of the Metrobus System with mathematical optimization methods. It is also found the total deadhead kilometers to cover the timetabled trips by 476 vehicles is equal to 13,136. Please note, line change is allowed in the MDVSP solution.

Two heuristics, namely IR and TM, are introduced in this thesis. The efficiency of these heuristics is studied by solving 15 different benchmark cases. It is found the IR heuristic improves the solution of the widely-used SFCS heuristic by 1.92%, where same indicator is equal to 3.21% for the TM heuristic. This is true because IR is the improved version of SFCS and TM is the improved version of IR. Interestingly, the IR heuristic improves the SFCS solution by 9.32% and the TM heuristic improves the by 20.95% when the vehicle scheduling problem of Metrobus System is solved. This is probably caused by the fact that in real-world problems most of the dead-running trip costs are equal to 0, which is not true for the benchmark instances. Future research is needed to show the correctness of this idea.

All of the benchmarks instances are solved to optimality by GUROBI® Solver to show the gaps between optimum and heuristic solutions. According to the results the average gap between the IR solution and optimal solution is equal to 11.56%, where the same indicator is equal to 10.40% for the TM heuristic. Although these heuristics improve the solution of the SFCS heuristic, the average gaps indicate more efficient heuristics can be devised for MDVSP.

In this study, MDVSP is solved separately as a single step of the bus planning process of bus transit companies. It is noted in the literature that solving each planning step separately leads sub-optimal solutions. Therefore, solving the vehicle scheduling problem of the Metrobus System simultaneously with driver scheduling and rostering problems may lead to reductions in the overall cost of each step. This issue may also be reserved as an item of future research.

Appendix A

Freqeuncies of All Lines of Metrobus System

Table A.1: Trip frequencies of line 34AS by terminals

Interval	**Avcilar**	**Sogutlucesme**
05:01 - 06:00	11	8
06:01 - 07:00	40	37
07:00 - 08:00	51	49
08:01 - 09:00	49	52
09:01 - 10:00	50	53
10:01 - 11:00	47	48
11:01 - 12:00	47	47
12:01 - 13:00	48	47
13:01 - 14:00	42	48
14:01 - 15:00	38	37
15:01 - 16:00	36	37
16:01 - 17:00	43	39
17:01 - 18:00	45	45
18:01 - 19:00	48	48
19:01 - 20:00	48	48
20:01 - 21:00	45	41
21:00 - 22:00	46	43
22:00 - 23:00	25	26
23:01 - 00:00	11	13
00:01 - 01:00	6	11
01:01 - 02:00	2	2
02:01 - 03:00	-	-
03:01 - 04:00	-	-
04:01 - 05:00	-	-

Table A.2: Trip frequencies of line 34 by terminals

Interval	**Avcilar**	**Zincirlikuyu**
05:01 - 06:00	6	4
06:01 - 07:00	23	17
07:00 - 08:00	41	29
08:01 - 09:00	26	40
09:01 - 10:00	26	27
10:01 - 11:00	14	20
11:01 - 12:00	11	14
12:01 - 13:00	14	10
13:01 - 14:00	9	10
14:01 - 15:00	7	10
15:01 - 16:00	26	7
16:01 - 17:00	16	31
17:01 - 18:00	37	17
18:01 - 19:00	20	38
19:01 - 20:00	15	19
20:01 - 21:00	7	15
21:00 - 22:00	12	8
22:00 - 23:00	8	7
23:01 - 00:00	2	2
00:01 - 01:00	2	1
01:01 - 02:00	-	2
02:01 - 03:00	-	-
03:01 - 04:00	-	-
04:01 - 05:00	-	-

TABLE A.3: Trip frequencies of line 34BZ by terminals

Interval	**Beylikduzu**	**Zincirlikuyu**
05:01 - 06:00	12	6
06:01 - 07:00	49	20
07:00 - 08:00	60	53
08:01 - 09:00	53	55
09:01 - 10:00	55	63
10:01 - 11:00	55	56
11:01 - 12:00	54	54
12:01 - 13:00	53	56
13:01 - 14:00	48	56
14:01 - 15:00	48	50
15:01 - 16:00	49	49
16:01 - 17:00	53	47
17:01 - 18:00	55	50
18:01 - 19:00	55	60
19:01 - 20:00	57	54
20:01 - 21:00	52	60
21:00 - 22:00	52	48
22:00 - 23:00	17	29
23:01 - 00:00	10	18
00:01 - 01:00	-	11
01:01 - 02:00	-	1
02:01 - 03:00	-	-
03:01 - 04:00	-	-
04:01 - 05:00	-	-

Table A.4: Trip frequencies of line 34U by terminals

Interval	**Uzuncayir**	**Zincirlikuyu**
05:01 - 06:00	-	-
06:01 - 07:00	12	-
07:00 - 08:00	54	-
08:01 - 09:00	54	-
09:01 - 10:00	33	-
10:01 - 11:00	6	-
11:01 - 12:00	-	-
12:01 - 13:00	-	-
13:01 - 14:00	-	-
14:01 - 15:00	-	-
15:01 - 16:00	-	-
16:01 - 17:00	-	-
17:01 - 18:00	-	-
18:01 - 19:00	-	-
19:01 - 20:00	-	-
20:01 - 21:00	-	-
21:00 - 22:00	-	-
22:00 - 23:00	-	-
23:01 - 00:00	-	-
00:01 - 01:00	-	-
01:01 - 02:00	-	-
02:01 - 03:00	-	-
03:01 - 04:00	-	-
04:01 - 05:00	-	-

Table A.5: Trip frequencies of line 34C by terminals

Interval	**Beylikduzu**	**Cevizlibag**
05:01 - 06:00	8	25
06:01 - 07:00	23	59
07:00 - 08:00	46	35
08:01 - 09:00	20	42
09:01 - 10:00	37	9
10:01 - 11:00	1	3
11:01 - 12:00	-	-
12:01 - 13:00	1	-
13:01 - 14:00	2	-
14:01 - 15:00	-	-
15:01 - 16:00	-	4
16:01 - 17:00	18	33
17:01 - 18:00	40	45
18:01 - 19:00	41	39
19:01 - 20:00	44	36
20:01 - 21:00	23	12
21:00 - 22:00	6	6
22:00 - 23:00	21	-
23:01 - 00:00	26	-
00:01 - 01:00	-	-
01:01 - 02:00	1	-
02:01 - 03:00	-	-
03:01 - 04:00	-	-
04:01 - 05:00	1	2

Table A.6: Trip frequencies of line 34G by terminals

Interval	**Beylikduzu**	**Sogutlucesme**
05:01 - 06:00	-	-
06:01 - 07:00	-	-
07:00 - 08:00	-	-
08:01 - 09:00	-	-
09:01 - 10:00	-	-
10:01 - 11:00	-	-
11:01 - 12:00	-	-
12:01 - 13:00	-	-
13:01 - 14:00	-	-
14:01 - 15:00	-	-
15:01 - 16:00	-	-
16:01 - 17:00	-	-
17:01 - 18:00	-	-
18:01 - 19:00	-	-
19:01 - 20:00	-	-
20:01 - 21:00	-	-
21:00 - 22:00	4	3
22:00 - 23:00	5	5
23:01 - 00:00	5	5
00:01 - 01:00	5	5
01:01 - 02:00	5	5
02:01 - 03:00	5	5
03:01 - 04:00	5	6
04:01 - 05:00	4	3

Table A.7: Trip frequencies of line 34Z by terminals

Interval	**Zincirlikuyu**	**Sogutlucesme**
05:01 - 06:00	4	2
06:01 - 07:00	28	20
07:00 - 08:00	77	34
08:01 - 09:00	91	39
09:01 - 10:00	68	41
10:01 - 11:00	30	21
11:01 - 12:00	22	22
12:01 - 13:00	25	24
13:01 - 14:00	22	23
14:01 - 15:00	23	22
15:01 - 16:00	23	26
16:01 - 17:00	59	48
17:01 - 18:00	62	68
18:01 - 19:00	65	70
19:01 - 20:00	62	63
20:01 - 21:00	36	37
21:00 - 22:00	25	19
22:00 - 23:00	15	3
23:01 - 00:00	1	1
00:01 - 01:00	-	-
01:01 - 02:00	-	-
02:01 - 03:00	-	-
03:01 - 04:00	-	-
04:01 - 05:00	-	-

Bibliography

[] J. G. Wardrop. Some theoretical aspects of road traffic research. In *Inst Civil Engineers Proc London/UK/*, 1900.

[] H. Guler. Model to estimate trip distribution: Case study of the marmaray project in turkey. *Journal of Transportation Engineering*, 140(11):05014006, 2014.

[] A. Ceder and N. H. M. Wilson. Bus network design. *Transportation Research Part B: Methodological*, 20(4):331–344, 1986.

[] S. B. Pattnaik, S. Mohan, and V. M. Tom. Urban bus transit route network design using genetic algorithm. *Journal of Transportation Engineering*, 124(4):368–375, 1998.

[] M. Bielli, M. Gastaldi, and P. Carotenuto. Multicriteria evaluation model of public transport networks. In *Advanced Methods in Transportation Analysis*, pages 135–156. Springer, 1996.

[] M. H. Baaj and H. S. Mahmassani. Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research Part C: Emerging Technologies*, 3(1):31–50, 1995.

[] K. Kepaptsoglou and M. Karlaftis. Transit route network design problem: review. *Journal of Transportation Engineering*, 135(8):491–505, 2009.

[] O. J. Ibarra-Rojas and Y. A. Rios-Solis. Synchronization of bus timetabling. *Transportation Research Part B: Methodological*, 46(5):599–614, 2012.

[] A. Wren and D. O. Wren. A genetic algorithm for public transport driver scheduling. *Computers & Operations Research*, 22(1):101–110, 1995.

[] A. Wren and J. M. Rousseau. Bus driver scheduling–an overview. In *Computer-Aided Transit Scheduling*, pages 173–187. Springer, 1995.

[] K. Nurmi, J. Kyngäs, and G. Post. Driver rostering for bus transit companies. *Engineering Letters*, 19(2):125–132, 2011.

[] Y. Israeli and A. Ceder. Transit route design using scheduling and multiobjective programming techniques. In *Computer-Aided Transit Scheduling*, pages 56–75. Springer, 1995.

[] A. A. Bertossi, P. Carraresi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, 17(3):271–281, 1987.

[] J. Li and R. S. K. Kwan. A fuzzy genetic algorithm for driver scheduling. *European Journal of Operational Research*, 147(2):334–344, 2003.

[] S. Ngamchai and D. J. Lovell. Optimal time transfer in bus transit route network design using a genetic algorithm. *Journal of Transportation Engineering*, 129(5): 510–521, 2003.

[] M. Ball, L. Bodin, and R. Dial. A matching based heuristic for scheduling mass transit crews and vehicles. *Transportation Science*, 17(1):4–31, 1983.

[] K. Haase, G. Desaulniers, and J. Desrosiers. Simultaneous vehicle and crew scheduling in urban mass transit systems. *Transportation Science*, 35(3):286–303, 2001.

[] M. M. Rodrigues, C. C. de Souza, and A. V. Moura. Vehicle and crew scheduling for urban bus lines. *European Journal of Operational Research*, 170(3):844–862, 2006.

[] M. Mesquita, M. Moz, A. Paias, and M. Pato. A decomposition approach for the integrated vehicle-crew-roster problem with days-off pattern. *European Journal of Operational Research*, 229(2):318–331, 2013.

[] A. Ceder and H. I. Stern. Deficit function bus scheduling with deadheading trip insertions for fleet size reduction. *Transportation Science*, 15(4):338–363, 1981.

[] A. Haghani and M. Banihashemi. Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints. *Transportation Research Part A: Policy and Practice*, 36(4):309–333, 2002.

[] M. Wei, W. Jin, W. Fu, and X. N. Hao. Improved ant colony algorithm for multi-depot bus scheduling problem with route time constraints. In *Intelligent Control and Automation (WCICA), 2010 8th World Congress on*, pages 4050–4053. IEEE, 2010.

[] A. Ceder. Optimal multi-vehicle type transit timetabling and vehicle scheduling. *Procedia-Social and Behavioral Sciences*, 20:19–30, 2011.

[] J. A. Ramos, L. P. Reis, and D. Pedrosa. Solving heterogeneous fleet multiple depot vehicle scheduling problem as an asymmetric traveling salesman problem. In *Portuguese Conference on Artificial Intelligence*, pages 98–109. Springer, 2011.

[] N. Kliewer, T. Mellouli, and L. Suhl. A time–space network based exact optimization model for multi-depot bus scheduling. *European Journal of Operational Research*, 175(3):1616–1627, 2006.

[] A. Löbel. Vehicle scheduling in public transit and lagrangean pricing. *Management Science*, 44(12-part-1):1637–1649, 1998.

[] M. A. Forbes, J. N. Holt, and A. M. Watts. An exact algorithm for multiple depot bus scheduling. *European Journal of Operational Research*, 72(1):115–124, 1994.

[] C. C. Ribeiro and F. Soumis. A column generation approach to the multiple-depot vehicle scheduling problem. *Operations Research*, 42(1):41–52, 1994.

[] A. S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1):17–30, 2009.

[] L. Bianco, A. Mingozzi, and S. Ricciardelli. A set partitioning approach to the multiple depot vehicle scheduling problem. *Optimization Methods and Software*, 3 (1-3):163–194, 1994.

[] A. Oukil, H. B. Amor, J. Desrosiers, and H. El Gueddari. Stabilized column generation for highly degenerate multiple-depot vehicle scheduling problems. *Computers & Operations Research*, 34(3):817–834, 2007.

[] P. C. Guedes, W. P. Lopes, L. R. Rohde, and D. Borenstein. Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem. *Optimization Letters*, pages 1–13, 2015.

[] A. Hadjar, O. Marcotte, and F. Soumis. A branch-and-cut algorithm for the multiple depot vehicle scheduling problem. *Operations Research*, 54(1):130–149, 2006.

[] S. Bunte and N. Kliewer. An overview on vehicle scheduling models. *Public Transport*, 1(4):299–317, 2009.

[] G. Carpaneto, M. Dell'Amico, M. Fischetti, and P. Toth. A branch and bound algorithm for the multiple depot vehicle scheduling problem. *Networks*, 19(5):531–548, 1989.

[] M. Fischetti and P. Toth. An additive bounding procedure for combinatorial optimization problems. *Operations Research*, 37(2):319–328, 1989.

[] A. Löbel. Experiments with a dantzig-wolfe decomposition for multiple-depot vehicle scheduling problems. 1997.

[] G. Desaulniers, J. Lavigne, and F. Soumis. Multi-depot vehicle scheduling problems with time windows and waiting costs. *European Journal of Operational Research*, 111(3):479–494, 1998.

[] M. Fischetti, A. Lodi, S. Martello, and P. Toth. A polyhedral approach to simplified crew scheduling and vehicle scheduling problems. *Management Science*, 47(6):833–850, 2001.

[] A. Löbel. *Optimal vehicle scheduling in public transit.* PhD thesis, TU Berlin, 1997.

[] V. Gintner, N. Kliewer, and L. Suhl. Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, 27(4):507–523, 2005.

[] P. C. Guedes and D. Borenstein. Column generation based heuristic framework for the multiple-depot vehicle type scheduling problem. *Computers & Industrial Engineering*, 90:361–370, 2015.

[] S. Raff. Routing and scheduling of vehicles and crews: The state of the art. *Computers & Operations Research*, 10(2):63–211, 1983.

[] I. Steinzen, V. Gintner, L. Suhl, and N. Kliewer. A time-space network approach for the integrated vehicle-and crew-scheduling problem with multiple depots. *Transportation Science*, 44(3):367–382, 2010.

[] A. Ceder. Public-transport vehicle scheduling with multi vehicle type. *Transportation Research Part C: Emerging Technologies*, 19(3):485–497, 2011.

[] N. Kliewer, B. Amberg, and B. Amberg. Multiple depot vehicle and crew scheduling with time windows for scheduled trips. *Public Transport*, 3(3):213–244, 2012.

[] S. Hassold and A. Ceder. Public transport vehicle scheduling featuring multiple vehicle types. *Transportation Research Part B: Methodological*, 67:129–143, 2014.

[] E.G. Talbi. A taxonomy of hybrid metaheuristics. *Journal of Heuristics*, 8(5): 541–564, 2002.

[] B. Laurent and J. K. Hao. A study of neighborhood structures for the multiple depot vehicle scheduling problem. In *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, pages 197–201. Springer, 2007.

[] B. Laurent and J. K. Hao. Iterated local search for the multiple depot vehicle scheduling problem. *Computers & Industrial Engineering*, 57(1):277–286, 2009.

[] T. Stützle and H. H. Hoos. Max–min ant system. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[] X. Hao, W. Jin, and M. Wei. Max-min ant system for bus transit multi-depot vehicle scheduling problem with route time constraints. In *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, pages 555–560. IEEE, 2012.

[] J. A. Ramos, L. P. Reis, and D. Pedrosa. Solving heterogeneous fleet multiple depot vehicle scheduling problem as an asymmetric traveling salesman problem. In *Portuguese Conference on Artificial Intelligence*, pages 98–109. Springer, 2011.

[] T. Otsuki and K. Aihara. New variable depth local search for multiple depot vehicle scheduling problems. *Journal of Heuristics*, pages 1–19, 2014.

[] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[] V. Gintner, N. Kliewer, and L. Suhl. Solving large multiple-depot multiple-vehicle-type bus scheduling problems in practice. *OR Spectrum*, 27(4):507–523, 2005.

[] P. C. Guedes, W. P. Lopes, L. R. Rohde, and D. Borenstein. Simple and efficient heuristic approach for the multiple-depot vehicle scheduling problem. *Optimization Letters*, pages 1–13, 2015.