DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

VEHICLE SCHEDULING WITH SEQUENCE DEPENDENT TRIPS

by
Sadık Serhat KARAKÜTÜK

June, 2006
İZMİR

VEHICLE SCHEDULING WITH SEQUENCE DEPENDENT TRIPS

A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for the Degree of Master of Science in Industrial Engineering, Industrial Engineering Program

by
Sadık Serhat KARAKÜTÜK

June, 2006
İZMİR

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **"VEHICLE SCHEDULING WITH SEQUENCE DEPENDENT TRIPS"** completed by **SADIK SERHAT KARAKÜTÜK** under supervision of **ASST. PROF. DR. M. ARSLAN ÖRNEK** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. M. Arslan ÖRNEK

Supervisor

Prof. Dr. Semra TUNALI

(Jury Member)

Dr. Deniz TÜRSEL ELİİYİ

(Jury Member)

Prof. Dr. Cahit HELVACI

Director

Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

In this part of the thesis, I would like to thank people who made completion of this study possible.

I would like to express thanks to my supervisor Asst. Prof. Dr. M. Arslan ÖRNEK and to Dr. Deniz TURSEL ELİİYİ for their guidance, encouragement and support throughout this research and for careful reading of the drafts and valuable feedback.

I would like to give the deepest thanks to my father Niyazi, my mother Fatma, my sister and her husband Nesrin & Süleyman SARIBAŞ, my brother and his wife Murat & Aytül for their love, encouragement, support, patience and their trust in me.

This thesis is dedicated to my family and my nephew Murat Arda.

Sadık Serhat KARAKÜTÜK

THE VEHICLE SCHEDULING WITH SEQUENCE DEPENDENT TRIPS

ABSTRACT

In this study, we consider a vehicle scheduling problem with sequence dependent trip times. The problem is assigning vehicles to a set of trips with fixed ready times and deadlines, while minimizing cost. The trip time for a vehicle between any two places is also known deterministically. A number of different types of vehicles are available for transportation, each with different capacities, fixed and variable costs. The costs for regular and overtime utilization also vary for different types of vehicles. The problem resembles the Tactical Fixed Job Scheduling Problem where ready times and deadlines of jobs are known in advance, and the objective is to minimize the cost of machines to perform all the jobs. A job cannot be processed unless a machine is available at its ready time.

The problem is formulated as an Integer Programming Model. A spread time constraint determines the regular time usage of the vehicles. The formulation is coded in LINGO 8.0 and GAMS 20.2 with CPLEX solver. Due to the complex nature of the problem, it is observed that the optimal solution of even middle-size instances is very time consuming. Hence, we develop three different heuristic approaches for the problem, each one having two different types based on overtime usage allowances. The algorithms are coded in C programmer Language using DEV C++ Compiler. The average behaviour of the algorithms is investigated through computational experiments. Lower bound values for the problem are found using GAMS developing some approaches, and the performances of the algorithms are compared based on these bounds. The problems whose number of trip is more than a hundred have been solved in a very short time. The least solution of heuristic approaches is actually 10% greater than optimum solution.

Key Words: Tactical Fixed Job Scheduling, Spread Time Constraints, Integer Programming, Heuristics.

SIRALI BAĞIMLI SEFERLERDE ARAÇ ÇİZELGELEMESİ

ÖZ

Bu çalışmada, biz sıralı bağımlı sefer zamanlarına sahip araç çizelgeleme problemi üzerine çalıştık. Problem, araçların sabit başlama ve bitiş zamanına sahip seferlere minimum maliyet ile atanması problemidir. Seferler arasındaki sefer süresi deterministlik olarak bilinmektedir. Taşıma yapabilecek, farklı kapasiteye, sabit ve değişken maliyetine sahip araç çeşitleri mevcuttur. Normal ve fazla kullanımlarındaki kullanım maliyeti araç tiplerine göre değişmektedir. Problem yapı bakımından başlama ve bitiş zamanları bilinen ve amaç fonksiyonu bütün işleri yapacak araç kullanım maliyetinin minimum olacağı taktiksel sabit iş çizelgeleme problemine benzemektedir. Eğer işin başlama zamanında uygun bir makine yoksa iş yapılamaz.

Problem Tam Sayılı Programlama Modeli kullanılarak formüle edilmiştir. Araçların normal sürede kullanımları yaygınlık zamanı kısıtı kullanılarak belirtilmiştir. Problemin matematiksel modeli LINGO 8.0 ve GAMS 20.2 CPLEX çözücü kullanılarak modellenmiştir. Problemin kompleks yapısı yüzünden orta hacimdeki problemlerde bile optimum çözümün bulunması çok zaman almaktadır. Bu yüzden fazla kullanıma dayalı ikişer değişik çözüm tipinde üç farklı sezgisel yaklaşım geliştirilmiştir. Algoritmalar DEV C++ derleyicisi kullanılarak C programlama dilinde kodlanmıştır. Algoritmaların ortalama performanslarının bulunması için örnek denemeler yaratılmıştır. Çeşitli yöntemlerle ve GAMS programı kullanılarak problemin performansını değerlendirmede kullanılacak alt limitler bulunmaya çalışılmıştır. Problemin sefer sayısı yüzden fazla olan denemelerde bile sezgisel yöntem kullanılarak çok kısa sürede çözüldüğü gösterilmiştir. Problem için, sezgisel yöntemle bulunan çözümlerin en küçüğünün optimum çözümden ortalama % 10 büyük olduğu gösterilmiştir.

Anahtar Kelimeler: Taktiksel Sabit İş Çizelgelemesi, Yaygınlık Zamanı Kısıtları, Tamsayılı Programlama, Sezgiseller

CONTENTS

PAGE

CHAPTER ONE
INTRODUCTION

1.1. Motivation of Research

It is possible to encounter a wide range of scheduling problems in real life. To model and solve such problems is quite a difficult task though. Because the variables in real life are not deterministic but of a stochastic nature, and therefore they are difficult to model. Although such problems are modelled within a varied number of assumptions, the solution of scheduling problems in real life is rather difficult due to the size of the problem. Therefore by developing different solution methods, such problems are attempted to be solved. Vehicle scheduling problems set an example to scheduling problems in real life. These set of problems similarly have complex modelling features and they are difficult to solve.

Vehicle scheduling problems are modelled under several assumptions using Integer Programming. Such problems may be linear or nonlinear in nature. As a general structure, there is a series of tasks that are necessary to be processed in vehicle scheduling problems, and the vehicles to perform them. The objective function is to process all tasks with minimum vehicle usage. Yet, as the solution of substantial problems with computers is difficult, the problems are attempted to be solved with various optimization methods and heuristic algorithms. Branch – and – bound, cutting plane method and enumeration method can be given as examples to optimization methods. Heuristic methods may vary according to the structure of the problem.

Although the subject of the study "Vehicle Scheduling Problem with Sequence Dependent Trips" is actually a kind of vehicle scheduling problem, it features different qualities. In this problem the tasks should be performed within certain intervals and all of the tasks require to be completed. Due to the fact that the tasks start and end within certain intervals, the problem resembles the Interval Scheduling

Problem. Besides, all of the tasks must be completed and due to the structure of the sources, the problem is a Tactical Fixed Job Scheduling Problem. The original problem is the undergraduate thesis prepared by Karakütük & Karaçizmeli in 2004. In this study, a scheduling of a model which would provide for the carrying of the participants in a large-scale organization was attempted. Starting out from that approach, it has been attempted to find out a general solution to similar problems in this study. As for the example, the problem structure of the UNIVERSIADE 2005 games which were used by Karakütük & Karaçizmeli has been chosen. However, the solution methods have been prepared in line with the problems conforming to all these features.

1.2. Purpose

The purpose of the study is to find out a solution which will enable vehicle scheduling with sequence dependent trip problems with minimum cost. While attempting to find out the solution, the method is aimed to solve substantial problems in small computation times since such scheduling is done daily.

1.3. Significance

As mentioned before, it is quite difficult to solve scheduling problems in real life, especially vehicle scheduling problems. Therefore it is essential to develop solution methods to solve such scheduling problems in small computation times. With the help of heuristic approaches that the thesis proposes as well, it is ensured to solve large-scale vehicle scheduling problems with sequence dependent trips in an acceptable time period which could be encountered in real life. Furthermore, the results have been tested to see their closeness to optimum. As an outcome of the thesis, appropriate heuristics have been proposed.

1.4. Thesis Outline

The main objective of the thesis is to generate a solution method to vehicle scheduling problem with sequence dependent trips.

A literature survey is provided about the problem structure, which is the subject of the study in chapter 2. Also, the structure of scheduling problems is presented. As the problem is in line with interval scheduling and fixed job scheduling problems, such problem structures are also mentioned. In addition, vehicle scheduling problems are reviewed in general. Finally, solution methods to scheduling problems are also given.

In Chapter 3, the problem is defined. The problem variables and the collected data are presented, the assumptions are stated and the mathematical model so formed is explained. In addition, the heuristic approaches developed through the problem solutions are given in this chapter; as well. The application of these methods are explained too.

In Chapter 4 a comparison between heuristic approaches developed for the problem and the optimization method has been made. An experimental design is conducted and problem instances are generated. Furthermore, in order to evaluate heuristic approaches, various lower bound algorithms are developed and outcomes of these are compared with the results of heuristics. The performance of the heuristics are discussed and evaluated.

Finally, Chapter 5 summarizes the research work, and outlines directions for future research.

CHAPTER TWO
LITERATURE REVIEW

2.1 Introduction to Scheduling

Within an organization, scheduling is related to determining the timing of the use of specific resources of that organization. It relates to the use of specified resources of the equipment, facilities, and human activities. Scheduling occurs in every organization, regardless of the nature of its activities. The objective of scheduling is to achieve trade – offs among conflicting goals, which include efficient utilization of staff, equipment and facilities, and minimization of costumer waiting time, inventories, and process time. (Stevenson, 1999, p.722)

The scheduling function uses mathematical techniques or heuristic methods to allocate those limited resources to the processing of tasks. A proper allocation of resources enables the company to optimize its objectives and achieve its goals. Resources may be machines in a workshop, runways at an airport, crews at a construction site, or processing units in a computing environment. Tasks may be operations in a workshop, takeoffs and landings at an airport or stages in a construction project. Each task may have a priority level, an earliest possible starting time, and a due date. The objectives may also take many forms, such as minimizing the time to complete all tasks or minimizing the number of tasks completed after their due dates. (Pinedo& Chao, 1999, p.2)

Unfortunately, scheduling can be difficult to perform and implement because scheduling problems are often complicated by large numbers of constraints relating to each other, resources to activities and to each other, and either resources or activities to events external to the system. For example, two particular activities may interfere with each other and be unable to use the same required resources simultaneously. Or there are overlapping activities and they can not be made by the

same resource. Because of these complex interrelationships and difficulties of modeling real world, solving of large scheduling problems is very difficult. For this reason, the complex problems are tried to solve with heuristics approaches. Some algorithms are developed. Then the algorithms are tested how sensitive the solutions are to these complex problems and an approximate solutions are found to difficult problems where the complexity proves central.

First, we provide a comprehensive but not exhaustive list of scheduling problems. Then, we present a detailed overview of interval scheduling, fixed job scheduling and vehicle scheduling as they are related to our study. Finally, we mention about solution methods for scheduling problems.

2.2 Types of Scheduling Problems

Scheduling is a complex but an important operation function. There are many different types of scheduling problems faced in the real life. A partial list is as follows.

- ***Job shop scheduling***. Job shop scheduling, known more commonly in practice as shop floor control, is the set of activities in the shop that transforms inputs to output.

- ***Personnel Scheduling.*** Scheduling personnel is an important problem for both manufacturing and service industries. Although shift scheduling on the factory floor may be considered one of the functions of shop control flow, personnel scheduling is a much larger problem. Scheduling health professionals in hospitals and bus driver scheduling problems are typical examples. Determining whether to meet peak demand with overtime shifts, night shifts, or subcontracting is another example of a personnel scheduling problem.

- ***Facilities scheduling.*** This problem is particularly important when facilities become a bottleneck resource. Scheduling operating rooms at hospital set

example. As the need for health care increases, some hospitals and health maintenance organizations find that facilities are strained. A similar problem occurs in colleges and universities in which enrollments have increased without commensurate increases in the size of the premises.

- ***Vehicle scheduling.*** Manufacturing firms must distribute their products in a cost – efficient and well-timed manner. Some service operations, such as dial – a – ride systems, involve pick-ups and deliveries of goods and/or people. Vehicle routing is a problem that arises in many contexts. Problems as scheduling snow removal equipment, postal and bank deliveries, and shipments to costumers with varying requirements at different locations are some examples.

- ***Vendor scheduling.*** For firms with just – in – time (JIT) systems, scheduling deliveries from vendors is an important logistics issue. Purchasing must be coordinated with the entire product delivery system to ensure that JIT production systems function efficiently.

- ***Project scheduling.*** A project may be broken down into a set of interrelated tasks. Although some tasks can be done concurrently, many tasks cannot be started until others are completed. Complex projects may involve thousands of individual tasks which must be coordinated for the project to be completed on time and within budget. Project scheduling is an important component of planning function.

- ***Dynamic versus static scheduling.*** Most scheduling theories view the scheduling problem as a static one. Numerous jobs arrive simultaneously to be processed on a set of machines. In practice, many scheduling problems are dynamic in the sense that jobs arrive continuously over time. One example is the problem faced by an air traffic controller who must schedule runways for arriving planes. The problem is a dynamic one in that the planes arrive

randomly and runways are freed up and committed randomly over time. Dynamic scheduling problems are analyzed using the tools of queuing theory.

- ***Interval scheduling.*** A formal schedule is given in advance; most actual processing is expected to conform to that schedule, even if unanticipated emergencies and other happenings force some change. A good example is an executive's appointment log. Interval scheduling is useful when the use of several critical resources must be coordinated. Interval scheduling can cause large gaps and inefficiencies in the schedule. In practice, schedules often become quite patchwork. (Morton & Pentico, 1993, p. 20)

2.3 Interval Scheduling

There are given a set M of machines, a set I of intervals, and a mapping $g : I \rightarrow 2^M$ which determines on which machines each interval can be scheduled. Each interval i has a fixed ready time r_i and a fixed processing time p_i. Interval i must be assigned to a machine immediately at time r_i, and it must be continuously processed until it is complete, or else it is lost. A machine can only process one interval at a time. In other words, overlapping intervals cannot be scheduled on the same machine. The goal of this problem is to find a schedule that processes the maximum number of intervals. These types scheduling problems are named Interval Scheduling Problem. (Wagner, 2001)

Interval Scheduling is an emerging area of scheduling where tasks, each equipped with specified ready times and deadlines, are to be processed on a number of resources. The problem is typical for reservation systems and has many real life applications such as classroom assignment, transportation systems, and shift scheduling. Reservation systems may arise in service environments like room reservation and car rental or repair services, where customers represent tasks and hotel rooms, cars or technicians correspond to resources. In production environments, the tasks give reservation request to resources during specified time windows of processing. The problem may involve the decision as to which orders are

to be accepted or how many resources are to be allocated to serve all orders. (Eliiyi, 2004)

Bouzina and Emmons (1996) studied interval scheduling where jobs with fixed start and end times were processed on identical parallel machines. The objective was to find a feasible schedule with the maximum number of completed jobs. When job weights were defined, the problem was to find a solution that maximized the sum of the weights of completed jobs. The non-weighted and weighted versions were referred to as the maximal interval scheduling and maximal-weight interval scheduling, respectively. The authors showed that these type problems are NP – hard.

There are two main types of interval scheduling problems. In the first type of problem, there are a fixed number of machines, and the goal is to determine if all of the jobs can be scheduled on the available machines, or to maximize the number of jobs that can be scheduled. It may be the case that some jobs have higher priority than others, in which case the job weighted, and the goal is to maximize the weight of the scheduled jobs. (Wagner, 2001)

In the second type of problem, there is an unlimited pool of machines and the goal is to find a set of machines with minimal cost that can process all of the jobs. The machines can be different types and they can have different costs. If the machines are identical, then any job can be scheduled on any machine, and the weight of the job is independent from the machine on which it is run. It is worth noting that in Interval Scheduling problems, the concept of machine speed is generally not relevant. Jobs have a fixed starting and ending time which does not depend on the machine on which the job is run. All machines operate at the same speed. In this case the goal of the second type interval scheduling problem is to find the smallest number of machines necessary to process all jobs.

The special types of interval scheduling problem on identical machines are described in the following subsections.

2.3.1 Restricted Interval Scheduling

If the machines are restricted, then a given job can only be scheduled on subset of the machines. All of the machines have the same speed, thus identical. These type problems will be referred to restricted interval scheduling problem.

Restricted Interval Scheduling is first considered by Arkin and Silverberg (1987). Their problem is the basic interval scheduling problem where each interval can be scheduled on arbitrary subset of the machines. The authors show that Restricted Interval Scheduling is NP – complete. They have developed a dynamic programming solution method.

Another version of Restricted Interval Scheduling Problem is Class Scheduling Problem, where the jobs and machines are divided into classes. In the most general case, each machine and job forms its own class. Kolen and Kroon (1991) show that if there are three or more dependent classes of machines, then deciding if all the intervals can be scheduled is NP – complete. A set of machine classes are dependent if the classes of jobs they can process overlap. Kolen and Kroon (1991) also show that if there are two or more dependent classes of machines then the optimization problem is NP – hard.

Another Class Scheduling Problem where each job has a fixed ready time, a fixed end time, and a value representing the job priority is discussed by Kolen and Kroon (1993). Machines are available in specific time intervals (shifts), and a job can be processed only if the interval between the start and end time is a subinterval of a machine's shift. A machine can process at most one job at a time and preemptions are not allowed. The objective is to find a feasible schedule for all jobs when such a schedule exists. Otherwise, the objective is to find a feasible schedule in which the subset of processed jobs yields the maximum total value. The feasibility problem is referred to as shift class scheduling while the optimization problem is referred to as maximum shift-class scheduling. Shift class scheduling is NP-complete but can be solved in polynomial time if preemptions are allowed. Maximum shift-class

scheduling is NP-hard. If the number of shifts and the start and end times of each shift are known in advance, shift class scheduling is NP-complete while maximum shift-class scheduling is NP-hard. Kolen and Kroon (1993) present some special cases where each can be solved in polynomial time.

Classroom Assignment Problem is an interval scheduling problem where the classrooms correspond to machines and the classes are jobs. This problem is discussed by Carter and Tovey (1992). They consider variations where several classes meet several times a week but must be scheduled in the same room, in which case of intervals must all be scheduled on the same machine. They also consider the problem where classes can be scheduled in any room that is large enough. This is a special case of restricted interval scheduling where machines can be arranged in a hierarchy. In general these problems remain hard.

2.3.2 Online Interval Scheduling

In online interval scheduling problem, jobs must be scheduled on a single machine which runs one job at a given time. The problem is online in that jobs are unknown until their ready time. Each job must be started or rejected immediately when it becomes known. Each job has a fixed value, which is gained if the job runs uninterrupted to completion. If a job arrives and there is a job running, the running job may be aborted in order to allow the new job to start. The goal is to maximize the sum of the values of those jobs which run to completion. This and similar scheduling problems arise in important application areas such as continuous media and call control. (Seiden, 1998)

Online interval scheduling problem is studied by Woeginger (1994). He considers the single machine case. Job length and weights are known when they arrive. Jobs can be preempted, but they are lost. The quality of online algorithms depends on the relation between the length of the jobs and the weight of the jobs. If the weights of the jobs have no relation to the lengths of the jobs then no online algorithm is competitive. An instance of the problem is called f – related if there is a function $f(x)$

that maps lengths to weights. If *f(x)* is concave or decreasing then there exists a *4 – competitive* online algorithm. The algorithm is straightforward. A job J_k is scheduled if the machine is idle, or if its value is twice the value of the job currently being processed, or if its end time is before the end time of the job currently being processed and its value is greater than the value of the job currently processed. They show that for concave *f- related* instances, no algorithm has a competitive ratio better tan $4 - \varepsilon$, so the straightforward algorithm is optimal.

2.3.3 Minimal Resources Interval Scheduling

Given a set of jobs called operations, each associated with a release time and a deadline, and a set of processors called components, the problem of minimal resource interval scheduling is to find a schedule of operations on the components, such that each operation is started after its release time and completed before its deadline while the total resource cost of the components is minimized. Minimal resource interval scheduling is an important scheduling problem with wide applications. It is also an efficient approximation to some other hard scheduling problems such as precedence constrained scheduling, etc.

Minimal resource interval scheduling problems were modeled by Shen and Jong (1999) using integer linear programming. Minimal resource interval scheduling problem is strongly NP – hard, as shown by Shen and Jong (1999).

2.4 Fixed Job Scheduling

If the job cannot be delayed after its ready time, then interval scheduling problem becomes a fixed job scheduling. The common feature in these problems is that the job *j* has a fixed starting time r_j, completes at d_j, and a job must be processed continually for a fixed amount of the time or else it is lost.

Given *n* tasks $T_{j,}(j = 1, \ldots, n)$ each of which requires processing without interruption from a given release time r_j to a given deadline d_j $(j = 1, \ldots\ldots, n)$, and an

unlimited number of identical parallel processors P_1, P_2, …, P_n each of which can process at most one task at a time. The Fixed Job Scheduling (FJS) Problem is to determine a feasible assignment of tasks to processors, such that the number of required processors is at minimum.

Two different generalization of operating time constraint of FJS problem have been considered in literature. These operating time constraints are spread time and working time constraints. Spread time constraints impose an upper bound on the time between the start and finish times of the operations on any machine. The working time constraints impose an upper bound on the sum of the processing times of the tasks assigned to each processor.

FJS problem with spread time constraint is considered by Martello, Fischetti and Toth (1987). In this problem each processor is available only for s time units from the release time of the earliest task assigned to it, in the sense that any pair of tasks (T_j, T_k) that are assigned to sense processor must satisfy the relation $d_j - r_k \leq s$. The idle time between the start and finish times are included in the spread time. This problem arises naturally in more general Bus Driver Scheduling Problem. They modeled this problem and showed that this problem is NP – hard and proposed a branch and bound algorithm for the exact solution of problem.

A real life application of bus driver scheduling problem with working time constraint is pointed out by Martello, Fischetti and Toth (1989). In this problem each processor is allowed to operate only for w time units. The processor must satisfy the relation $d_j - r_j \leq w$. The working time constraint is limited by the working time of bus drivers. Unlike spread time case, it does not include the idle times. They modeled this problem and showed that this problem is NP – hard and proposed a branch and bound algorithm for the exact solution of problem.

Another FJS Problem is considered by Yuan and Lin (2005). They consider the single machine preemptive scheduling problem with some fixed jobs being previously given. The fixed jobs are already fixed in the schedule. The remaining

jobs are to be assigned to the remaining time-slots of machine in such a way that they do not overlap each other. The objective is to minimize a tardiness related criterion. If the jobs are processed without preemption, in literature this problem is strongly NP-hard.

The FJS problem has two types based on the objective function.

The first type of the FJS problem is the Operational Fixed Job Scheduling (OFJS) problem, where each job j has a weight w_j that represents its value or relative importance, and the concern is maximizing the total weight of the processed jobs with a given number of processors. When all jobs have equal weights, the objective reduces to maximizing the number of jobs processed (Eliiyi, 2004).

Eliiyi (2004) considers the OFJS problem on the identical machines. The author uses two different constraints for this problem. It is assumed that the jobs have fixed ready times and deadlines and working time constraint at the first model and spread time constraint at the second model are imposed on machines. Their objective is to select a set of jobs for processing so as to maximize the total weight. They show that the problem is strongly NP-hard. They use branch and bound algorithm to find an optimal solution.

The second type of the FJS problem is the Tactical Fixed Job Scheduling (TFJS) problem. In these type problems, there are unlimited pools of machines and the goal is to find a set of machines with minimal cost or minimum number of machines that can process all of the jobs (Wagner, 2001). In a study by Kroon (1990), the TFJS problem is used as the core model in tactical capacity planning of aircraft maintenance personnel for an airline company.

A special case of FJS problem is the Operational Fixed Interval Scheduling Problem (OFISP). OFISP is characterized as the problem of scheduling a number of jobs, each with a fixed starting time, a fixed finishing time, a priority index, and a job

class. The objective is to find an assignment of jobs to machines with maximal total priority. The problem is complicated by the restrictions that:

i. Each machine can handle only one job at a time,

ii. Each machine can handle only jobs from a prespecified subset of all possible job classes,

iii. Preemption is not allowed.

It follows from the above that OFISP has both the character of a job scheduling problem and that of an assignment problem. (Kroon, Solomon and Wassenhove, 1993)

Several projects in which OFISP plays an important role were proposed by different researches. Some examples of these projects are briefly discussed below:

- *The assignment of airplanes to gates*. This problem was taken up at Schiphol Airport, where airplanes of different types have to be assigned to gates during fixed intervals. However, each gate can only handle a limited set of aircraft types due to technical restrictions. The problem here is to find an assignment of airplanes to gates where the number of unassigned airplanes - whose passengers have to be transported to the terminal by bus – is minimized.

- *The scheduling of operating rooms in a hospital.* In most hospitals a limited number of operating rooms are available. Some of these operating rooms may be general purpose, but others may be suitable for only a subset of the various types of operations. In general the time slot for an operation is fixed.

- *The assignment of holiday bungalows to vacationists*. Usually holiday bungalows are booked a long time in advance for a period of one or more weeks. The holiday bungalows may differ in several aspects, i.e. size, location,

accommodation, quality, and price. Each season the booking office is faced with the problem of finding an assignment of holiday bungalows to vacationists, such that there is a matching between the requirements of the vacationists with respect to e.g. comfort, and the available accommodation.

2.5 Vehicle Scheduling Problem

The vehicle scheduling problem (VSP) consists of assigning a set of scheduled trips to a set of vehicles, in such a way that each trip is associated to one vehicle and a cost function is minimized. The VSP is a classical optimization problem which is faced in the operational planning of public transportation systems. (Baita et al., 1990)

Vehicle scheduling problems have different objective functions. First objective type is minimizing the total number of vehicles that is, minimizing the capital cost or fixed cost of vehicles. Second objective type is the total deadhead time or deadhead cost of operations. Third objective type is maximizing the utilization of vehicles, which is equivalent to minimizing the idle time or arc time.

VSP is modeled by integer programming. Arc time, which is defined as the time it takes for a vehicle to go from the arrival point of the first trip to the departure point of the second trip, overlapping trips set, which is a set of trips which can not be made by a vehicle, spread time constraint and working time constraint are used in modeling VSP.

There are efficient algorithms for some versions of the VSP, i.e., when all vehicles are equal and share the same depot. Nevertheless, real-life applications may turn out to be complex due not only to the dimension of the problem but also, and more importantly, to the particular requirements which are present in practical situations but are hard to be modeled. Practical requirements for this problem, usually not considered in the literature, include considering several performance evaluation criteria, producing different alternative solutions, and getting hints on how data could be modified to improve the quality of the solutions. (Baita et al.,1990)

2.5.1 Why Vehicle Scheduling Problems Are Complex

Most real vehicle scheduling problems are difficult for modeling and complex for solving. Schrage (1981) lists 10 features that make real problems difficult to solve. These include:

- ***Frequency requirements.*** Visiting customers may have to occur at a certain frequency and that frequency may vary from customer to customer.

- ***Time windows.*** This refers to the requirement that visits to customer locations be made at specific times. Dial – ride systems and postal and bank pickups and deliveries are typical examples.

- ***Time dependent travel time.*** When deliveries are made in urban centers, rush – hour congestion can be made an important factor. Travel time depends on the time of days.

- ***Multi dimensional capacity constraints.*** There may be constraints on weight as well as on volume. This can be a thorny issue, especially when the same vehicles are used to transport a variety of different products.

- ***Vehicle types.*** There may be several vehicle types to choose from. Vehicle types may differ according to capacity, the cost of operation, and whether the vehicle is constrained to closed trips only. When several types of vehicles are available, the number of feasible alternatives increases dramatically.

- ***Split deliveries.*** If one customer has a particular requirement, it could make sense to have more than one vehicle assigned to that customer.

- ***Uncertainty.*** Scheduling algorithms invariably assume that the information is known in advance. In practice, however, the time required to cross certain

portions of network could be highly variable depending on factors such as traffic conditions, weather, and vehicle breakdowns.

2.5.2 Types of Vehicle Scheduling Problems

The general vehicle scheduling problems are the problems in which a number of vehicles starting at one or more depot have to collectively visit a number of demand points then return to the depot from which they start.(Haghani et al., 2003) There are two different main types of vehicle scheduling problem: Single Depot and Multi Depot Vehicle Scheduling Problem.

2.5.2.1 Single Depot Vehicle Scheduling Problem

The single depot vehicle scheduling problem (SDVSP) contains the basic structure of the scheduling problem. Suppose there are *n* trips to be served by vehicles starting from a single depot. Every trip has a starting point, an ending point, a starting time, and an ending time. The trips could be served by the same vehicle if the starting time of one trip is greater than the ending time of another trip plus the travel time between these two trips. The objective of the problem is to find the minimum number of the vehicles to serve all the trips.

A network could be constructed to represent this problem: each trip represented by a node, and an arc *(i; j)* exists if the ending time of trip *i* plus the travel time between ending point of trip *i* and starting point of trip *j* is less than the starting time of trip *j*. Then the problem is to find the minimum number of paths in this network that cover all the nodes. The path here is treated as a vehicle scheduled to start from the depot and end at the depot. This problem could be solved as a minimum cost flow problem.

Several network flow type of algorithms have been proposed for the SDVSP. The SDVSP problem has been formulated as a linear assignment problem, a transportation problem, a minimum cost flow problem, a quasi – assignment problem

and a matching problem. Freling et al (1999) review the most relevant algorithms and they propose new two phase algorithm, which is valid in case of the special cost structure and a new core oriented approach.

The single depot scheduling problem is well – known to be solvable in polynomial time. However, when other constraints such as the route time constraints that make the problem more realistic are introduced, it becomes a NP-hard problem. Large problems of this type can be solved only by heuristic procedures.

2.5.2.2 Multi Depot Vehicle Scheduling Problem

The multiple depot vehicle scheduling problems (MDVSP) are extensions of the SDVSP problem. The major difference between the two is that in the multiple depot case vehicles are housed at different depots. The objective is to determine the minimum number of vehicles to serve all trips and to identify the optimal locations of the vehicles in order to minimize the total cost. The MDVSP can be formulated as a mixed integer programming problem in two different ways: ‘‘Trip Based’’ formulations, in which the trips are the components to which the variables are related, and ‘‘Block Based’’ formulations, in which the blocks serve that purpose. Bertossi et al. (1987) proved that the problem is NP-hard.

Some researchers formulated the MDVSP as a multi – commodity formulation which is a network flow formulation that accounts for multiple commodities shipped from different origins to different destinations in the network. Desaulniers et al (1998) studied MDVSP with time windows and waiting cost. They formulate this problem as an integer nonlinear multi – commodity network flow model with time variables and they solve it using a column generation approach embedded in branch – and – bound framework.

Many branch and bound or heuristics approaches are used to solve the MDVSP but the most successful approach for solving it is the work of Löbel (1997). He solved large real world problems using a specific type of column generation called

‘‘Lagrangean Pricing’’. He solved a problem consisting of 25,000 trips with more than 13 million variables to optimality and a similar problem with about 70 million variables to a good feasible solution.

2.6 Solution Methods for Scheduling Problems

Solution methods for scheduling problems can be classified as exact optimization approaches and heuristic approaches. Exact optimization approaches include integer programming solutions, dynamic programming solutions. A heuristic method includes basic dispatch schedule and search techniques.

2.6.1 Exact Optimization Approaches

2.6.1.1 Integer Programming (IP)

A surprisingly wide class of problems can be modeled using integer variables and linear constraints. Sometimes such a model consists solely of integer variables. That is a *pure integer programming (IP) model*. More commonly there are both conventional continuous variables together with integer variables present. Such a model is said to be a *mixed integer programming (MIP) model.* (Williams, 2003, p: 144)

We can think of situations where it is only meaningful to make integral quantities of certain goods or use integral quantities of some resources. In these cases IP model might be used of many different types of problem. *Knapsack problem, sequencing problem, location problem, scheduling problem, transportation problem, assignment problem, set partitioning problem, depot location problem* are some example problems for which IP models may be built.

There is a considerable danger in building an IP model only to see that it is not possible to solve it in a reasonable time using a modern computer and package programs. For this reason some methods of solving IP are developed. Branch – and

bound method, cutting planes method and enumerative methods are some different methods by which IP models may be solved.

Enumerative methods are generally applied to the special class of 0 – 1 PIP problems. In theory there are only a finite number of possible solutions to a problem in this class. Although it would be quite difficult to examine all these possibilities, by use of a tree search it is possible to examine only some solutions and systematically rule out many others being infeasible or non – optimal.

Branch and bound method is one of the most popular methods of solving IP. The branching refers to a partitioning of the solution space; each part of the solution space is then considered separately. The bounding refers to the development of lower bounds for parts of solution space. If a lower bound on the objective in one part of the solution space is larger than an integer solution already found in a different part of solution space, the corresponding part of the former solution space can be disregarded. (Pinedo& Chao, 1999, p.396)

Cutting plane methods usually start by solving an IP problem as if it were an LP problem by dropping the Integrality requirements. If the resultant LP solution is integer, this solution will also be the integer optimum. Otherwise extra constraints are systematically added to the problem, further constraining it. The new solution to the further constrained problem may or may not be integer. By continuing the process until an integer solution is found or the problem shown to be infeasible the IP problem can be solved.

2.6.1.2 Dynamic Programming

Dynamic programming is a technique that can be used to solve many optimization problems. Dynamic programming obtains solution by working backward from end of a problem toward the beginning. The problem can be divided into stages with decision required at each stage. Each stage has a number of states associated with it. Decision makers choose the state which is an optimal decision at their stage and they

transform it into the state at the next stage. Dynamic programming technique is an implicit enumeration method, i.e., it evaluates only a subset of all possible solutions.

Since the number of stages becomes large rapidly in scheduling problems, dynamic programming approach is difficult to implement from the computational point of view, due to its memory and time requirements. (Edis, 2004)

2.6.2 Heuristics Approaches

For both integer programming and dynamic programming, however, while very small problems can be solved in a reasonable time; solving the large problems are very hard. These problems are usually known *NP – complete.* For this reason, various heuristics approaches are developed for approximate solution of the large problems.

2.6.2.1 Dispatch Schedule

A formal scheduling may or may not be given in advance, but simple practical changes may be handled just by adjusting/slipping the whole schedule in a flexible way. The emphasis is on scheduling resource by resource, keeping each resource busy with the most important activity available. When resource becomes free, the highest priority activity available is performed next. This tends to produce a compact, efficient schedule, but necessary complicated resource matching may be more difficult. While some users may object to a schedule with constant minor change, the entire schedule remains logical and tightly knit in terms of priorities. (Morton & Pentico, 1993, p. 20)

Dispatching heuristic has some rules that prioritize all the activity that are waiting for using resource. These rules can be classified in various ways. First way of classifying dispatching rules is according to the information they are based upon. A *local* rule uses information pertaining to either the queue where activity is waiting or the resources where the activity is queued. A *global* rule may use information

pertaining to other resources, such as the processing time of the activity on the next resources on its route or current queue length at that resource. Another classification of dispatching rules is *static* or *dynamic* rules. Static rules are not time dependent but dynamic rules are.

2.6.2.2 Search Techniques

There are a number of heuristic search techniques to solve large scale problems (Pinedo& Chao, 1999, p.421).

- ***Neighborhood Search***: Neighborhood search is a rather general technique used for scheduling. First pick a feasible starting solution, using any method. Next, try all possible ways of schedule slightly and evaluate each resulting schedule. If there is no improvement, the method is finished. Otherwise, take the largest improvement and begin looking small changes from that, and so on. It arrives at only local optimum, but is very useful especially if used in conjunction with good started heuristic method.

- ***Tabu Search:*** Tabu search which is neighborhood search with list of recent search position is another search technique. The best solution to date is also always saved in case no better solution is ever found.

- ***Simulated Annealing:*** Simulated annealing also adds diversification to a neighborhood search procedure, but in a somewhat different way. A random amount is added to each possible move's evaluation.

- ***Genetic Algorithms:*** There is a current population of possible solutions to the problem. In each generation, the best solutions are allowed to produce new solutions (children) by mixing features of the parents; the worst children die off to keep the population stable, and on to the next generation.

- ***Beam Search****:* Beam search is a derivative of branch – and – bound, which tries to eliminate branches in an intelligent way so that not all branches have to be examined. It thus requires less computer time, but can no longer guarantee an optimal solution.

Savings heuristics, a time oriented – nearest neighbor heuristics, insertion heuristics and *a time – oriented sweep heuristics* are other heuristics methods to solve vehicle scheduling and routing problems. Solomon (1987) considers the design and analysis of such heuristic algorithms for vehicle scheduling and routing problems with time window constraints. They show that insertion heuristics prove to be very successful.

CHAPTER THREE

PROBLEM STATEMENT AND SOLUTION APPROACH

3.1 Introduction

In this chapter, the problem will be defined, factors and objectives will be identified and the mathematical model appropriate to the goal and to the structure of the problem will be constructed. Besides, the modelling of this problem with the optimization package programs and the heuristic solution approaches will be fully explained through this chapter. The problem can be defined as the assignment of the jobs which have certain time intervals to the machines. What is meant by the time interval here is that the jobs have a fixed ready time and operation period. All the jobs should be done right on time. The number of the available machines is infinite. The objective is to schedule – the minimum number of machines to process all jobs.

Such scheduling problems are encountered in real life. In this study, a problem is examined where the machines correspond to vehicles and the jobs to trips. Generally, this problem is a vehicle scheduling problem. The problem examined in the study is inspired by the study of Karakütük and Karaçizmeli (2004).

Though possible, it is rather difficult to model the variables and situations in real life mathematically. Therefore, to model such problems, several assumptions are made. It is rather difficult to find out the optimum solutions of the scheduling problems in real life. Thus, heuristic approaches have been prepared for the problem in compliance with the assumptions to be used in modeling.

The method to be used in the modeling of the problem is the mixed integer programming (MIP), and the constraints are structurally the ones to be encountered

in interval scheduling and fixed job scheduling problems. The solution algorithm has been prepared in compliance with similar scheduling problems. In addition, by making slight adjustments in algorithms, heuristic approaches have been created apart from such vehicle scheduling problems, applicable to similarly structured problems in a different domain.

3.2 Problem Definition

The problem is to form the cost-efficient vehicle schedules, which will be used in a large–scale organization for the transportation between the participants' accommodation facilities and the facilities where the activities will take place, in compliance with the terms of the organization. Problem will be solved daily; that is, a new schedule is made for each day. The main reason is that the organization program changes from day to day. The daily programs are known beforehand. There are two basic parameters that form the problem. The first one is a trip which corresponds to the participants' transportation activity from one center to another. The other is the vehicle, which will conduct the carrying of the demands of the trips. The collected data, the data sets formed out of these data change with respect to these two parameters. In order to render the problem comprehensible, these two parameters and the data sets connected to them are required to be explained in detail.

3.2.1 Data Related with Trips (j)

A trip is the act of transportation of passengers from a center to another, and it is required to be performed in a certain time interval. Daily organization program is known, for this reason the daily required number of trips is known in advance. Furthermore, the trip set stands for an act of transportation between only two centers, that is to say, the trips do not consist of consecutive transportations between three or more centers. In fact, due to the structure of the problem, a trip stands for the transportation between just two centers.

The trips are the acts that have a ready time and deadline. As seen in Figure 3.1, each trip in the time axis corresponds to a certain time interval. These time intervals sometimes overlap with one another. Each time interval has a departure and arrival point on time axis, known in advance. A trip demand emerges at each time interval. The trip demand corresponds to the number of passengers on that trip. The transportation of the demand should necessarily be performed within this time interval.

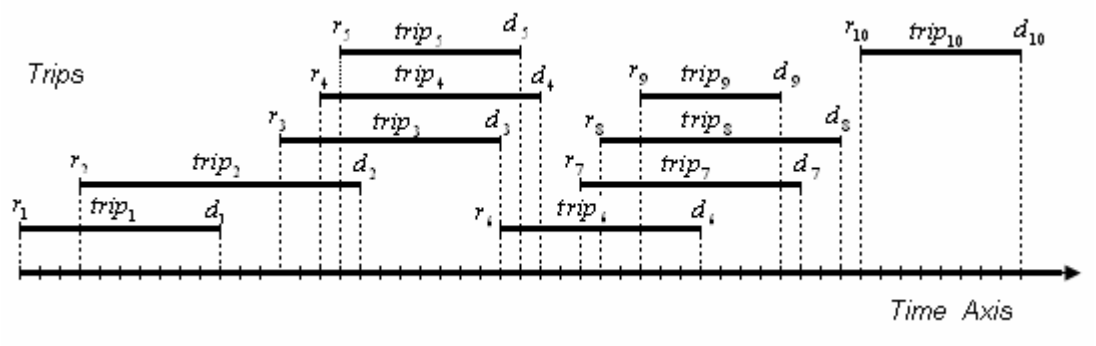


Figure 3.1 The Time Interval of Trips

All trips have a definite ready time (r_j) and deadline (d_j). The ready time and deadline are known in advance and deterministic ones. Thus, every single trip has a definite processing time ($p_j = d_j - r_j$). Namely, the time to be spent on each trip is deterministic and known beforehand. The vehicle to be assigned to each trip necessarily has to be at the departure point in ready time. It is not possible to delay the trips after the ready time.

The passenger demands for the trips (D_j) are deterministic and they are known in advance. It is necessary to assign adequate number of vehicles to supply the demand for each trip. That is, the demand for each trip has to be satisfied at any rate. There are two kinds of trips based on demands.

The demand for the first type of trips is split. That is to say, in order to meet the demand, more than one vehicle may be used. The important thing here is to supply the adequate number of seats to meet the demand. Therefore, there is a necessity to

assign more than one vehicle for the trips whose demands exceed the capacity of the vehicle with the largest capacity; since all the demands of the trips should be satisfied. Furthermore, the trips pertaining to the first type, and whose demands are smaller than the capacity of the vehicle with the largest capacity also can be conducted with more than one vehicle according to the problem.

The second type of trips is the ones which are required to be conducted with only one vehicle. Due to the organizational conditions, the demands of some trips should necessarily be satisfied with a single vehicle; therefore some trips should necessarily be taken with a single vehicle. The demands of such trips should be equal to or smaller than the capacity of the vehicle with the largest capacity. Otherwise, an assignment without a division is not possible and the problem would be infeasible. The set (*T*) constituted by the second type demands of the trips formed during the day time should be known in advance.

Each trip has a departure and arrival point. If a vehicle is to take another trip after finishing one, it has to get from the arrival point of the first trip to the departure point of the other trip. When the vehicle takes a trip *b* after a trip *j*, the elapsed time that the vehicle gets from the arrival point of *j* to the departure point of *b* is called the arc time (a_{jb}) between the places *j* and *b*. Arc time could be regarded as a setup time. That is to say, it is the preparation time of the machine from the latest task to its new task.

The arc time between the trips can be explained more clearly with Figure 3.2. Here the departure point of the first trip is DP_1, and the arrival point of the same trip is AP_1. Let's assume the departure point of the next trip after the first one as DP_2, and the arrival point as AP_2. In such a case, the arc time (a_{12}) is the elapsed time that the vehicle travels from the first trip's arrival point AP_1 to the second trip's departure point DP_2. In order for the vehicle to take the second trip, the sum of the first trip's deadline (d_1) and the arc time between the trips (a_{12}) has to be smaller than the

second trip's ready time (r_2), in other words the condition ($a_{12} + d_1 \leq r_2$) has to be met. Otherwise, the vehicle cannot take that trip, but might another trip.

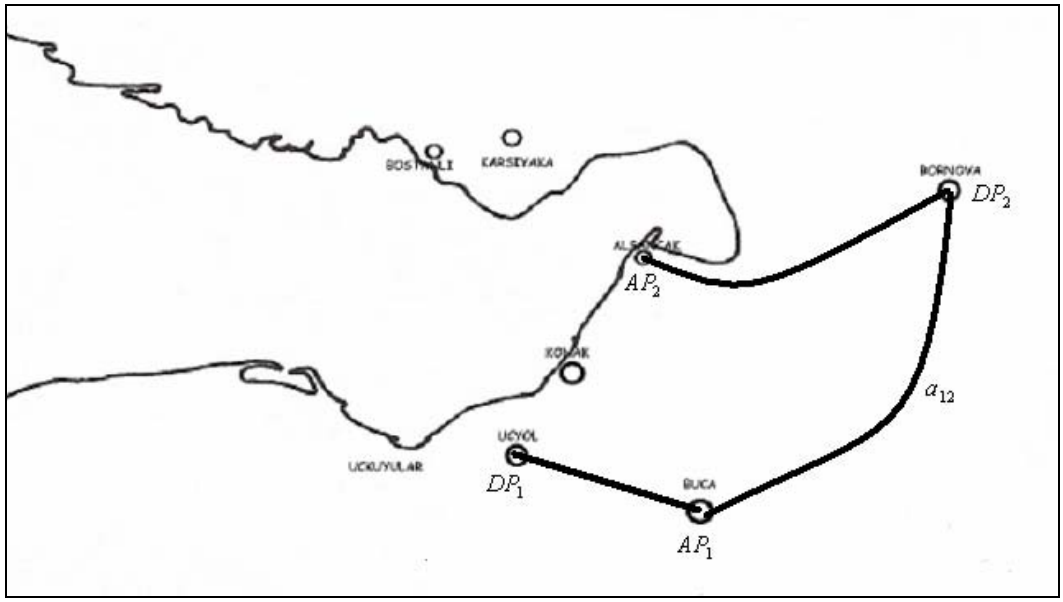


Figure 3.2 Arc Time between Two Trips

3.2.2 Data Related with Vehicles (i)

The second significant parameter is the one dependent on the vehicles to transport the demands of the trips from one center to another. As a matter of fact, the problem is assigning the vehicles to the trips. The vehicles have cost of use, usage period and types with definite capacities.

Single or various types of vehicles may be used in the problem. It has been assumed that there are an enough number of vehicles in each size. That is, if there are k types of the vehicles, then that is assumed an enough number of all types of vehicles are available. By this way, the number of the vehicles to be used will be determined dependent on the structure of the trips.

Each vehicle has a definite capacity (c_i). The vehicles cannot exceed these limits, yet in order for the vehicle to complete a trip; not all the seats are required to be

occupied. Even though only one person is to be transported, the vehicle should be assigned to that trip. The maximum passenger capacity for each vehicle differs.

As we have mentioned before when explaining the trips, each vehicle can be assigned to only one trip and can transport passengers between two centers at a single slot of time. In other words, even though the vehicle's capacity is fit for more, it is allowed to transport the passengers of a single trip.

The average speed of the vehicles is assumed as constant. Actually, based on this assumption, it has been mentioned before that each trip and arc time are deterministic.

Each vehicle has a daily fixed cost dependent on its size. When a vehicle is assigned to a trip, its fixed cost should be tolerated. In exchange of the fixed cost (f_i) the right to use the vehicle for a certain period is obtained. This period is called daily available regular time (S). Throughout this period, the vehicle can be used without extra charges. In addition, each vehicle has a variable cost (v_i) dependent on its type. Variable cost is the unit overtime (o_i) cost of use. If the vehicle is used more than the standard or variable time, an extra cost is charged for every unit of time as much as the variable cost. Each vehicle has a limit for overtime usage, which is called the daily available overtime (O) limit. A vehicle can be used as much as the sum of the daily available regular time and the daily available overtime ($S+O$) at the most. The usage limits for all the vehicles are the same. The vehicle's cost of use is made up of the fixed and variable cost. As a matter of fact, the objective function of the problem is to minimize the total fixed cost plus the total variable costs of the vehicles.

3.2.3 Incompatibility Set

Some trips cannot be handled by the same vehicle. The set of the trips that cannot be transported by the same vehicle form incompatibility sets (Q'_j). This set varies

dependent on both the trips and the vehicles. There can be three reasons for the two trips not to be taken by the same vehicle. These three cases can be seen in Figure 3.3. separately.

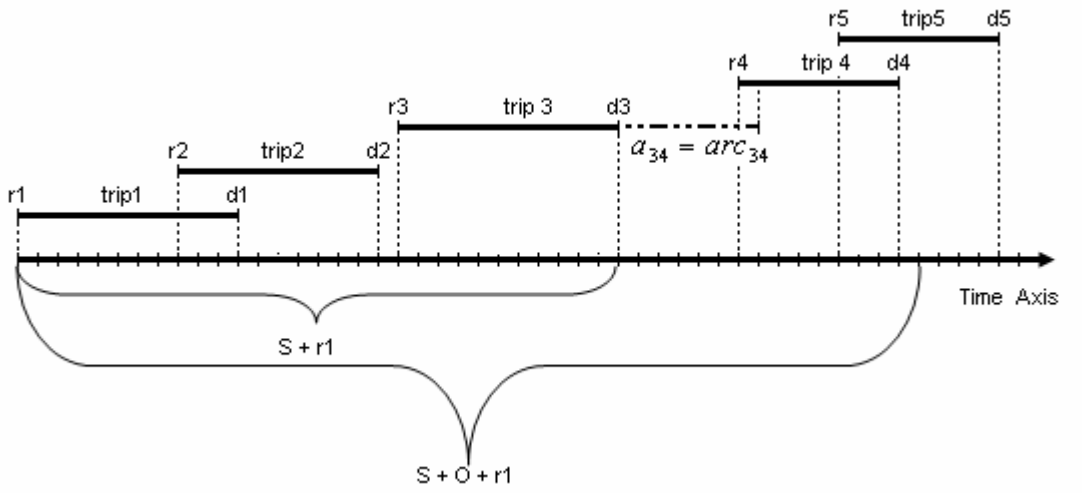


Figure 3.3 An Example Time Intervals of Trips for Incompatibility Set

The first reason is the overlapping of the intervals of two trips. For any trip *j*, if there is a trip *b* which provides $r_j \le r_b \le d_j$, trips *j* and *b* overlap and therefore they cannot be performed by the same vehicle. To explain this, let us examine the overlapping 1st and 2nd trips in Figure 3.2. If the vehicle has been assigned to the 1st trip, then it is busy from r_1 to d_1 time. As the vehicle is busy and it can only be assigned to a single trip, it will not be present at the ready time of the second trip r_2. Therefore, 1st and 2nd trips cannot be taken with the same vehicle.

Another reason is that, a vehicle cannot perform two trips due to the arc time between the two destinations. If $r_j \le r_b < d_j + a_{jb}$ for the trip *b* following the trip *j*, these trips cannot be taken by the same vehicle. The 3rd and 4th trips in Figure 3.3 can be shown as an example for the case. The time intervals of these do not overlap, but the arc time between the 3rd and 4th trips is more than the difference between the ready times and deadlines of these trips. Thus these trips cannot be taken by the same vehicle.

As the last reason, if the difference of the trip *j*'s ready time (r_j) and the trip *b*'s deadline (d_b) exceeds the total use time ($S + O < d_b - r_j$), then the vehicle cannot perform the trips *j* and *b* simultaneously. This case is observed between the 1st and 5th trips in Figure 3.3, these two trips cannot be taken by the same vehicle.

If these three relations can be observed any two trips, *i* and *j*, these trips would be the ones to be taken by the same vehicle. Then, job j will be added to set Q'_i, and job *i* will be added to the set Q'_j.

3.3 Model Assumptions

Before mathematical modeling of the problem, the conditions due to the organization and the assumptions for modeling should be restated briefly. Examination of these subjects will provide for a better understanding of the problem's context.

i. The upper bound value for the passenger demands in nonsplit trips is equal to the capacity of the largest vehicle.

ii. It is assumed that we have enough number of vehicles in each size.

iii. It is assumed that the trip times are deterministic and constant.

Besides the assumptions, there are some constraints due to the organizational structure. These include following:

i. The vehicles can transport between only two facilities, that is, only the passengers of a single trip can be carried with the same vehicle.

ii. Some trips are nonsplit due to the organizational conditions. That is, they have to be performed with a single vehicle.

iii. The postponement, delay or cancellation of the trips is not possible. All the trips should be taken in the time interval assigned to them.

The data of the problem include:

i. *Daily program*: The information about the exact timing of the activities, the facilities that the transportation will take place in between, and also the number of passengers to be transported.

ii. *The distance and the route between the facilities:* Since the distance of all trips is constant and the distances are known, time intervals of the trips are known in advance. In addition to this, the arc times of the trips is also known beforehand.

iii. *Capacity and cost of vehicle:* The information about the vehicles is known in advance. It has been already stated that there are enough number of vehicles in each size.

3.4 Mathematical Formulation

Parameters

- $i = 1,2,.....,I_1, I_1+1,.....,I_2, I_2+1,.....,I_k$: Vehicle index

  $(I_k - I_{k-1})$ is an upper bound on the number of available vehicles of type *k*.
- j = 1, ……,*J*: Trip index
- D_j : Number of passengers on trip *j*
- r_j : Departure time (ready time) of trip *j*.
- p_j : Trip duration (processing time) of trip *j*
- c_i : Capacity of vehicle *i*
- f_i : Daily fixed cost for vehicle *i*

- v_i : Overtime cost per time unit for vehicle i
- S : Daily regular time unit available for each vehicle
- O : Daily available overtime units for each vehicle
- Q'_j : Incompatibility set of trip j – The set of trips which can not be made by a vehicle, if that vehicle takes trip j.

  $Q'_j = \{b > j : r_j \le r_b \le r_j + p_j + a_{jb} \text{ or } r_b + p_b - r_j > S + O\}$

  - a_{jb} (arc time) is the time it takes for a vehicle to go from the arrival point of trip j to the departure point of trip b.
- T : The set of indexes of trips in which must use only one vehicle
- M: A large integer number. ($M = J$)
- N: A large integer number ($N = \max(r_1 + r_j + p_j)$)

Decision Variables

- $y_i = \begin{cases} 1, & \text{if vehicle } i \text{ is used} \\ 0, & \text{otherwise} \end{cases}$

- $x_{ij} = \begin{cases} 1, & \text{if vehicle } i \text{ takes trip } j \\ 0, & \text{otherwise} \end{cases}$

- o_i : Overtime usage of vehicle i

Mathematical Model

Objective Function:

$$\min \ z = \sum_{i=1}^{I_k} (f_i y_i + v_i o_i) \tag{3.1}$$

Subject To:

$$\sum_{i=1}^{I_k} c_i x_{ij} \geq D_j, \qquad \forall j \tag{3.2}$$

$$x_{ij} + x_{ib} \leq 1, \qquad \forall\ i, j, \text{ and } b \in Q'_j \tag{3.3}$$

$$\sum_{j=1}^{J} x_{ij} \leq M\, y_i, \qquad \forall i \tag{3.4}$$

$$(r_b + p_b)x_{ib} - r_j x_{ij} \leq S + o_i + (1 - x_{ij})N \qquad \forall\ i, j, \text{ and } b \notin Q'_j \tag{3.5}$$

$$\sum_{i=1}^{I_k} x_{ik} = 1 \qquad k \in T \tag{3.6}$$

$$x_{ij}, y_i \text{ binary}, \quad \forall i, j \tag{3.7}$$

$$o_i \geq 0 \qquad \forall i \tag{3.8}$$

Constraints

The objective function in (3.1) minimizes the total cost of the vehicles. If vehicle *i* is used, the decision variable y_i take value of 1 and the fixed cost of the vehicle must be tolerated. If the vehicle performs overtime, its overtime period is multiplied by the unit overtime cost and added to the objective function.

Constraint set (3.2) ensures the assignment of the adequate number of vehicles that transport all the passengers in each trip. This constraint set controls that the total amount of the capacities of the entire vehicles to be assigned to a trip is more than the demand of the trip.

The constraints in constraint set (3.3) are called as the spread time constraint. If vehicle *i* is assigned to the trip *j*, the vehicle will be unable to take the other trips which are in the incompatibility set (Q'_j) of this trip. By way of this constraint, if the vehicle takes a trip *j* ($x_{ij} = 1$) it is ensured that the trip *b* ($x_{ib} = 0$) in the incompatibility set (Q'_j) not to be taken.

Constraint set (3.4) is used to find out whether the vehicle is used or not. The main reason to use this constraint is to add the fixed cost of the vehicle to the objective function on condition that the vehicle is assigned. In this model, *M* value is equal to the number of the trips to be taken in daytime.

Constraint set (3.5) is used for determining the total overtime period of the vehicle, if it goes overtime. For this purpose, the time difference between the first and last trip is compared with *S* value. If this difference is bigger than the *S* value, addition of the exceeding time to the objective function as overtime cost is enabled. The *N* value here is equal to the difference between the ready time of the vehicle with the earliest ready time and deadline of the vehicle with the latest deadline.

Constraint set (3.6) enables the assignment of a single vehicle to the trips that are required to be taken by a single vehicle.

Constraint sets (3.7) and (3.8) enable the decision variables to be nonnegative and binary.

3.5 Optimal Solution

The mathematical model of the problem is *mixed integer programming model.* The number of constraints in the model changes depending on the number of trips and the number of vehicles. It can be thought that the solution period may extend with rising number of trips and vehicles in the problem, since the decision variables

in the model are mostly binary. The mathematical model is attempted to be solved optimally with two different optimization programs.

Firstly, formulation in LINGO 8.0 is attempted to be solved with the optimization software by a P4 2.4 MHz. 256 Mb computer. However, it has been deduced that daily optimum schedule of organizations of a daily program that includes more than 100 trips can not be ascertained with this software in an acceptable time period. The result is that because the problem is daily, maximum time period allowable is one day, but within this time solving the problem is not possible with this software.

Secondly, problem is modeled in GAMS 20.2 optimization program by the use of CPLEX SOLVER and it has been attempted to be solved by the same computer. There has not been any change in the result. It has also been extrapolated that even though the performance is better than LINGO optimization software, in the situations like when there are over 100 trips in a day; this program will not be able to as certain the optimum solution of the problem within an acceptable time period, as well. LINGO and GAMS models of an sample problem and the preparations of the data sets inside these programs are presented in Appendix A.

As it has been pointed in the beginning, the existence of binary decision variables and the large number of constraints may be indicated as the reasons demonstrating why the problem cannot be solved in reasonable times by the optimization software. Many problems especially with Constraints (3.3) -that is spread time constraint- are presented as NP – complex and NP – hard in literature. (See also Fischetti et al. 1987)

The question that whether the problem is NP – complex or NP - hard is beyond the scope of this study. Yet, by using the optimization software, it has been inferred that the problem can not be solved within a reasonable time period. Because of that, some methods should be developed that will enable to figure out the problem with more trips in a shorter time. In this problem, heuristic search techniques will be utilized as solution methods.

3.6 Heuristics

Since the problem can not be solved within a reasonable period of time, heuristic solution methods have been developed which are capable of producing suboptimal or near optimal solutions in shorter times. Each of these methods has been based upon various criteria. Three heuristics approaches which are established on different algorithms have been developed. Each different heuristic is solved by two distinct solutions in itself; in the first solution, vehicles are used only during their daily regular time limit (*S*). In the second solution, vehicles are forced to make overtime; i.e. each vehicle is used in a period of length (*S+O*).

In this way, a total of six different solution methods are constructed. The first one of the three different heuristics methods is the *Vehicle Based Heuristic (VBH)* method, which performs the assignments considering vehicles one by one. The second one is the *Trip Based Heuristic (TBH)* method, which performs the assignments considering the trips one by one. The third one is the *Group Based Heuristic (GBH)* method, where the trips are grouped and then assigned with respect to this group order. The process of each method and their detailed algorithms are presented below.

3.6.1 Vehicle Based Heuristic

In this method, the assignments are based on vehicles. A daily work schedule of the vehicle is formed by assigning the vehicle to all trips that they are able to do within the performance time of the vehicles. The performance time may be *S* or *(S+O)*, depending on the type of algorithm, as will be stated below. Thus, the daily work schedule of each assigned vehicle is done, and it continues until the process is over. The general algorithm of the method is as follows.

1. Determine the first trip to take.

2. Assign a new vehicle to the chosen trip.

3. Form the trip set that can be assigned to the vehicle.

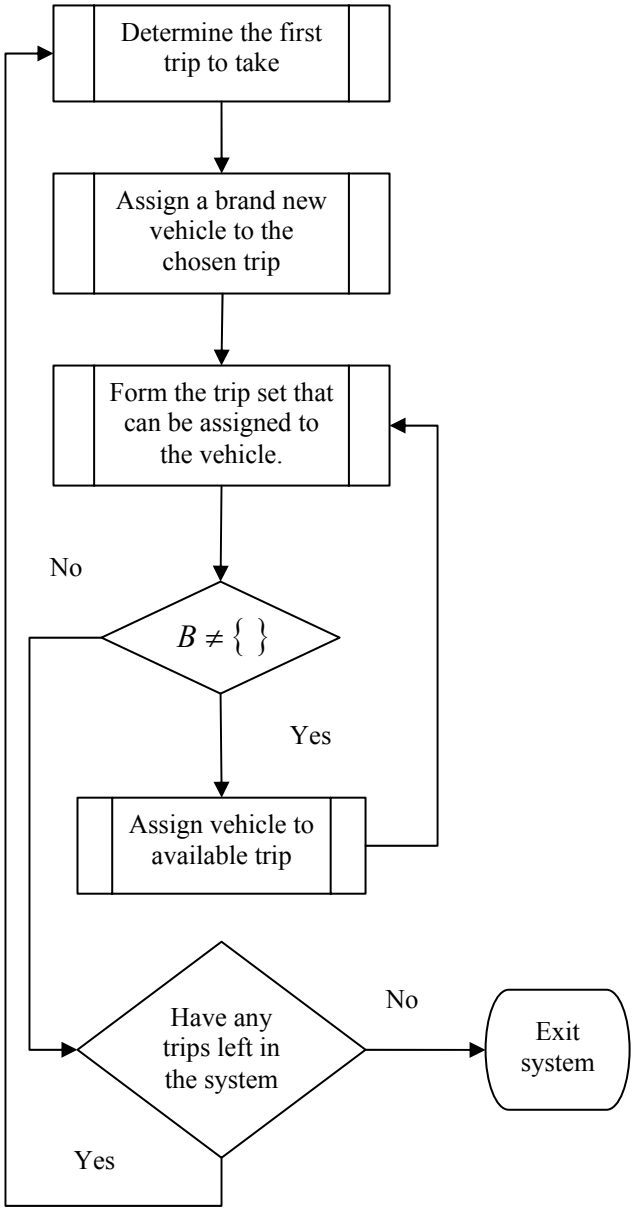
4. If there are any trips (i.e. the trip set is nonempty) which can be made by the same vehicle after assigning the previous trip and go to step *5*, otherwise go to step *6*.

5. Assign the vehicle to the most proper trip with respect to its performance criteria. Then follow step *3* again.

6. If any trip is left in the system, go to step *1*; if not stop.

In this method, the problem is solved by two different approaches. In the first approach overtime is not allowed, in the latter one all vehicles perform overtime. The only difference in the algorithm is that in the first approach daily available overtime units for each vehicles equals to zero ($O=0$).

The general flowchart of the method is displayed in Figure 3.4.



B: The set of trips which can be made by same vehicle after assigning

Figure 3.4 Flowchart of Vehicle Based Heuristic

The detailed algorithm of heuristic method and its parameters are as follows:

Parameters:

- F : The set of unused vehicles.
- T_i' : The set of nonsplit trips that cannot be assigned to vehicle *i*.
- *S:* The regular time limit of a vehicle.
- *O:* The maximum over time unit of a vehicle.

  $O = 0$; for vehicle based heuristic without overtime (*VBH*)

  $O > 0$; for vehicle based heuristic with overtime (*VBHO*)
- Q_j : The *Compatibility Set* of trips which the vehicle to be assigned to *trip j* is able to take within daily usage time period (*S*+*O*).
- k_i : The index of the first assigned trip of vehicle *i*.
- w_{ijb} : The idle time of the vehicle *i* in case that it takes trip *b* after its trip *j* whether on regular or on overtime.
- w'_{ijb} : The idle time of the vehicle *i* in case that it takes trip *b* after its trip *j* on overtime.
- p'_{ijb} : The usage time period of the vehicle that performs overtime after its trip *j*.
- R_{ijb} is the performance criteria for assignment of vehicle *i* to trip *b* after trip *j*.

R_{ijb} = cost of idle time on regular time + cost of idle time on overtime
+ cost of unused seats on regular time + cost of unused seats on overtime

$$R_{ijb} = (w_{ijb} - w'_{ijb})\frac{f_i}{S} + w'_{ijb} * v_i + \frac{\max(c_i - D_b, 0)}{c_i}((p_b - p'_{ijb})\frac{f_i}{S} + p'_{ijb} * v_i)\ ;$$
$$\forall b \in (Q_j - T_i')$$

R_{ijb}, is calculated for every trip *b* which vehicle *i* can take after its trip *j* and it is regarded as an assignment criterion. Vehicle *i* is assigned to trip *b* that gives

$\min R_{ijb}$. This ratio enables the vehicle to be assigned to the busiest trip with minimum delay. The vehicle is assigned to the trip with the lowest free seat cost and shortest delay time this way.

Detailed Algorithm

S0: Initialization: The process of necessary preparations for algorithm.

i. Arrange the vehicles by their sizes in descending order.

ii. Create a list of trips by arranging the trips in a chronological order according to their ready time.

iii. Form a T_i' set for each vehicle; $T_i' : \{j \in T : c_i < D_j\}$

iv. Form a Q_j set for each trip;

$$Q_j = \{b > j : r_j + p_j + a_{jb} \le r_b \ or \ r_j + r_b + p_b \le S + O\}$$

v. Calculate the w_{ijb} values for each trip; $w_{ijb} = r_b - (p_j + r_j), \quad \forall b \in Q_j$

vi. Calculate the w'_{ijb} and p'_{ijb} values for each trip;

a. If there is not any overtime; w'_{ijb} =0 and $p'_{ijb} = 0$

b. If there is overtime;

$$w'_{ijb} = \max(r_b - (r_{k_i} + S), 0), \quad \forall b \in Q_j$$

$$p'_{ijb} = \max(r_b + p_b - (r_{k_i} + S), 0), \quad \forall b \in Q_j$$

S1: Choose the first trip from the list which is not done and let it be trip *j*.

S2: Assign an unused vehicle to the first chosen trip.

A. If $D_j \ge \max(c_i)$, then assign the vehicle *i* which is $\max(c_i) \ i \in F$ to trip *j*. If not, $U_i = \{c_i - D_j\} \ \forall i \in F$. Calculate for all vehicles *i* belonging to set *F*, the quantity of passengers who cannot transport in the trip *j* in the event of vehicle *i* assignment to this trip. In the trip *j* that gives $U_i \ge 0$ value, assign the *vehicle*

i with $\min(U_i)$ to the trip j. Thus, the assigned vehicle's free seats will be in minimum number.

B. Delete vehicle i from the set F.

C. $k_i = j$.

S3: Reduction of the demand of the assigned trip as much as the capacity of the vehicle assigned to the trip.

A. $D_j = \max((D_j - c_i),0)$

B. If $D_j = 0$, then delete trip j from list of trips.

S4: Assignment of the vehicle in use to its new trips within the validity period.

If $(Q_j - T_i') \neq \varnothing$ then

i. Calculate the R_{ijb} ratio for every trip b in this format: $b \in (Q_j - T_i')$

ii. Assign vehicle i to trip b whose R_{ijb} is minimum.

iii. $b = j$ (Consider the newly assigned trip b as trip j).

iv. Go to step S3.

Otherwise go to step S5.

S5: If there is any work to be done in list of trips, go to S1; if not exit the system.

According to the types of the vehicles employed, the total fixed costs are calculated. In order to count the overtime costs, we calculate the time length between the ready time of the first trips and the deadline of the last trips, and then subtract the S value from this time so as to get the overtime period.

3.6.2 Trip Based Heuristic

In this method, the assignments are trip based. The trips are ordered according to their ready times and a combination of vehicles with the minimum number of vehicle is assigned to those trips. All assignments are conducted in this way until all the trips are taken. For the assignments, the priority belongs to the vehicles that are used before. In this method, the schedule is formed as trips. The general algorithm of the method is as follows:

1. Determine the first trip to take.
2. Assign enough vehicles to meet the demand of the trip (If possible from used vehicle).
3. If there is a trip left go to the first step, if not exit the system.

The flowchart created according to this algorithm is displayed in Figure 3.5.

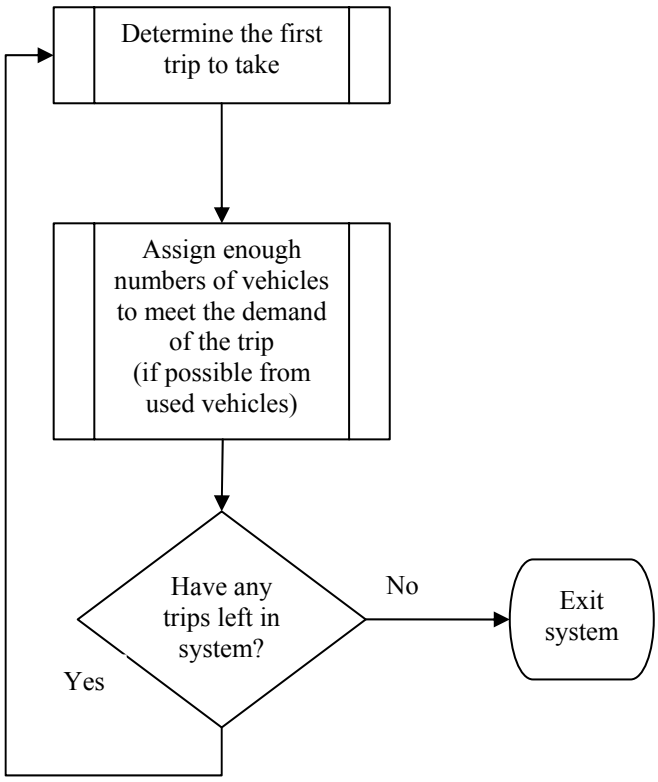


Figure 3.5 Flowchart of Trip Based Heuristic

In this method, there are two approaches like the vehicle based heuristic method, depending on whether the vehicle performs overtime or not. The detailed algorithm is stated below.

Q_{1j} is the *Compatibility Set* of trips which the vehicle to be assigned before *trip j* is able to take within daily usage time period (*S*+*O*). Other notations are the same as VBH and VBHO.

Detailed Algorithm

S0: Initialization: The process of necessary preparations for algorithm.

- **i.** Arrange the vehicles by their sizes in descending order.
- **ii.** Create a list of trips by arranging the trips in a chronological order according to their ready time.
- **iii.** Form a T_i' set for each vehicle; $T_i' : \{ j \in T : c_i < D_j \}$
- **iv.** Form a Q_{1j} set for each trip;

$$Q_{1j} = \{ b < j : \ r_j + p_j + a_{jb} \le r_b \ \ or \ \ r_j + r_b + p_b \le S + O \}$$

- **v.** Calculate the w_{ijb} values for each trip; $w_{ijb} = r_b - (p_j + r_j), \quad \forall b \in Q_{1j}$
- **vi.** Calculate the w'_{ijb} values for each trip;
  - **a.** If there is not any overtime; w'_{ijb} =0 and $p'_{ijb} = 0$
  - **b.** If there is overtime;

$$w'_{ijb} = \max(r_b - (r_{k_i} + S), 0), \quad \forall b \in Q_{1j}$$

$$p'_{ijb} = \max(r_b + p_b - (r_{k_i} + S), 0), \quad \forall b \in Q_{1j}$$

S1: Choose the first trip from the list of trips and let it be trip *j*.

S2: If there is not any other trip to take with the chosen trip *j* ($Q_{1j} = \varnothing$) then follow the step S4, otherwise goes to step S3.

S3: If the available vehicle *i* is $(j \notin T_i')$; namely if the previously assigned vehicle is able to take trip *j*.

i. Calculate R_{ijb} ratio.

ii. Form an "Assignment" set by ordering the R_{ijb} values in ascending order.

a. If the assignment set is $\neq \varnothing$,

b. Assign the first vehicle *i* in the row and delete it from the "Assignment" set.

c. $D_j = \max((D_j - c_i),0)$.

d. If $D_j \neq 0$, then go to step a.

iii. If $D_j \neq 0$, then goes to step S4, if not goes to step S5.

S4: Assignment of an unused vehicle to the chosen first trip.

A. If $D_j \geq \max(c_i)$, then assign the vehicle *i* which is $\max(c_i)$ $i \in F$ to trip *j*. If not, $U_i = \{c_i - D_j\}$ $\forall i \in F$. Calculate for all vehicles *i* belonging to set *F*, the quantity of passengers who cannot transport in the trip *j* in the event of vehicle *i* assignment to this trip. In the trip *j* that gives $U_i \geq 0$ value, assign the *vehicle i* with $\min(U_i)$ to the trip *j*. Thus, the assigned vehicle's free seats will be in minimum number.

B. Delete vehicle *i* from the set *F*.

C. $k_i = j$.

D. $D_j = \max((D_j - c_i),0)$

E. If $D_j \neq 0$ then go back to A.; if not go to step S5.

S5: If there is any work to be done in the list of trips, goes to S1; if not exit the system.

Overtime costs are calculated as in the vehicle based heuristics.

3.6.3 Group Based Heuristic (GBH)

This heuristic method has a different approach. In this method the trips are divided into two main groups. The first group is composed of nonsplit trips while the latter group is formed with split trips. Each group has its subgroups based on their vehicle capacity. The assignments begin with the groups belonging to nonsplit trips and during the time left from the trips, the vehicle of this group is assigned to the trips of other groups respectively. In this way the vehicle scheduling is attempted to be made. The aim of this grouping is assignment of the vehicle primarily to the trips appropriate for its capacity. By assigning the convenient vehicle to the trips of other groups, the idle time of the vehicle is minimized. The algorithm of this method is briefly like this:

1. Group the trips as split and nonsplit.

2. Classify subgroups according to the capacity of the vehicles.

3. Select a trip in a subgroup, repeat steps 4, 5 and 6 for all trips, when no trip is left, exit the system.

4. Assign the appropriate vehicle in the group primarily to the trips within the same group.

5. If there are trips in the other groups with the idle time of the assigned vehicle, then assign the vehicle to those trips.

6. Go back to step 3.

In Figure 3.6 the general flowchart of the method is demonstrated. Herein after, the detailed algorithm of the method is presented.

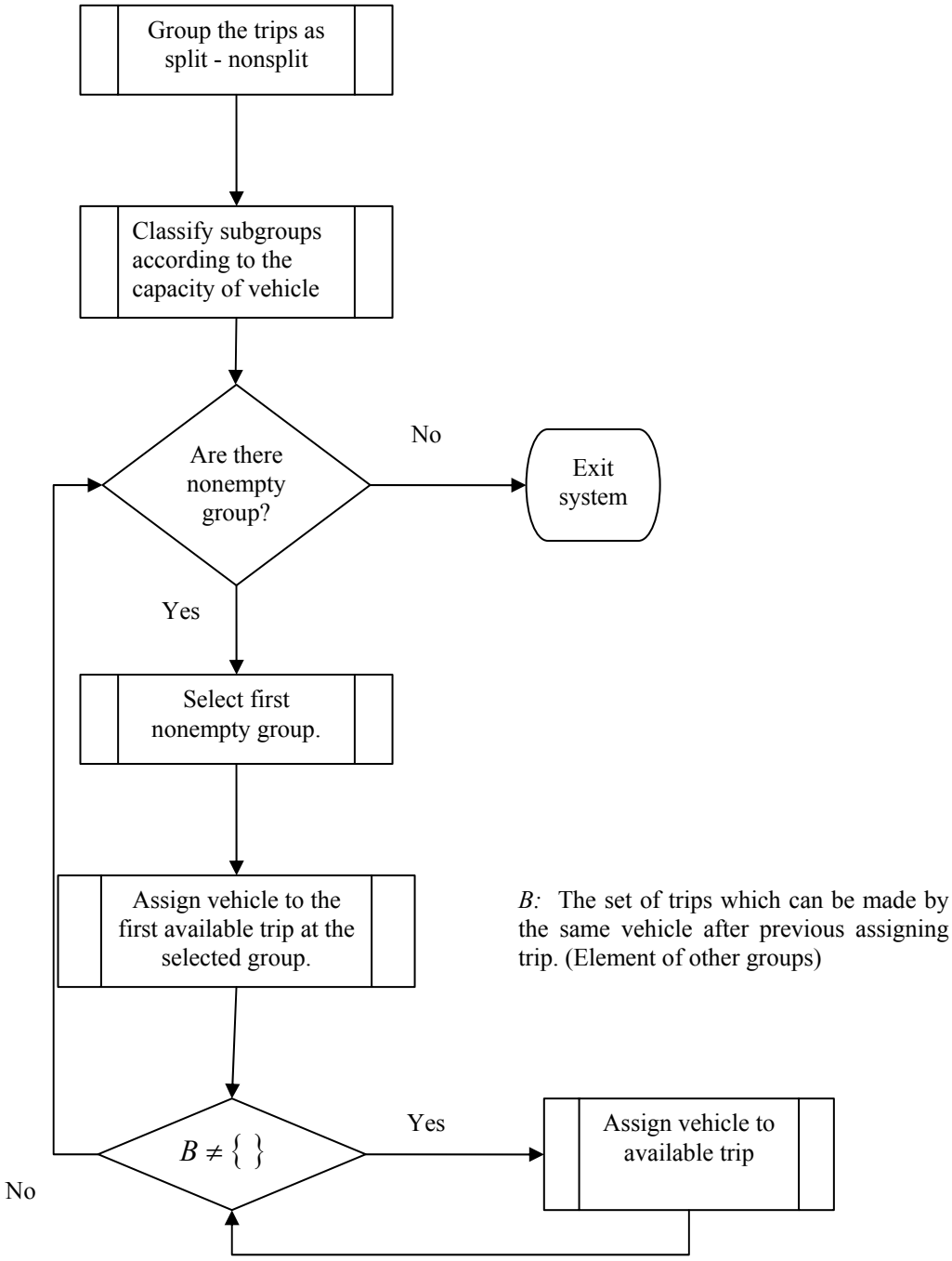


Figure 3.6 Flowchart of Group Based Heuristic

Algorithm:

S0. Initialization:

a. Index the trips in chronological order of their ready times.

b. Index vehicles in decreasing order of their capacities.

c. Form two sets, *A* and *B*. Gather all trips in which must use only one vehicle (nonsplit trips), in set *A*. Put all remaining trips in set *B*.

d. Group the trips in sets *A* and *B* separately as follows in (2*K*+1) groups:

$Group\,1_A = \{j \in A : c_1 \geq D_j > c_2\}$, $\quad Group\,1_B = \{j \in B : c_1 \geq d_j > c_2\}$

$Group\,2_A = \{j \in A : c_2 \geq D_j > c_3\}$, $\quad Group\,2_B = \{j \in B : c_2 \geq d_j > c_3\}$

... ...

$Group\,K_A = \{j \in A : c_k \geq D_j\}$, $\quad Group\,K_B = \{j \in B : c_k \geq D_j\}$

$Group\,K+1_B = \{j \in B : c_1 \leq D_j\}$

The elements in $Group\,K+1_B$ are the trips that must be divided.

S1. Scheduling of Trips in set *A* (Nonsplit Trips):

Consider the nonempty groups sequentially. For each group k_A, $k = 1,...,K$:

A. Let *f* and *l* be the first and last trips of the group, respectively.

B. If $(d_l - r_f) \leq S$, the minimum number of vehicles required to carry out the trips in this group is equal to the maximum number of trip overlaps in the group (Hashimoto and Stevens, 1971). This theorem can not be exactly applied because in addition to overlapping trips in this problem, some trips are also taken by the same vehicle due to arc time. Therefore, with the light of this theorem, a new corollary is proposed.

Corollary 1: If the $(d_l - r_f) \leq S$ condition is provided for one group, incompatibility sets which provide $Q'_j = \{b > j : r_j \leq r_b \leq r_j + p_j + a_{jb}\}$

conditions for all trips in that group are formed. The trip is determined, whose number of elements of incompatibility set is maximum $(\max(s(Q'_j))$. The total number of overlapping trip of this group is equal to $\max(s(Q'_j)+1$. The number of vehicle for the optimum scheduling of this group is equal to the number of overlapping trips. By employing as many vehicles as the overlapping trips, optimum scheduling of the trip in this group can be formed. While assigning the vehicles to the trips, if separate vehicles are assigned to the overlapping trips, optimum scheduling is acquired.

C. Else, assign job *f* to a new vehicle of type *k* (call this vehicle *i*). The ending time of the regular shift for vehicle *i* then becomes ($r_f + S$). Determine the trips in Group k_A that can be assigned to vehicle during its regular shift, i.e., determine set $Q_i = \{j \in k_A : r_j \geq d_f, d_j \leq r_f + S\}$. Do the following for the trips in Q_i:

i. $W_i = \{f\}$. W_i represents the set of trips assigned to vehicle *i*.

ii. Consider the trips in Q_i sequentially. At each ready time, add the arriving trip to set W_i. If the vehicle is not available at the ready time of a trip, remove the trip with the latest deadline from set W_i (but not trip *f*).

iii. Remove the trips in set W_i from consideration.

D. Check if any other trip can be assigned to the vehicle(s) scheduled in Steps (B) or (C):

i. Start the checking procedure with Group $K+1_B$ (if not empty). If there exists a trip (*j*) in that group that can be assigned to the vehicle in its regular shift, then split the trip in two. D_j becomes C_1, while D_{j+1} become ($D_j - C_1$). Assign trip *j* to the vehicle. Place trip $J+1$ in the appropriate group. Update the total number of trips as $J+1 (J \leftarrow J+1)$ After considering all trips in Group $K+1_B$, proceed with groups $1_B, 2_A, 2_B, 3_A$, ...

ii. Remove all assigned trips from consideration.

S2. Scheduling of Trips in set *B*:

Consider the nonempty groups in decreasing order of their number of elements (cardinalities). For each group k_B, $k = 1,...,K$: :

A. Let *f* and *l* be the first and last trips of the group, respectively.

B. If $(d_l - r_f) \le S$, then by using corollary 1, overlapping trips are determined and the scheduling of this group can be formed when vehicles as many as the number of trips are employed. In this process, the fact that vehicles should primarily be assigned to the overlapping trips must be taken into consideration.

C. Else, assign job *f* to a new vehicle of type *k* (call this vehicle *i*). The ending time of the regular shift for vehicle *i* then become $(r_f + S)$. Determine the trips in Group k_B that can be assigned to vehicle during its regular shift, i.e., determine set $Q_i = \{j \in k_A : r_j \ge d_f, d_j \le r_f + S\}$ Do the following for the trips in Q_i:

i. $W_i = \{f\}$. W_i represents the set of trips assigned to vehicle *i*.
ii. Consider the trips in Q_i sequentially. At each ready time, add the arriving trip to set W_i. If the vehicle is not available at the ready time of a trip, remove the trip with the latest deadline from set W_i (but not trip *f*).
iii. Remove the trips in set W_i from consideration.

D. Check if any other trip can be assigned to the vehicle(s) scheduled in Steps (B) or (C):

i. Start the checking procedure with Group $K + 1_B$ (if not empty). If there exists a trip (*j*) in that group that can be assigned to the vehicle in its regular shift, then

split the trip in two. D_j becomes C_1, while D_{j+1} become ($D_j - C_1$).. Assign trip *j* to the vehicle. Place trip $J+1$ in the appropriate group. Update the total number of trips as $J+1(J \leftarrow J+1)$. After considering all trips in Group $K+1_B$, proceed with groups $1_B, 2_B, 3_B, \ldots$. Remove all assigned trips from consideration.

This method is solved by two distinct approaches, as well. These approaches require the vehicles to work overtime or not as in other approaches. If this algorithm is figured out with the use of the overtime, the algorithm is solved by placing *S* with *S*+*O*.

The overtime cost calculations are similar to the other heuristics.

CHAPTER FOUR

EXPERIMENTATION

4.1. Introduction

There is an important point which is as significant as the structure and processing of the solution methods; the fact that how effective the solution methods can perform during a real problem. The points that for how many trips can we find optimum solution with using mathematical model and to figure out how close are the result of the heuristics solution methods and we could lay a path for interpreting the performances of the methods. Therefore, a problem appropriate to the real problem generated by using a determined experimental design and evaluating the results found with solution methods is required.

In this chapter an experimental design appropriate to the problem is prepared and sample problems are produced. The optimum values of these problems are found by mathematical models, upper bounds with heuristic methods and lower bound values by use of algorithms. The performances of the methods are evaluated by taking these values into consideration.

4.2. Design of Experiment

By using experimental design, appropriate samples to the problem are generated. During composing the experimental design, the ready information within the system, that is to say, the data known in advance are attempted to be created by a certain systematic. Consequently, some variables are formed by use of a certain distribution and some are formed by use of the data from real life. The experimental design used during the formation of sample problems is as follows:

While generating the problem, a daily working amount is assumed as two hundred units. It has been also assumed that there are four types of vehicles. The capacities of these different types, their usage periods and their fixed and variable costs are displayed in Table 4.1. In addition to this, there is enough number of vehicles in each size.

Table 4.1 The Variables Depend on Vehicle

Vehicle Types	c_j (person)	S (time unit)	O (time unit)	f_j (YTL)	v_j (YTL)
I	45	100	50	1200	20
II	27	100	50	900	15
III	16	100	50	720	12
IV	10	100	50	600	10

While designing the variables dependent on trips, the numbers of trips are determined first. The problem are generated starting from n = 20 trips and increasing by increments of 10 to n = 100 trips. We generated 10 random instances of each problem combination. Thus, nine separate groups based upon the number of trips are formed. It has been supposed that 40 % of these daily trips are nonsplit trips. Daily demand is prepared by using U [8, 100] discrete uniform for split trips and U [5, 30] discrete uniform distribution for the nonsplit trips. In this way, the demands of the nonsplit trips are less than the capacity of the largest vehicle.

The ready times of the trips are generated by two different methods for each number of daily trips. Hence, for every nine group there are two subgroups which make eighteen separate data sets. The ready times are created by discrete uniform distribution in the first method and by uniform distribution with peak periods in the second method. The distribution of ready time is stated below:

i. Ready Time Type 1 (r = 1): Discrete Uniform distribution U [0, 200]
ii. Ready Time Type 2 (r = 2): Uniform with peak periods:
    a. 30% of the ready times ~ U [30, 40]

b. 30% of the ready times ~ U [130, 140]
c. 40% of the ready times uniformly random in the ranges [0, 29], [41,129], and [141, 200].

The processing times of the trips are generated by two different methods like their ready times. Thus, there are two subgroups of eighteen separate data sets which make thirty six separate data sets. The processing times are created by using only one discrete uniform distribution nonsplit while split trips are generated by using two different discrete uniform distributions.

i. Processing Time Type 1 (p = 1): For split trips U [5, 10] and for nonsplit trips U [5, 40]
ii. Processing Time Type 2 (p = 2): For split trips U [2, 20] and for nonsplit trips U [5, 40]

Arc time is determined by using discrete uniform distribution, as well. The arc time between two trips is generated by use of U [0, 10] distribution. With this design, four combinations are made for each number of daily trips. Ten sample problems are generated from each combination. Thus, from nine separate trip sets, in four time combinations and from ten trips, three hundred sixty total sample problems are obtained.

So as to form the sample problems, an algorithm appropriate to experimental design is created in C coded program and three hundred sixty sample problems are composed by utilizing this algorithm. In that way sample problems could be constituted automatically.

4.3 Optimal Solution of Mathematical Model

The mathematical model of the problem was mentioned in chapter three. It was also mentioned in the same chapter that the mathematical model prepared here has been modelled by making use of LINGO 8.0 and GAMS 20.2. In the previous

attempts before experimental design has been performed, the GAMS 20.2 models had been determined to yield better and more productive results. Furthermore, another advantage of GAMS 20.2 was the opportunity to choose a solver. Therefore, the optimum results of the prepared sample problems has been researched by making use of GAMS 20.2 and setting a one hour time limit.

During the solution, CPLEX solver has been used in GAMS 20.2. The reason to use CPLEX solver for the solution was that CPLEX solver is an algorithm designed to solve large, structurally complex and difficult mixed integer programming (MIP) problems. The problem to be dealt here is a problem with the same structure. GAMS 20.2 have been chosen due to the advantage to make use of CPLEX solver for the solution and thus better results could be achieved.

GAMS models of all the attempts have been created and the attempts have been solved with a P4 2.4 MHz. 256 Mb. computer. However, in the attempts of only twenty and thirty trips with one-hour limit an insufficient number of problems optimum results could be achieved. In the problems with more trips no optimum result could be achieved. Computer could hardly operate in the solutions of the algorithms. As mentioned above, this demonstrates the complexity of the problem and the difficulty in the solution of the problem by making use of optimization programs.

However, so as to achieve more optimum results than one – hour limit, five attempts out of each combination of twenty and thirty trips have been solved without any time limit and the optimum values of these attempts have been found. This will help to evaluate the performance of the upper and lower bound methods which will be covered in the further sections.

4.4 Upper Bound – Solutions of Heuristics

The solutions which found in heuristics for the samples problem is defined as upper bound (k). Each solution found by way of heuristic methods is an upper

bound for the problem. The reason is that the value found by way of heuristics methods is always larger than or equal to optimum.

In the previous chapter it has been mentioned there are three different heuristic solution methods for the problem and the algorithms of these methods have been stated. In addition, there has been two different solution approaches for each method. According to this, there are six different upper bound solutions ($k = 1, 2, ..., 6$). In order to find these values the samples require to be solved using these heuristics. So as to render these solutions, software programs for six different methods have been produced belonging to the algorithms of heuristics in C programming language with usage of DEV – C ++ compiler. The entire sample attempts have been solved with the software programs coded for heuristic solution methods. Unlike the optimization program, the sample problems have been solved with the coded software programs in a very short time, which demonstrates that the algorithms can solve the problem in a short time. Larger problems can be solved with this C program faster.

The fast solution of the algorithms with C provided an advantage to obtain the upper bound of the problem. As stated before, the problem had been required to be solved daily and there had been no optimum solution of the problem with the optimization programs in a reasonable period of time. However, as the solution period of algorithms coded with C program is quite short, the problem is solved for six different methods separately and 6 different upper bounds (UB_k) are found. These values are as follows:

- UB_1 = Solution of trip based heuristics without overtime;
- UB_2 = Solution of trip based heuristics with overtime;
- UB_3 = Solution of vehicle based heuristics without overtime;
- UB_4 = Solution of vehicle based heuristics with overtime;
- UB_5 = Solution of group based heuristics without overtime;
- UB_6 = Solution of group based heuristics with overtime;

With these six different upper bounds, a set of upper bounds (UBS_L) is formed for each sample ($l = 1, 2, ..., 360$). $UBS_l = \{UB_{l1}, UB_{l2}, UB_{l3}, UB_{l4}, UB_{l5}, UB_{l6}\}$

By selecting the minimal value ($\min(UBS_l)$ out of UBS_L, it could be regarded as the least upper bound found with heuristics. Thus, instead of comparing the heuristic methods and seeking the best method, the problem is solved with the entire heuristics, the set of upper bounds is formed and a closer result to optimum is achieved by selecting the minimal value. The minimal upper bound in the set of upper bounds is called "supremum". Sundaram (1999) defines this concept as: "The supremum is defined to be the least upper bound of the set of upper bounds of problem." (p. 14).

The important task here is the examination of the relation of the supremum with the optimum solution. The closer the supremum to optimum the more successful the heuristic solution methods are. Yet, as stated in the previous chapter, not enough optimum solution could be found to make comparison. Therefore, lower bounds have been attempted to be found with the some methods which will be explained in the next section.

4.5. Lower Bounds

In this section, we present methods to find lower bounds (*LB*) that are used to evaluate the upper bound performance. Five different methods are employed to find the lower bound for the problem, which are as follows.

LB 1: In this method, a vehicle usage cost of an average one time unit (v') is calculated based on the capacities, fixed costs, variable costs, available regular and overtime unit per day of the vehicles. A cost is found for the works to be done by the multiplication of the processing times of regular trips and newly found average cost. The formulation of this method is as follows.

v' : Average cost of a vehicle per time unit: $v' = \dfrac{\sum_{i=1}^{K} \dfrac{c_i (f_i + v_i O)}{S + O}}{\sum_{i=1}^{K} c_i}$;

$LB_1 = z$: Total Cost: $z = \sum_{j=1}^{J} p_j v'$.

Thus, the cost (LB_1) which is $LB_1 \le opt(z)$ according to the total processing time is found.

Proof: The cost (LB_1) is found by way of *LB 1* method and it is a smaller value than optimum solution ($LB_1 \le opt(z)$); besides, this method has a big gap between optimum. The main reason is that the idle time of the vehicles and the arc times between the trips have been disregarded while LB_1 is found. Furthermore, the capacities of the trips and vehicles have not been taken into account. Also, another reason for the solution found with this method to be infeasible and smaller than the optimum is that overlapping trips have been disregarded because the assignments are performed merely according to the processing times. Normally, it is not possible to compile daily feasible vehicle schedules without using processing times, capacities and demands.

LB 2: In this method, an average seat cost for one time unit is found according to the average usage times of the vehicles. The cost of trip is calculated according to the demand and processing time of the trips to be taken. The cost for the whole day is calculated with the sum of the cost of the trips. The formulation of this method is as follows:

v'' : Average cost of seat per time unit: $v'' = \dfrac{\sum_{i=1}^{K} \dfrac{(f_i + v_i O)}{S + O}}{\sum_{i=1}^{K} c_i}$;

$LB_2 = z$: Total Cost: $z = \sum_{j=1}^{J} D_j p_j v''$.

Thus, the cost (LB_2) which is $LB_1 \le LB_2 \le opt(z)$ according to the total processing times and demand is found.

Proof: In this method, the solution is always smaller than the optimum value, too; however, it is much closer to the optimum than the result found by first method ($LB_1 \le LB_2 \le opt(z)$), because the demands also affected the cost function. Similar to *LB 1*, the arc and idle times are disregarded in this solution. The overlapping trips are not taken into account. All the vehicles are accepted to take full capacity in all trips. There is no possibility to schedule a normal day this way, the result found with this method is infeasible at all.

LB 3: Third method of finding lower bound is grouping the tasks according to the demands and finding the number of vehicles according to the sum of the processing times of each group. In this method, the set (Q_k) of index of the tasks to be performed according to each vehicle types is formed. The tasks whose demands are larger than the vehicle with the highest capacity are divided into groups of the least number of vehicles. By this way, the required number of vehicles out of every vehicle class is found and the cost for the particular day according to this method is calculated by multiplying with the cost of use. Algorithm is as follows.

1**.** Sort all vehicles in descending order, i.e. $\max(c_k) = c_1$ *and* $\min(c_k) = c_K$ ($k = 1, 2, ..., K$).

2. Determine which vehicle fits to the current demand;

2.1 if $c_k \ge D_j$; then $j \in Q_k$;

2.2 if $c_{k-1} \ge D_j > c_k$; then $j \in Q_{k-1}$;

…….

2.(K-1) if $c_1 \ge D_j > c_2$; then $j \in Q_1$

2.(K) if $D_j > c_1$; then $j \in Q_1$ and $D_j = D_j - c_1$ and go to step 2.1.

3. Find minimum number of vehicle in each type $k, k = 1, 2, ..., K$,

$$W_k = \frac{\sum_{j \in Q_k} p_j}{S} \quad , \forall k \text{ and } j \in Q_k$$

4. Calculate total cost: $LB_3 = z = \sum_{k=1}^{K} W_k f_k$

The cost (LB_3) which is $LB_1 \le LB_2 \le LB_3 \le opt(z)$ is found according to the trips grouped with this algorithms and the use time of the trips in each group.

Proof: In this *LB 3* method, the overlapping trips, the idle times between the trips and arc times are disregarded as well. Therefore, the result found with this method is infeasible and it always has a smaller value than the optimum. The value found with this method (LB_3) has more accurate results to the optimum than the other two methods ($LB_1 \le LB_2 \le LB_3 \le opt(z)$). Its main reason is that each trip is considered separately during the classification of the trips according to the vehicles and the vehicle is not used 100% occupancy rate.

LBS 4: This method is the same as the *LB 3*, but the only difference here is the calculation of the cost of use. Here, instead of calculating the cost by finding the numbers of all types of vehicles and multiplying them with fixed cost, total processing times of the trip set for each vehicle type are found. This total cost for each group is calculated by way of usage cost for unit time. This cost is found according to the fixed and variable costs of the vehicles. Unit cost and total cost can be found as follows:

v_i''': Average cost per time unit of a vehicle type k: $v_k''' = \frac{f_k + v_k O}{S + O}$;

Total cost: $LB_4 = z = \sum_{k=1}^{K} \sum_{j \in Q_k} p_j v_k'''$ $\qquad j \in Q_k$.

The cost (LB_4) which is $LB_1 \le LB_2 \le LB_3 \le LB_4 \le opt(z)$ according to unit cost is found.

Proof: The total cost value found with this method (LB_4) is an infeasible solution because of the reasons explained in the previous method. However, a closer result is achieved to optimum compared to the previous method because the total cost is calculated by way of a one unit cost whose overtime period is included ($LB_1 \le LB_2 \le LB_3 \le LB_4 \le opt(z)$).

LBS 5: A lower bound has been attempted to be found by using heuristic solution algorithms which were explained in the previous chapter. The vehicle capacity and use cost has been modified while the algorithm has remained the same. During the solutions of the algorithms, the capacities of the entire vehicles are accepted to be equal with the capacity of the largest vehicle. The usage costs of the vehicles to be accepted equal with the capacity of the largest vehicle are equalized with the vehicle with the lowest use costs. That is to say, the capacities and the costs are as follows;

$$
\begin{aligned}
&* \; c_i = \max(c_k) \qquad \forall i, \\
&* \; f_i = \min(f_k) \qquad \forall i, \\
&* \; v_i = \min(v_k) \qquad \forall i.
\end{aligned}
$$

Totally six heuristic solution methods with two different solution approaches out of three different algorithms are solved with the vehicle set having these modifications and six different costs, in other words lower bounds, which need to be tolerated are found for each day. Their listing in order is as follows;

- LB_{51} = Solution of trip based heuristics without overtime;
- LB_{52} = Solution of trip based heuristics with overtime;
- LB_{53} = Solution of vehicle based heuristics without overtime;
- LB_{54} = Solution of vehicle based heuristics with overtime;
- LB_{55} = Solution of group based heuristics without overtime;

- LB_{56} = Solution of group based heuristics with overtime;

Proof: All of the six values found with this method give smaller values than the optimum. Its main reasons are that during the assignments, the capacities of the entire vehicles are accepted equal with the vehicle with the maximum capacity and the fixed and variable costs of this vehicle are accepted with the vehicle with the lowest capacity. As the cost is taken low, even though the daily scheduling turns out to be the optimum scheduling, the found cost would be less than normal. In addition, as the vehicles are accepted to be the vehicle with the highest capacity, they operate to overcome the unproductiveness of the assignment procedure of heuristic approaches by enabling the trip, which requires less capacity than the vehicle with the highest capacity, to be taken with a single vehicle in the demands they were assigned to. However, the cost of the optimum assignment would be higher than the one found with this method because there are different vehicles with different capacities and costs in the actual problem.

LBS 6: This lower bound (LB_6) is the Best node value whose solution is found by the GAMS optimization program in one hour limit. It had been explained that CPLEX solver has been used as solving the problem with GAMS program. While the CPLEX algorithm solves MIP problems, it uses an algorithm which includes branch and bound method. In this algorithm, a best node is found in each step and the optimum solution is sought over this best node. This best node is an infeasible solution which is found by a method peculiar to CPLEX and its value is smaller than the optimum. The gap between the optimum solution and this best node diminishes as the result reaches to the optimum. That is to say, as the algorithm gets closer to the optimum solution, the gap between the best node and the objective function goes to 0. The optimum solution is achieved as the gap falls under a particular limit. Therefore, the best node GAMS find along with CPLEX solver after one hour period is lower bound value. Because, as the solution approaches to the optimum this best node get closer to the optimum and its value increases. By this way, a definite lower bound value of one hour is found for some of the prepared sample problems. However, as this node value cannot be found in some problems; that is to say, as the

one hour period is not sufficient in order to find a start node, or the program can iterate very lowly, the number of the usable lower bound value to be found by this method is very few.

For the entire attempts to be prepared with the experimental design, the set of lower bounds (LBS_l) is formed by making use of these six different lower bounds for each sample ($l = 1, 2, ..., 360$).

$$LBS_l = \{LB_{l1}, LB_{l2}, LB_{l3}, LB_{l4}, LB_{l51}\ LB_{l52}\ LB_{l53}\ LB_{l54}\ LB_{l55}\ LB_{l56}, LB_{l6}\}$$

The largest lower bound ($LB_l = \max(LBS_l)$) out of the LBS_l set belonging to each sample is chosen. The largest one of the lower bound values in the LBS_l set is called the infimum of the set. Sundaram (1999) has defined infumum concept as such: "The infimum is defined to be the greatest lower bound of the set of lower bounds of problem." (p. 14). By way of this lower bound, the necessary data has been achieved in order to assess the performances of the heuristic solutions. However, there is another key point which is the relationship between lower bound and the optimum solution. This should be taken into account during the performance assessment.

4.6 Computational Results

In this section, lower bounds, upper bounds and optimum solutions will be evaluated. The computational experiences solved with optimization program whereas lower and upper bound solutions are figured out with algorithms. The results are evaluated in order to assess the performance of the solution.

An optimum solution for the problem is tried to be found by using the optimization program yet enough optimum solution could not be found because of one hour time limit. For the trips whose the number of trips is more than sixty, optimization program could not even find a starting solution. Because of these reasons given above, the optimum solution values are tried to be found by choosing

five samples from ten samples tests belonging to each data set without the limitation of time. In the twenty trips attempts, five problems of four different data sets are solved at average two hours CPU time per sample. When trip number is increased to thirty, the solution of twenty problems of four different data sets has taken ten hours CPU time per sample at average, as well. Moreover, when the number of trip is raised to forty, the computer could not find the optimum solution of the problem even in sixty hours CPU time for only one sample of operating. Therefore, it can be deduced that the more number of trips, the longer solution period of the problem is and also the increase in the solution period is much more than the number of trips.

However, it has been observed that the optimum solution period of the problem does not change depending on vehicle number and vehicle type. Within the same data set, some attempts are conducted with single type vehicles less in number but the same problems are solved in the same period. This shows that any change in vehicle number does not influence the solution period of the problem.

The solution of the same problems acquired by C programs which conform to the heuristic solution algorithms takes at average of five seconds CPU time per sample. By this way, the heuristic methods enable to solve the problems in a very short time which normally takes quite long when they are solved with optimization program. It should be noted that the relation of result with the optimum is as important as the shortness of time. As it has been pointed earlier, a few kinds of lower bounds are improved since there are limited optimum solutions. The problems without an optimum solution are evaluated by lower bounds. The GAMS solutions show that the total of forty one attempts has optimum solutions. In the Appendix B there are tables demonstrating the results of the attempts with optimum values, the lower bounds and upper bounds of these results together with their comparisons.

The evaluation of the problems is shown in the tables numbered; 4.2, 4.3, 4.4, 4.5 and 4.6. Before analyzing these tables, the abbreviations are:

n – shows the number of the trip and it changes as n = 20, 30, 40, …, 100. p – shows processing time distribution types and there are two different distribution composing *p. r* - shows ready time distribution types and this is composed by use of two different distributions as well. The structure of processing time and ready time distributions are explained in section 4.2.

The average and maximum gap between the minimum upper bounds (supremum) which are acquired by the six separate upper bounds of each attempt and the optimum solutions are calculated as follows:

$$\text{Gap between opt. and min. UB} = \frac{\sum_{l=1}^{L}\left(\frac{\min(UB_l) - opt_l}{opt_l}\right)}{L}$$

Similarly, the gaps between the maximum lower bounds and the optimum solution are calculated as follows:

$$\text{Gap between opt. and max. LB} = \frac{\sum_{l=1}^{L}\left(\frac{(opt_l - \max(LB_l)}{opt_l}\right)}{L}$$

The gaps between max. LB and min. UB are calculated as follows:

$$\text{Gap between min. UB and max. LB} = \frac{\sum_{l=1}^{L}\left(\frac{(\min(UB_l) - \max(LB_l)}{\max(LB_l)}\right)}{L}$$

However, while forming the set (LBS_l), the lower bounds found by the GAMS are not used here. Because GAMS gives us the optimum solution and the lower bound are very close to each other.

The performances of our infimum and supremum are investigated first in Table 4.2. Table 4.2 reports the average and maximum supremum and infimum deviations as a percentage of optimal solutions for r =1 and 2, p =1 and 2, n = 20 and 30 trips. The GAP among those three values is calculated. The average gap between optimal solutions and supremum are about 10 % while the GAP between optimal solutions and infimum is about 24 %.

Table 4.2 Comparing Optimum Solution, Infimum and Supremum

# of Trip	p	r	Optimum - Min UB (Supremum)		Optimum - Max. LB (Infimum)		Min. UB - Max. LB (Supremum - Infimum)	
			Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap
20	1	1	0,107	0,220	0,218	0,305	0,290	0,370
20	1	2	0,142	0,178	0,241	0,299	0,334	0,405
20	2	1	0,076	0,225	0,243	0,273	0,294	0,354
20	2	2	0,109	0,157	0,244	0,331	0,316	0,415
30	1	1	0,139	0,191	0,229	0,287	0,323	0,379
30	1	2	0,161	0,216	0,201	0,316	0,312	0,405
30	2	1	0,096	0,151	0,246	0,344	0,312	0,413
30	2	2	0,076	0,128	0,254	0,353	0,308	0,376

Table 4.3 reports the average and maximum upper bound deviations as a percentage of optimal solutions for r =1 and 2, p =1 and 2, n = 20 and 30 trips . In this table, the average and maximum gap between the optimum solution and every heuristic solution is calculated separately. The mean of average gap is 10.8 % for n =20 and 11.8 % for n =30, while mean of maximum gap is 19.5 % for n =20 and 17.1 % for n =30. TBHO gives the best results 19.3 % for n =20 and VBH gives the best result 19.1% for n =30 but the worst result for n =20 is 26.7 % from GBH and the worst result for n =30 is 21.7 % from GBHO. The means of total solution from each method do not differentiate much. That is to say, this method yields better results is not an exactly true claim. Because of that, a set of upper bounds is created for every problem and the minimum one is considered to be the best upper bound. In addition to this, in three of the attempts optimum result is found by heuristic methods as it has been displayed in Appendix B.

Furthermore, in Table 4.4, the values found by the distinct lower bounds and the optimum solutions for r =1 and 2, p =1 and 2, n = 20 and 30 trips are compared. As it may be inferred from the table, the greatest lower bound value is found by LB_5 method. The lower bound values found by GAMS are not used here, too.

The comparisons of all upper bounds and the best lower bound for r =1 and 2, p =1 and 2, n = 20, 30, …, 100 trips are in Table 4.5. The best average gap between the upper and lower bound is achieved by TBHO method as 43 %. The average gap of the other methods are as follows; TBH 53 %, VBH 53 %, VBHO 50 %, GBH and GBHO 55 %. When the heuristic methods are evaluated severally, it may be stated that the best result is reached by TBHO method. Yet, since the least one of these 6 different methods will be accepted as the solution, the infimum and the supremum are compared in Table 4.5, as well. The result of this comparison shows that the gap between lower bound and upper bound is 35 % at average. At all attempts, the minimum average gap between the infimum and the supremum is 19 %, while maximum gap is 73 %. The means of average gap for each n are not different after n = 30, they are about 36 %.

Lastly, the effects of p and r on the solution performance report in Table 4.6. The gap for p =2, r =1 has closer gap 33 % than the other combination and the gaps for other combination are not different. The heuristics have best performance in the combination p =2, r =1 than the other combinations. Because the mean of p = 2 has closer gap between heuristics and maximum lower bound than the means of p = 1, it is about 35 % and the mean of r = 1 has closer gap than the mean of r = 2 too, it is about 38 %.

The 35% difference of the average value in fact does not give effective result. However, there has been a percentage of 24% between the outcomes acquired by the improved lower bound solution methods and the optimum outcomes showing the difference, this reveals that the percentage between the optimum and the best upper bound is 9%. This may be schematized as:

$$\frac{\overline{UB}-\overline{LB}}{\overline{LB}} \cong 0{,}35 \; and \; \frac{\overline{opt.}-\overline{LB}}{\overline{LB}} \cong 0{,}24 \qquad \Rightarrow \frac{\overline{UB}-\overline{opt.}}{\overline{opt.}} \cong 0{,}09$$

As displayed in Table 4.2, the average percentage between optimum and the least upper bound of 20 and 30 trips is 10%, actually. The percentage presented above is close to this one, as well. At this point, it may be concluded that the difference between the optimum solution and the least upper bound is around 10%.

Table 4.3 Optimum Solution and Upper Bounds

# of Trip	p	r	Upper Bound Solutions (Heuristics)													
			TBH (UB1)		TBHO (UB2)		VBH (UB3)		VBHO (UB4)		GBH (UB5)		GBHO (UB6)		Min UB	
			Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap
20	1	1	0,231	0,387	0,227	0,331	0,221	0,387	0,253	0,380	0,212	0,373	0,236	0,490	0,107	0,220
		2	0,176	0,258	0,208	0,285	0,172	0,284	0,207	0,327	0,283	0,391	0,384	0,617	0,142	0,178
	2	1	0,310	0,489	0,137	0,242	0,299	0,458	0,137	0,261	0,285	0,427	0,155	0,338	0,076	0,225
		2	0,295	0,399	0,199	0,268	0,264	0,371	0,197	0,299	0,288	0,311	0,183	0,304	0,109	0,157
	av.		**0,253**	**0,384**	**0,193**	**0,282**	**0,239**	**0,375**	**0,199**	**0,317**	**0,267**	**0,376**	**0,239**	**0,437**	**0,108**	**0,195**
30	1	1	0,216	0,325	0,201	0,329	0,189	0,283	0,281	0,491	0,335	0,630	0,321	0,430	0,139	0,191
		2	0,251	0,391	0,304	0,494	0,212	0,342	0,346	0,409	0,202	0,304	0,291	0,402	0,161	0,216
	2	1	0,217	0,359	0,137	0,177	0,221	0,359	0,171	0,263	0,132	0,298	0,122	0,183	0,096	0,151
		2	0,158	0,301	0,133	0,207	0,143	0,256	0,174	0,314	0,123	0,262	0,133	0,446	0,076	0,128
	av.		**0,211**	**0,344**	**0,194**	**0,302**	**0,191**	**0,310**	**0,243**	**0,369**	**0,198**	**0,374**	**0,217**	**0,365**	**0,118**	**0,171**
Overall Av.			**0,232**	**0,364**	**0,193**	**0,292**	**0,215**	**0,343**	**0,221**	**0,343**	**0,232**	**0,375**	**0,228**	**0,401**	**0,113**	**0,183**

Table 4.4 Optimum Solution and Lower Bounds

# of Trip	p	r	Lower Bounds											
			LB1 - Opt.		LB2 - Opt.		LB3 - Opt.		LB4 - Opt.		LB5 - Opt.		Max. LB - Opt.	
			Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap
20	1	1	0,656	0,749	0,640	0,714	0,492	0,570	0,450	0,537	0,218	0,305	0,218	0,305
		2	0,725	0,747	0,559	0,598	0,465	0,496	0,417	0,452	0,241	0,299	0,240	0,299
	2	1	0,755	0,827	0,701	0,788	0,600	0,721	0,566	0,698	0,243	0,273	0,243	0,273
		2	0,743	0,827	0,657	0,737	0,575	0,691	0,538	0,664	0,244	0,331	0,236	0,331
	av.		**0,720**	**0,787**	**0,639**	**0,709**	**0,533**	**0,619**	**0,493**	**0,587**	**0,237**	**0,302**	**0,234**	**0,302**
30	1	1	0,670	0,727	0,583	0,676	0,463	0,580	0,418	0,545	0,229	0,287	0,234	0,287
		2	0,596	0,619	0,502	0,566	0,341	0,414	0,285	0,365	0,210	0,316	0,199	0,316
	2	1	0,720	0,777	0,672	0,720	0,566	0,636	0,529	0,604	0,246	0,344	0,252	0,344
		2	0,715	0,783	0,647	0,686	0,543	0,594	0,504	0,559	0,254	0,353	0,261	0,353
	av.		**0,675**	**0,726**	**0,601**	**0,662**	**0,478**	**0,556**	**0,434**	**0,518**	**0,235**	**0,325**	**0,237**	**0,325**
Overall Av.			**0,697**	**0,757**	**0,620**	**0,686**	**0,506**	**0,588**	**0,463**	**0,553**	**0,236**	**0,314**	**0,235**	**0,314**

Table 4.5 Upper Bounds and Lower Bounds

# of Trip	p	r	Average of Heuristic's Results													
			TBH		TBHO		VBH		VBHO		GBH		GBHO		Supremum	
			Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap
20	1	1	0,339	0,603	0,367	0,811	0,347	0,798	0,434	1,108	0,450	1,081	0,437	0,959	**0,265**	**0,603**
		2	0,342	0,661	0,365	0,593	0,344	0,686	0,403	1,062	0,469	0,786	0,519	0,820	**0,294**	**0,593**
	2	1	0,510	0,779	0,298	0,749	0,484	0,746	0,275	0,559	0,408	0,696	0,252	0,603	**0,191**	**0,559**
		2	0,422	0,710	0,264	0,478	0,384	0,637	0,315	0,709	0,420	0,694	0,359	0,924	**0,214**	**0,478**
	average		**0,403**	**0,688**	**0,323**	**0,658**	**0,390**	**0,717**	**0,357**	**0,859**	**0,437**	**0,814**	**0,392**	**0,826**	**0,241**	**0,558**
30	1	1	0,404	0,787	0,384	0,671	0,374	0,733	0,485	0,867	0,525	0,973	0,567	1,095	**0,330**	**0,671**
		2	0,382	0,607	0,418	0,794	0,353	0,589	0,464	0,870	0,414	0,702	0,536	0,862	**0,298**	**0,501**
	2	1	0,478	0,763	0,262	0,447	0,485	0,763	0,301	0,474	0,362	0,730	0,240	0,497	**0,213**	**0,447**
		2	0,315	0,616	0,219	0,387	0,303	0,616	0,256	0,498	0,273	0,642	0,216	0,438	**0,178**	**0,387**
	average		**0,395**	**0,693**	**0,321**	**0,575**	**0,379**	**0,675**	**0,377**	**0,677**	**0,393**	**0,762**	**0,390**	**0,723**	**0,255**	**0,501**
40	1	1	0,436	0,591	0,468	0,689	0,454	0,640	0,565	0,740	0,628	0,825	0,729	0,957	**0,408**	**0,550**
		2	0,345	0,536	0,468	0,620	0,357	0,604	0,576	0,729	0,502	0,641	0,730	1,022	**0,321**	**0,493**
	2	1	0,587	0,739	0,349	0,488	0,590	0,734	0,343	0,473	0,502	0,687	0,334	0,598	**0,297**	**0,458**
		2	0,504	0,775	0,339	0,627	0,507	0,811	0,355	0,604	0,482	0,796	0,354	0,761	**0,292**	**0,604**
	average		**0,468**	**0,660**	**0,406**	**0,606**	**0,477**	**0,697**	**0,460**	**0,636**	**0,529**	**0,738**	**0,536**	**0,835**	**0,329**	**0,526**
50	1	1	0,460	0,595	0,528	0,598	0,456	0,589	0,626	0,740	0,653	0,870	0,849	1,282	**0,435**	**0,566**
		2	0,459	0,749	0,645	0,886	0,455	0,729	0,720	0,920	0,661	0,838	0,803	1,107	**0,422**	**0,729**
	2	1	0,639	0,756	0,333	0,490	0,636	0,778	0,398	0,669	0,567	0,671	0,374	0,578	**0,328**	**0,490**
		2	0,640	0,715	0,426	0,597	0,629	0,715	0,477	0,762	0,603	0,728	0,403	0,711	**0,366**	**0,539**
	average		**0,550**	**0,704**	**0,483**	**0,642**	**0,544**	**0,703**	**0,555**	**0,772**	**0,621**	**0,777**	**0,607**	**0,919**	**0,388**	**0,581**
60	1	1	0,479	0,700	0,550	0,698	0,468	0,657	0,667	0,852	0,594	0,862	0,817	1,045	**0,435**	**0,611**
		2	0,488	0,637	0,626	0,850	0,482	0,564	0,720	0,946	0,660	0,791	0,923	1,039	**0,459**	**0,564**
	2	1	0,662	0,804	0,341	0,539	0,653	0,798	0,386	0,563	0,573	0,704	0,350	0,537	**0,295**	**0,401**
		2	0,635	0,718	0,425	0,619	0,636	0,750	0,452	0,649	0,530	0,679	0,446	0,893	**0,374**	**0,566**
	average		**0,566**	**0,715**	**0,485**	**0,677**	**0,560**	**0,692**	**0,556**	**0,752**	**0,589**	**0,759**	**0,634**	**0,879**	**0,391**	**0,535**
70	1	1	0,460	0,673	0,549	0,815	0,486	0,740	0,720	0,932	0,613	0,815	0,935	1,198	**0,427**	**0,608**
		2	0,402	0,598	0,531	0,655	0,409	0,547	0,658	0,810	0,591	0,838	0,822	0,989	**0,382**	**0,547**
	2	1	0,629	0,715	0,360	0,480	0,617	0,731	0,404	0,516	0,463	0,549	0,367	0,518	**0,322**	**0,420**
		2	0,647	0,765	0,461	0,678	0,643	0,756	0,504	0,746	0,561	0,734	0,503	1,110	**0,414**	**0,670**
	average		**0,535**	**0,688**	**0,475**	**0,657**	**0,539**	**0,694**	**0,572**	**0,751**	**0,557**	**0,734**	**0,657**	**0,954**	**0,386**	**0,561**
80	1	1	0,470	0,607	0,553	0,770	0,485	0,605	0,703	0,870	0,697	0,974	0,860	1,115	**0,439**	**0,575**
		2	0,426	0,529	0,568	0,670	0,441	0,526	0,659	0,808	0,592	0,839	0,799	1,051	**0,401**	**0,482**
	2	1	0,654	0,739	0,401	0,543	0,640	0,733	0,469	0,678	0,516	0,649	0,372	0,614	**0,339**	**0,481**
		2	0,679	0,790	0,405	0,509	0,676	0,787	0,437	0,627	0,592	0,719	0,461	0,622	**0,386**	**0,509**
	average		**0,557**	**0,666**	**0,481**	**0,623**	**0,560**	**0,663**	**0,567**	**0,746**	**0,599**	**0,795**	**0,623**	**0,850**	**0,391**	**0,512**
90	1	1	0,418	0,612	0,540	0,772	0,414	0,667	0,637	0,821	0,670	0,919	0,934	1,333	**0,388**	**0,552**
		2	0,354	0,514	0,510	0,785	0,361	0,500	0,596	0,769	0,540	0,707	0,793	1,032	**0,347**	**0,500**
	2	1	0,650	0,724	0,328	0,396	0,649	0,733	0,361	0,513	0,557	0,671	0,375	0,429	**0,306**	**0,350**
		2	0,667	0,752	0,481	0,718	0,661	0,746	0,535	0,719	0,509	0,656	0,540	0,729	**0,419**	**0,535**
	average		**0,522**	**0,651**	**0,465**	**0,668**	**0,521**	**0,662**	**0,532**	**0,705**	**0,569**	**0,738**	**0,661**	**0,881**	**0,365**	**0,484**
100	1	1	0,404	0,804	0,498	0,886	0,393	0,811	0,634	0,946	0,583	0,974	0,797	1,333	**0,378**	**0,729**
		2	0,399	0,498	0,518	0,601	0,398	0,545	0,628	0,791	0,608	0,794	0,812	1,011	**0,379**	**0,486**
	2	1	0,682	0,743	0,418	0,597	0,661	0,757	0,480	0,698	0,564	0,670	0,402	0,625	**0,343**	**0,440**
		2	0,683	0,781	0,392	0,502	0,661	0,707	0,445	0,634	0,565	0,676	0,488	0,695	**0,388**	**0,502**
	average		**0,542**	**0,706**	**0,456**	**0,646**	**0,528**	**0,705**	**0,547**	**0,767**	**0,580**	**0,778**	**0,625**	**0,916**	**0,372**	**0,539**
Overall Av.			**0,534**	**0,686**	**0,428**	**0,639**	**0,530**	**0,690**	**0,495**	**0,741**	**0,547**	**0,766**	**0,552**	**0,865**	**0,348**	**0,533**

Table 4.6 Effects of p and r on the Solution Performance

	p	r	Effects of p and r on the Solution Performance													
			TBH		TBHO		VBH		VBHO		GBH		GBHO		Supremum	
			Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap	Av. Gap	Max. Gap
Overall of All Problem	1	1	0,484	0,746	0,555	0,839	0,485	0,780	0,684	0,984	0,677	1,037	0,866	1,290	**0,438**	**0,683**
		2	0,450	0,666	0,581	0,807	0,450	0,661	0,678	0,963	0,630	0,867	0,842	1,117	**0,413**	**0,612**
	2	1	0,686	0,845	0,386	0,591	0,677	0,847	0,427	0,643	0,564	0,753	0,383	0,625	**0,329**	**0,506**
		2	0,649	0,828	0,426	0,639	0,637	0,816	0,472	0,743	0,567	0,791	0,471	0,860	**0,379**	**0,599**
	average p=1		**0,467**	**0,706**	**0,568**	**0,823**	**0,467**	**0,721**	**0,681**	**0,974**	**0,653**	**0,952**	**0,854**	**1,203**	**0,426**	**0,647**
	average p=2		**0,668**	**0,837**	**0,406**	**0,615**	**0,657**	**0,831**	**0,450**	**0,693**	**0,565**	**0,772**	**0,427**	**0,743**	**0,354**	**0,552**
	average r=1		**0,585**	**0,796**	**0,470**	**0,715**	**0,581**	**0,813**	**0,556**	**0,814**	**0,620**	**0,895**	**0,625**	**0,957**	**0,384**	**0,594**
	average r=2		**0,549**	**0,747**	**0,504**	**0,723**	**0,544**	**0,738**	**0,575**	**0,853**	**0,598**	**0,829**	**0,657**	**0,989**	**0,396**	**0,605**

CHAPTER FIVE
CONCLUSION

5.1 Summary and Conclusion

This study attempts to develop methods for vehicle scheduling in a large organization which will provide for the transportation of the participants between the activity centers conforming to the conditions of the organization with the minimum cost. In this study, a mathematical model is developed for a real life application and general solution methods for similar problems are proposed.

The problem is a vehicle scheduling problem with sequence dependent trips. The trips in the problem are given within a particular time interval. Each trip should be proposed within this time interval. There are an enough number of vehicles in each size and cost to perform the trip. The objective function is to form the minimum cost scheduling to assign all the trips to vehicles. The problem is a tactical fixed job scheduling problem in this respect. However, unlike the problems in the literature, the vehicles here are not identical and they have fixed and variable usage times and costs. The purpose in this problem is to minimize these usage costs and make a schedule which will provide for the implementation of the entire tasks.

The definitions of the variables and the problem structure are explained in chapter 3, where the variables dependent on those parameters and the available data about the system are stated. Furthermore, as the problem includes real life data, particular assumptions are given as well.

The problem is modeled mathematically by using mixed integer programming, conforming to the assumptions. The mathematical model is attempted to solve with optimization programs by preparing samples consisting with the actual problem. However, by making use of optimization programs, the optimum result could be

achieved only in the problems with small number of trips. This demonstrates that as the number of variables increases, the solution time takes longer as well. What makes the problem complex is the number of trips. If the number of trips increases, the solution time will increase more than the number of trips increasing ratio. Actually, during the solution of the problem, the optimum is achieved at an average of two hours CPU time in attempts of twenty trips, while the solution time increases to an average of eight hours CPU time in attempts of thirty trips, and finally, in attempts of fourty trips the optimum result could not be found even after fifty - sixty hours CPU time of solution time. This shows the complexity of the problem.

Heuristic approaches have been improved as the longer solution time of the problem. Three different heuristic solution algorithms, namely trip based, vehicle based and group based, have been constructed. Each algorithm has two different solution approaches as with and without overtime. The algorithms have been modeled by making use of Dev C++ compiler in C programming language and an upper bound for the problem have been obtained by solving these models. It has been observed that even the solution time for the problems with a hundred trips have taken one - two seconds CPU time.

As well as the problem can be solved by using heuristic methods very quickly, the performance of the solution should be determined with the optimum. Therefore, a certain experiment design has been done and problems with various numbers of trips and different data sets have been created. The created problems have been sought to be solved through heuristic algorithms. In order to determine the performance of the heuristic method, the samples have been attempted to solve in a definite time limit in GAMS optimization program by using CPLEX solver. However, sufficient number of optimum solution could not be achieved. Also, by making use of Gantt chart, the trip orders assigned to the vehicles of the schedules which have been created with optimum solution and heuristic methods for a sample with thirty trips have been presented in Appendix C. The assignments created for the problem can be seen in the gantt charts in Appendix C.

As there is not sufficient optimum solution for the problem, lower bounds have been found in order to assess the upper bounds. The difference between the newly found lower bounds and the upper bounds is too much, therefore the optimum solution has been attempted to be found by using GAMS program without any time limit for the problems. In the problems with twenty and thirty trips, sufficient optimum solutions could be achieved, however in the attempts with forty trips, no optimum solution could be achieved in even fifty – sixty hours CPU time of solution periods. By this way, the performances of the heuristic approaches have been attempted to be assessed by comparing the *UB, LB* and optimum solution values for the attempts with twenty and thirty trips. For the entire samples, *UB* and *LB* values have been compared. The smallest result out of the six results found by heuristic approaches for each sample, namely the least value of the upper bounds (supremum) is 10 % more than the optimum solution of the problem at an average.

If this problem is solved through the proposed heuristic methods, we can find an average of 10% an extra cost than the cost of optimal solution but heuristics could be achieved the schedule in the short run. Also, by modifying the solution methods in accordance with the organization conditions, these solution methods could be used in the scheduling of the passenger transportations in large-scale organizations. Consequently, the vehicle scheduling problems with these sequence dependent trips could be solved at a very short period by using these methods or by the methods which will be based on these methods in accordance with the problem structure.

5.2 Future Research

Fixed job scheduling problems of this type are prone to improvement and study. First of all, such scheduling problems are not common in the literature. It is also a problem having many fields of application. Especially, the interval scheduling and vehicle scheduling problems are very commonly used in real life. Solution methods for these problems can be improved.

The fixed job scheduling problem studied in this thesis can be improved by changing its assumptions. Here, we have assumed the processing times, demands and ready times as deterministic data. In real life, the demands can be deterministic; however, the ready times and processing times are always stochastic. The problem can be remodeled as stochastic interval scheduling or stochastic vehicle scheduling problem thus solution methods could be improved.

Furthermore, new solutions which can produce closer values to the optimum can be generated by making improvements in the heuristic solution methods existing in this study. New heuristic approaches producing better results could also be created. Apart from the heuristic solution methods, branch and bound solution algorithm for the problem could be improved by proving the problem to be NP – hard or NP – Complex.

Moreover, it has been stated in the previous parts that the problems had been solved through the specially prepared heuristic solution methods which are coded in C program. Making a package software can provide for a convenient solution of the problems by the end users which can be produced from the methods developed for the problem or out of the algorithms prepared for the solution methods of another vehicle scheduling problem in real life following these methods. By creating a user interface to the C programs, a package program might be developed having the capability to find the best result which uses the heuristic algorithms for the problems in real life.

Interval scheduling, tactical fixed job scheduling problem and vehicle scheduling problems have many ranges of application. They are commonly encountered in real life and are prone to continuous development. In this study, an actual scheduling problem having the same attributes with this problem have been attempted to be solved in a reasonable period of time by developing heuristic approaches which will operate to find a result close to the optimum. However, as it has been stated, the subject is open to development.

REFERENCES

Arkin, E. A. & Silverberg, E. B. (1987). Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics*, 18, 1-8.

Baita F., Pesenti R., Ukovich W. & Favaretto D., (2000). A comparison of different solution approaches to the vehicle scheduling problem in a practical case. *Computers & Science Research,* 27, 1249 - 1269.

Bertossi A.A., Carraresi P.& Gallo G. (1987). On same matching problems arising in vehicle scheduling models. *Networks* 17, 271 - 281.

Bouzina, K. I. & Emmons, H. (1996). Interval scheduling on identical machines. *Journal of Global Optimization*, 9, 379 – 393.

Brooke A., Kendrick D., Meeraus A. & Raman R. (1998). *GAMS A User's Guide*. United States of America, GAMS Development Corporation.

Carter, M. W. & Tovey, C.A. (1992). When is the classroom assignment problem hard? *European Journal of Operational Research*, 40, 28 – 39.

Desaulniers G., Lavigne J. & Soumis F. (1998). Multi - depot vehicle scheduling problems with time windows and waiting cost. *European Journal of Operational Research,* 111, 479 - 494.

Eliiyi D. T. (2004). Operational Fixed Job Scheduling Problem. PHD Thesis, Department of Industrial Engineering, Middle East Technical University.

Fischetti, M., Martello, S. & Toth, P. (1987). The fixed job schedule problem with spread – time constraints. *Operation Research*, 35(6), 849 – 858.

Fischetti, M., Martello, S. & Toth, P. (1989). The fixed job schedule problem with working – time constraints. *Operation Research*, 37, 395 – 403.

Fischetti, M., Martello, S. & Toth, P. (1992). Approximation algorithm for job schedule problems. *Operation Research*, 40(1), 96 – 108.

Freling R, Paixao J.M.P. & Wagelmans A.P.M. (1999). Models and algorithms for single depot scheduling problem.

Hagnagi A., Banihashemi M. & Chiang K.H, (2003). A comparative analysis of bus transit vehicle scheduling models. *Transportation Research Part B*, 37, 301 - 322.

Hashimoto A. & Stevens J.E., (1971). Wire Routing by Optimizing Channel Assignments within Large Apertures. *Proceedings of the 8th Design Automation Workshop*, 155-169.

Karakütük S.S. & Karacizmeli H.İ. (2004). Example Vehicle Scheduling Procedure for UNIVERSIADE 2005. BS Thesis, Department of Industrial Engineering, Dokuz Eylul University – Izmir.

Kolen, A. W. J. & Kroon, J. G. (1993). On the computation complexity of (maximum) shift class scheduling. *European Journal of Operational Research*, 64; 138 – 151.

Kroon, L. G. (1990). Job scheduling and capacity planning in aircraft maintenance. PHD Thesis, Rotterdam School of Management, Erasmus University, The Netherlands.

Kroon, L., Solomon, M. & Wassenhove, L. N. V. (1995). Exact and approximation algorithms for the operational fixed interval scheduling problem. *European Journal of Operational Research*, 82, 190 – 205.

LINDO Systems Inc. (1999). *Optimization Modeling with LINGO.* (3rd ed.). United States of America, LINDO Systems Inc.

Löbel A. (1997). Optimal Vehicle Scheduling in Public Transit. PhD Thesis, Technician University of Berlin.

Morton T.E. & Pentico D.W. *Heuristics Scheduling Systems*. Canada: John Wiley & Sons. Inc..

Nahmias, S. (1992). *Production and Operation Analysis. (2nd ed.).* Canada: IRWIN.

Pinedo, M., & Chao, X. (1999). *Operations Scheduling with Applications in Manufacturing and Services.* Singapore: McGraw-Hill.

Rojenasoonthon S. & Bard J. (2005). A GRASP for parallel machine scheduling with time windows. *INFORMS Journal of Computing,* 17 (1), 32 – 51.

Schrage, L. (1981). Formulation and structure of more complex/realistic routing and scheduling problems. *Networks,* 11, 229 - 232.

Seiden, S. S. (1998). Randomized online interval scheduling. *Operation Research Letters*, 22, 171 – 177.

Shen, Z. X. & Jong, C. C. (1999). A lower bound on general minimal resources interval scheduling with arbitrary component selection. *IEEE Transaction*, 11, 378 – 381.

Solomon M.M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraint. *Operations Research,* 35 (2), 254 – 265.

Stevenson, W. J. (1999). *Productions Operations Management (6th ed.)*. USA: McGraw-Hill Companies, Inc..

Sundaram, K.R. (1999). *A First Course in Optimization Theory. (1st. ed.)*. United States of America, Cambridge University Press

Wagner, S. (2001). Restricted cache scheduling. PhD Thesis, Department of Computer Science and Engineering, Michigan State University.

Williams P.H. (2003). *Model Building in Mathematical Programming (4th ed.)*. England, John Wiley & Son Inc..

Winston W.L. (2004). *Operation Research Application and Algorithms (4th ed.)*. Canada, Thomson.

Woeginger, G. J. (1994). On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130, 5 – 16.

Yuan, J. & Lin, Y. (2005). Single machine preemptive scheduling with fixed jobs to minimize tardiness related criteria. *European Journal of Operational Research*, 164, 851 – 855.

Appendix A.1: Data sets of the example problem with 30 trips.

Table A.1.1 Trip Data

Trip Number (*j*)	**Demand** (D_j)	**Ready Time** (r_j)	**Deadline** (d_j)	**Processing Time** ($p_j = d_j - r_j$)	**Team Trip** (T)
1	66	1	6	5	0
2	42	2	7	5	0
3	17	4	10	6	0
4	65	6	12	6	0
5	9	7	33	26	1
6	22	23	42	19	1
7	84	24	33	9	0
8	60	35	41	6	0
9	36	55	61	6	0
10	30	56	66	10	0
11	13	78	95	17	1
12	66	82	87	5	0
13	18	86	102	16	1
14	58	92	97	5	0
15	14	92	99	7	1
16	81	95	105	10	0
17	8	103	113	10	0
18	58	123	133	10	0
19	5	129	156	27	1
20	9	130	139	9	0
21	96	155	162	7	0
22	24	158	168	10	1
23	11	159	171	12	1
24	12	164	175	11	1
25	35	174	179	5	0
26	20	175	214	39	1
27	74	177	185	8	0
28	30	179	186	7	1
29	9	193	199	6	0
30	20	199	228	29	1

Table A.1.2 Arc Matrix

FROM \ TO	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	1	8	7	9	5	1	4	7	7	4	3	3	3	5	3	2	2	1	3	8	8	2	2	5	6	2	5	7	1
2	0	0	3	8	5	4	7	5	4	7	1	1	7	8	6	7	8	8	7	6	6	3	3	5	1	8	9	2	5	9
3	0	0	0	4	5	4	7	8	2	8	8	9	5	3	6	9	8	9	2	1	1	9	4	6	8	6	2	9	8	9
4	0	0	0	0	3	8	6	4	5	9	4	6	1	6	7	6	1	1	3	4	5	6	3	9	1	9	4	7	8	1
5	0	0	0	0	0	8	4	5	9	9	5	9	2	9	2	7	4	1	5	7	5	9	3	2	6	4	7	1	2	5
6	0	0	0	0	0	0	2	1	3	9	2	6	9	9	8	8	7	8	5	2	2	7	4	5	5	4	9	1	4	1
7	0	0	0	0	0	0	0	2	9	6	8	4	6	6	1	7	4	1	6	7	4	1	2	1	6	4	6	8	1	6
8	0	0	0	0	0	0	0	0	9	2	4	2	1	4	4	2	2	5	8	6	7	2	9	7	5	2	6	9	4	6
9	0	0	0	0	0	0	0	0	0	4	3	6	3	2	1	4	2	2	2	8	7	4	3	5	7	1	9	5	8	9
10	0	0	0	0	0	0	0	0	0	0	3	9	6	5	1	9	1	8	9	8	1	5	3	7	8	9	6	9	5	5
11	0	0	0	0	0	0	0	0	0	0	0	6	7	2	4	2	9	4	5	1	6	6	7	5	7	8	2	7	1	7
12	0	0	0	0	0	0	0	0	0	0	0	0	8	4	9	9	9	3	5	5	3	1	7	9	4	7	7	5	5	4
13	0	0	0	0	0	0	0	0	0	0	0	0	0	8	6	4	7	3	4	6	1	3	8	3	5	6	8	1	8	5
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	9	7	3	8	8	4	7	2	4	7	4	1	3	9	7
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	4	3	7	8	2	3	4	9	5	1	3	8	4	8
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	9	8	3	9	2	2	9	3	8	7	7	5	3
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	7	4	4	2	3	1	4	8	6	8	8	6
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	8	9	5	5	9	7	2	4	7	9	3
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	8	7	4	8	6	9	7	8	5
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	5	1	5	1	2	7	9	9	2
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	3	9	9	9	4	7	3	1
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	4	2	1	9	8	2	5
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	3	3	8	3	9	7
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	9	2	3	7	6
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	6	7	3
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	7	9	6
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	1	6
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	7
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table A.1.3 Vehicle Data

Vehicle Types	c_i **(person)**	S **(time unit)**	O **(time unit)**	f_i **(YTL)**	v_i **(YTL)**
I	**45**	**100**	**50**	**1200**	**20**
II	**27**	**100**	**50**	**900**	**15**
III	**16**	**100**	**50**	**720**	**12**
IV	**10**	**100**	**50**	**600**	**10**

Table A.1.4 Incompatibility Matrix

Q Matris / Trips	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	2	3	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	21	22	23	24	25	26	27	28	29	30
2	0	0	3	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	21	22	23	24	25	26	27	28	29	30
3	0	0	0	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	21	22	23	24	25	26	27	28	29	30
4	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	22	23	24	25	26	27	28	29	30
5	0	0	0	0	0	6	7	8	0	0	0	0	0	0	0	0	0	0	0	0	21	22	23	24	25	26	27	28	29	30
6	0	0	0	0	0	0	7	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	25	26	27	28	29	30
7	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	25	26	27	28	29	30
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	28	29	30
9	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	30
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	30
11	0	0	0	0	0	0	0	0	0	0	0	12	13	14	15	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	15	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	14	15	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	20	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	21	22	23	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	23	24	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	24	0	0	27	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	25	0	27	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	26	27	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	27	28	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	28	29	30
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Appendix A.2: GAMS 20.2 Model of the Example Problem

```
Sets
i0 vehicles index /1*4/
j0 trips index /1*30/
b0 trips index /1*30/
ALIAS (j0,j0p) ;
 Parameters
fcost0(*)/ 1 1200
2 900
3 720
4 600
/
vcost0(*)/ 1 20
2 15
3 12
4 10
/
capacity0(*)/ 1 45
2 27
3 16
4 10
/
S0 daily avaliable standart time / 100 /
O0 daily avaliable overtime / 50 /
M0 big number / 30 /
N0 big number / 250 /;
```

```
r0(*)/1 1
2 2
3 4
4 6
5 7
6 23
7 24
8 35
9 55
10 56
11 78
12 82
13 86
14 92
15 92
16 95
17 103
18 123
19 129
20 130
21 155
22 158
23 159
24 164
25 174
26 175
27 177
28 179
29 193
30 199
/
```

```
p0(*)/1 5
2 5
3 6
4 6
5 26
6 19
7 9
8 6
9 6
10 10
11 17
12 5
13 16
14 5
15 7
16 10
17 10
18 10
19 27
20 9
21 7
22 10
23 12
24 11
25 5
26 39
27 8
28 7
29 6
30 29
/
```

```
d0(*)/1 66
2 42
3 17
4 65
5 9
6 22
7 84
8 60
9 36
10 30
11 13
12 66
13 18
14 58
15 14
16 81
17 8
18 58
19 5
20 9
21 96
22 24
23 11
24 12
25 35
26 20
27 74
28 30
29 9
30 20
/
```

```
ttt0(*)/
1 0
2 0
3 0
4 0
5 1
6 1
7 0
8 0
9 0
10 0
11 1
12 0
13 1
14 0
15 1
16 0
17 0
18 0
19 1
20 0
21 0
22 1
23 1
24 1
25 0
26 1
27 0
28 1
29 0
30 1
/
```

Table Q0(j0,b0) incompatibility matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	2	3	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	21	22	23	24	25	26	27	28	29	30
2	0	0	3	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	21	22	23	24	25	26	27	28	29	30
3	0	0	0	4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	19	0	21	22	23	24	25	26	27	28	29	30
4	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	21	22	23	24	25	26	27	28	29	30
5	0	0	0	0	0	6	7	8	0	0	0	0	0	0	0	0	0	0	0	0	21	22	23	24	25	26	27	28	29	30
6	0	0	0	0	0	0	7	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	25	26	27	28	29	30
7	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	25	26	27	28	29	30
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	28	29	30
9	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	30
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	0	0	0	30
11	0	0	0	0	0	0	0	0	0	0	0	12	13	14	15	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	13	0	15	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	14	15	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	17	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	20	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	21	22	23	0	0	0	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22	23	24	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	23	24	0	0	27	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	24	25	0	27	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	26	27	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	26	27	28	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	28	29	30
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	28	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	30
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

```
Variables
z10 total cost
y10(i0) use of vehicle
x10(i0,*) vehicle-trip
ov10(i0) overtime
Binary Variables y10,x10;
Positive Variables ov10;

Equations
cost0 objective function
demand0(j0) demand constraint
comp0(i0,j0,b0) overlapping trips
use0(i0) vehicle use
overtime0(i0,j0,b0) overtime usage
teamtrip0(j0) teamtrip;
cost0.. z10 =e= sum(i0,fcost0(i0)*y10(i0)+vcost0(i0)*ov10(i0)) ;
demand0(j0).. sum(i0,capacity0(i0)*x10(i0,j0)) =g= d0(j0) ;
comp0(i0,j0,b0)$(Q0(j0,b0) ne 0).. x10(i0,j0)+x10(i0,b0) =l= 1 ;
use0(i0).. sum(j0,x10(i0,j0)) =l= y10(i0)*M0 ;
overtime0(i0,j0,b0)$(Q0(j0,b0) = 0).. (r0(b0)+p0(b0))*x10(i0,b0)-r0(j0)*x10(i0,j0) =l= S0+ov10(i0)+(1-x10(i0,j0))*N0
teamtrip0(j0)$(ttt0(j0) ne 0).. sum(i0, x10(i0,j0)) =e= 1;

Model serhat0 /all/ ;
serhat0.reslim = 40000;
serhat0.iterlim = 10000000;

Option MIP = Cplex ;
solve serhat0 using MIP minimizing z10 ;
```

Appendix A.3: LINGO 8.0 Model of the Example Problem

```
OVERTIME(VEHICLE):O;
USE(VEHICLE):Y;
USING(VEHICLE,TRIP):X;
ARC(TRIP,TRIP):Q;
TEAM(TRIP,TRIP):T;

END SETS

DATA:
F=@ole('C:\lingo.xls','fixedcost');
V=@ole('C:\lingo.xls','variablecost');
C=@ole('C:\lingo.xls','capacity');
R=@ole('C:\lingo.xls','readytime');
P=@ole('C:\lingo.xls','processingtime');
D=@ole('C:\lingo.xls','demand');
Q=@ole('C:\lingo.xls','arctime');
T=@ole('C:\lingo.xls','undividedtrip');

END DATA

MIN = @SUM(VEHICLE(I):Y(I)*F(I)+ V(I)*O(I));

@FOR(TRIP(J):@SUM(VEHICLE(I):X(I,J))>=D(J));

@FOR(VEHICLE(I):@FOR(TRIP(J):@FOR(TRIP(B)|B#GT#J #AND# B#EQ#Q(J,B):X(I,J)+ X(I,B)<=1);));

@FOR(VEHICLE(I):@SUM(TRIP(J):X(I,J))<=Y(I)*30);

@FOR(VEHICLE(I):@FOR(TRIP(J):@FOR(TRIP(B)|B#GT#J #AND# B#NE#Q(J,B):X(I,B)*(R(B)+P(B))-R(J)*X(I,J)<=100+O(I)
250);));

@FOR(TRIP(J)| J#EQ#T(J,J):@SUM (VEHICLE(I):X(I,J))=1);

@FOR(VEHICLE(I):O(I)>=0);
@FOR(USING(I,J):@BIN(X(I,J)));
@FOR(VEHICLE(I):@BIN(Y(I)));
```

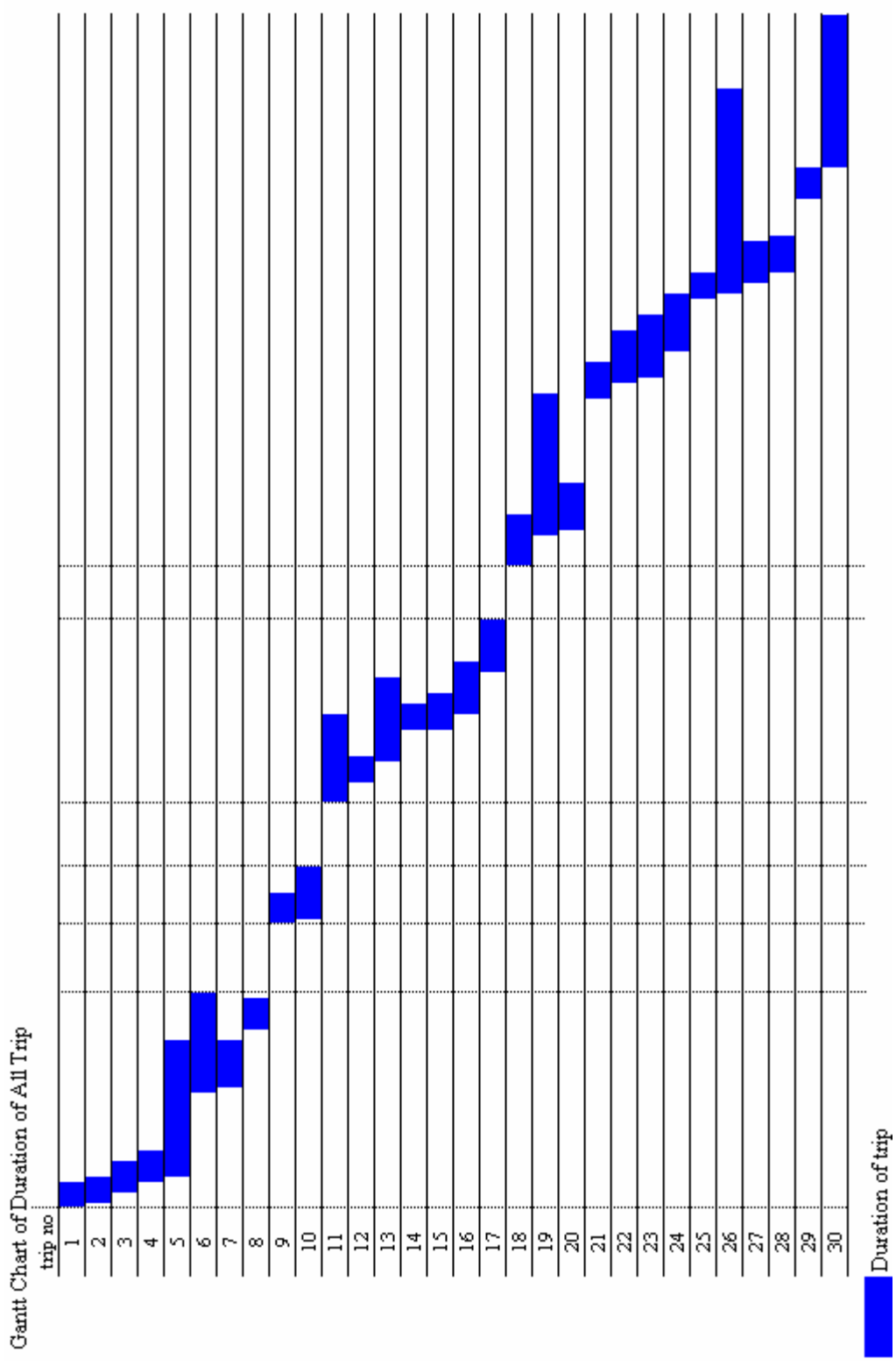
Appendix B.1: Comparison of All Optimum Solutions with Upper Bound Values

# of Trip	p	r	Opt. Solution	Upper Bound Solution												min UBV	Gap
				TBH (UB1)		TBHO (UB2)		VBH (UB3)		VBHO (UB4)		GBH (UB5)		GBHO (UB6)			
				Value	Gap	Value	Gap	Value	Gap	Value	Gap	Value	Gap	Value	Gap		
20	1	1	8705	11220	0,289	11076	0,272	10620	0,220	11260	0,294	11940	0,372	12970	0,490	10620	0,220
			9264	12180	0,315	10920	0,179	11460	0,237	11730	0,266	12720	0,373	11757	0,269	10920	0,179
			11820	12240	0,036	13660	0,156	13380	0,132	13280	0,124	12480	0,056	14043	0,188	12240	0,036
			11220	12660	0,128	13405	0,195	12660	0,128	13505	0,204	11220	0,000	12690	0,131	11220	0,000
			8780	12180	0,387	11686	0,331	12180	0,387	12116	0,380	11040	0,257	9675	0,102	9675	0,102
20	1	2	11540	14520	0,258	14457	0,253	14820	0,284	13817	0,197	14220	0,232	13595	0,178	13595	0,178
			12991	15120	0,164	14614	0,125	14520	0,118	14054	0,082	17340	0,335	21011	0,617	14054	0,082
			14240	15840	0,112	16738	0,175	15960	0,121	17358	0,219	16140	0,133	18525	0,301	15840	0,112
			13540	15780	0,165	17402	0,285	15720	0,161	17974	0,327	18840	0,391	21206	0,566	15720	0,161
			12836	15120	0,178	15452	0,204	15120	0,178	15516	0,209	16980	0,323	16155	0,259	15120	0,178
20	2	1	13535	20160	0,489	14238	0,052	19740	0,458	14099	0,042	19320	0,427	16155	0,194	14099	0,042
			13090	16920	0,293	13980	0,068	16620	0,270	14160	0,082	17520	0,338	15014	0,147	13980	0,068
			13651	18900	0,385	16725	0,225	18900	0,385	17220	0,261	19200	0,406	18268	0,338	16725	0,225
			8973	10380	0,157	11148	0,242	10380	0,157	10720	0,195	10860	0,210	8973	0,000	8973	0,000
			18155	22260	0,226	19898	0,096	22260	0,226	20090	0,107	18960	0,044	19890	0,096	18960	0,044
20	2	2	11715	14460	0,234	14205	0,213	13560	0,157	15213	0,299	14640	0,250	15278	0,304	13560	0,157
			17151	24000	0,399	19775	0,153	23520	0,371	18890	0,101	22440	0,308	19605	0,143	18890	0,101
			16977	22260	0,311	20693	0,219	22260	0,311	20360	0,199	22260	0,311	16977	0,000	16977	0,000
			9860	11760	0,193	12505	0,268	11280	0,144	11759	0,193	12540	0,272	12185	0,236	11280	0,144
			14390	19260	0,338	16405	0,140	19260	0,338	17203	0,195	18660	0,297	17695	0,230	16405	0,140
30	1	1	11742	13380	0,139	12652	0,077	13380	0,139	12819	0,092	19140	0,630	16786	0,430	12652	0,077
			14308	18960	0,325	17715	0,238	18360	0,283	19107	0,335	19380	0,354	16419	0,148	16419	0,148
			12675	14640	0,155	15640	0,234	14640	0,155	15826	0,249	15240	0,202	16619	0,311	14640	0,155
			12900	16680	0,293	17144	0,329	15480	0,200	19237	0,491	15360	0,191	18394	0,426	15360	0,191
			18740	21900	0,169	21084	0,125	21900	0,169	23219	0,239	24360	0,300	24182	0,290	21084	0,125
30	1	2	15560	18720	0,203	21106	0,356	17880	0,149	21920	0,409	17460	0,122	19596	0,259	17460	0,122
			12340	17160	0,391	18436	0,494	16560	0,342	16687	0,352	15000	0,216	16014	0,298	15000	0,216
			11390	14580	0,280	14159	0,243	13680	0,201	15154	0,330	13860	0,217	14997	0,317	13680	0,201
			14040	17760	0,265	16313	0,162	17580	0,252	17325	0,234	16140	0,150	16574	0,180	16140	0,150
			16152	18060	0,118	20430	0,265	18060	0,118	22670	0,404	21060	0,304	22643	0,402	18060	0,118
30	2	1	19596	26640	0,359	22590	0,153	26640	0,359	22550	0,151	25440	0,298	23187	0,183	22550	0,151
			19940	25200	0,264	22855	0,146	25200	0,264	22554	0,131	23100	0,158	22955	0,151	22554	0,131
			19130	19920	0,041	20130	0,052	19920	0,041	20685	0,081	20040	0,048	20945	0,095	19920	0,041
			16046	19980	0,245	18533	0,155	20400	0,271	20268	0,263	16680	0,040	16660	0,038	16660	0,038
			24710	29100	0,178	29081	0,177	28860	0,168	30301	0,226	27600	0,117	28235	0,143	27600	0,117
30	2	2	21125	22800	0,079	24230	0,147	22500	0,065	25763	0,220	23280	0,102	26514	0,255	22500	0,065
			27810	28920	0,040	28845	0,037	28920	0,040	29705	0,068	29520	0,061	29685	0,067	28845	0,037
			19024	22980	0,208	22953	0,207	23280	0,224	24994	0,314	20100	0,057	21299	0,120	20100	0,057
			20346	26460	0,301	23111	0,136	25560	0,256	23376	0,149	25680	0,262	22945	0,128	22945	0,128
			18734	21780	0,163	21366	0,140	21180	0,131	20950	0,118	21180	0,131	20485	0,093	20485	0,093
40	1	1	21660	23220	0,072	26038	0,202	22320	0,030	27438	0,267	26460	0,222	31320	0,446	22320	0,030
Total			620403	756060	0,219	733393	0,182	746520	0,203	752872	0,214	755400	0,218	758121	0,222	685827	0,105

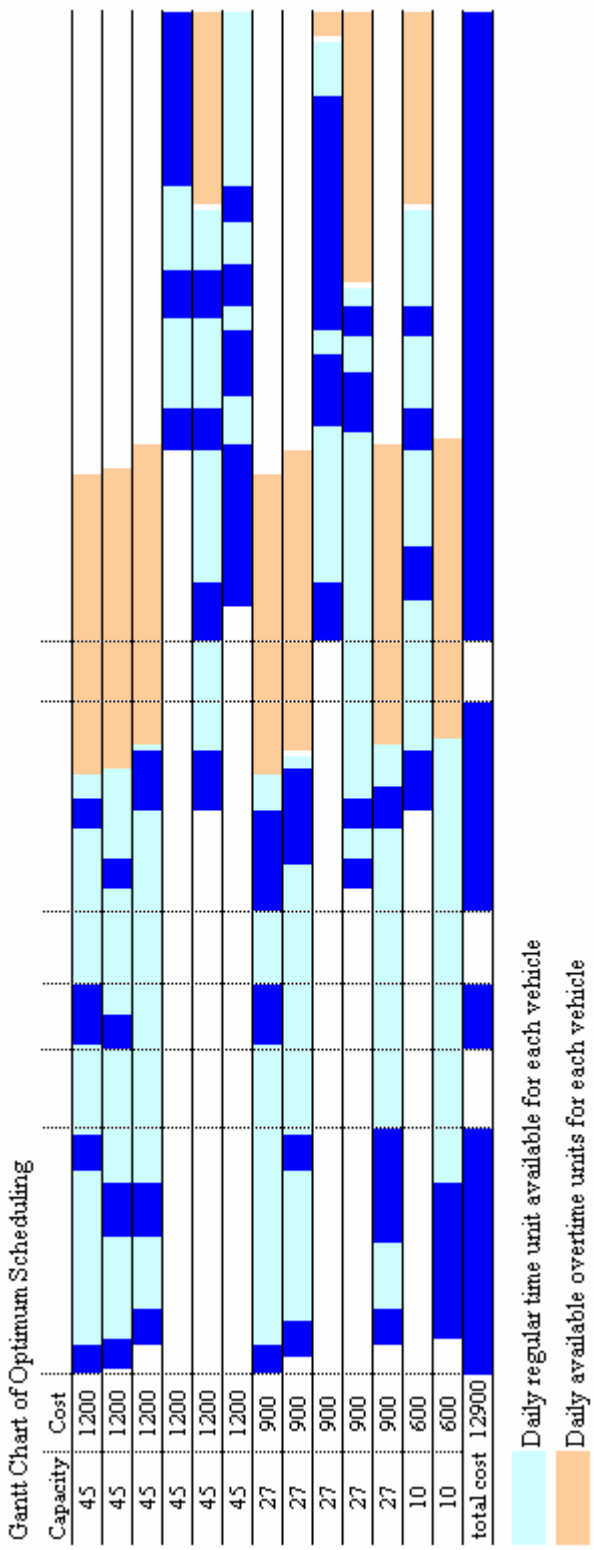
Appendix B.2: Comparison of All Optimum Solutions with Lower Bound Values

# of Trip	p	r	Opt. Solution	Lower Bound Solution											
				LB1	LB2	LB3	LB4	LB51	LB52	LB53	LB54	LB55	LB56	max. LB	Gap
20	1	1	8705	2184	3393	4041	4390	6000	6310	6000	5580	6600	7000	7000	0,196
			9264	3492	3639	5384	5832	7200	6220	7200	6240	7800	7260	7800	0,158
			11820	5376	4343	6760	7299	7200	8010	7200	7910	7800	8580	8580	0,274
			11220	3840	3211	4825	5200	7800	9470	7800	9300	7800	8680	9470	0,156
			8780	2592	3206	4305	4678	5400	6100	5400	5840	5400	4640	6100	0,305
20	1	2	11540	3240	4634	5820	6329	7200	7250	7200	6840	7200	8160	8160	0,293
			12991	3408	5901	6974	7607	9000	7650	8400	7660	9000	10380	10380	0,201
			14240	4080	6542	7975	8681	9000	9080	9000	8640	9000	10240	10240	0,281
			13540	3432	6070	6866	7480	8400	9790	8400	8930	10200	11750	11750	0,132
			12836	3756	5650	7296	7953	9000	8610	9000	8140	9000	7680	9000	0,299
20	2	1	13535	2856	3726	4899	5322	10800	7410	10800	7170	10800	7780	10800	0,202
			13090	3084	3924	5101	5556	9600	7840	9600	7670	9600	8230	9600	0,267
			13651	3324	4748	5883	6393	10800	9860	10800	8970	10800	9940	10800	0,209
			8973	3252	3247	4825	5212	6000	5970	6000	5910	6600	4920	6600	0,264
			18155	3144	3848	5073	5487	13200	12480	13200	11820	13200	12300	13200	0,273
20	2	2	11715	3768	4366	5911	6408	8400	8510	8400	6950	8400	8530	8530	0,272
			17151	2964	4503	5296	5771	13200	10420	13200	10200	12000	10560	13200	0,230
			16977	3672	5101	6034	6553	13200	12110	13200	11770	13200	12320	13200	0,222
			9860	3156	4415	5229	5681	6600	6530	6600	5910	6600	6200	6600	0,331
			14390	3624	4759	6133	6671	12000	10050	12000	9240	12000	9890	12000	0,166
30	1	1	11742	4728	5623	7252	7856	6600	7320	7200	6320	9000	7760	9000	0,234
			14308	4212	6283	7872	8568	10200	10020	10200	9600	10200	9080	10200	0,287
			12675	4548	5859	7395	8028	8400	8790	8400	8060	7800	10240	10240	0,192
			12900	4176	4898	6621	7179	9000	11000	9000	9750	9000	10910	11000	0,147
			18740	5112	6074	7863	8519	12000	12220	12000	10810	12600	13430	13430	0,283
30	1	2	15560	5976	7902	9992	10833	10200	12050	10200	10540	9600	10490	12050	0,226
			12340	5844	7641	10220	11085	9000	10580	9000	9470	7800	9420	11085	0,102
			11390	4380	5275	7113	7714	7800	7890	8400	6760	8400	7540	8400	0,263
			14040	5580	6543	8626	9370	9600	8870	9600	8630	9600	9070	9600	0,316
			16152	6156	7006	9469	10261	10800	13340	10800	12010	12600	14520	14520	0,101
30	2	1	19596	5664	6849	8701	9439	16800	11850	16800	12130	15600	12860	16800	0,143
			19940	5184	6038	7736	8388	16200	13510	16200	13610	16200	14500	16200	0,188
			19130	4272	5351	6966	7566	12600	11890	12600	11320	12600	11980	12600	0,341
			16046	6144	6185	8746	9475	12000	10120	12600	9800	10800	9880	12600	0,215
			24710	6036	7938	10545	11480	16200	15880	16200	14410	16200	15910	16200	0,344
30	2	2	21125	5496	7279	9698	10537	13800	15420	13800	14120	14400	15480	15480	0,267
			27810	6036	9337	11290	12273	18000	17500	18000	16920	16800	17360	18000	0,353
			19024	5088	5977	7981	8660	14400	13900	14400	13330	12600	13610	14400	0,243
			20346	5796	7259	9235	10009	16200	13410	16200	12770	17400	13470	17400	0,145
			18734	7380	7740	10259	11114	13800	13390	13800	13020	13800	12900	13800	0,263
40	1	1	21660	6060	8889	11101	12094	11400	14550	12000	13300	12600	15940	15940	0,264
Total			620401											471955	0,239

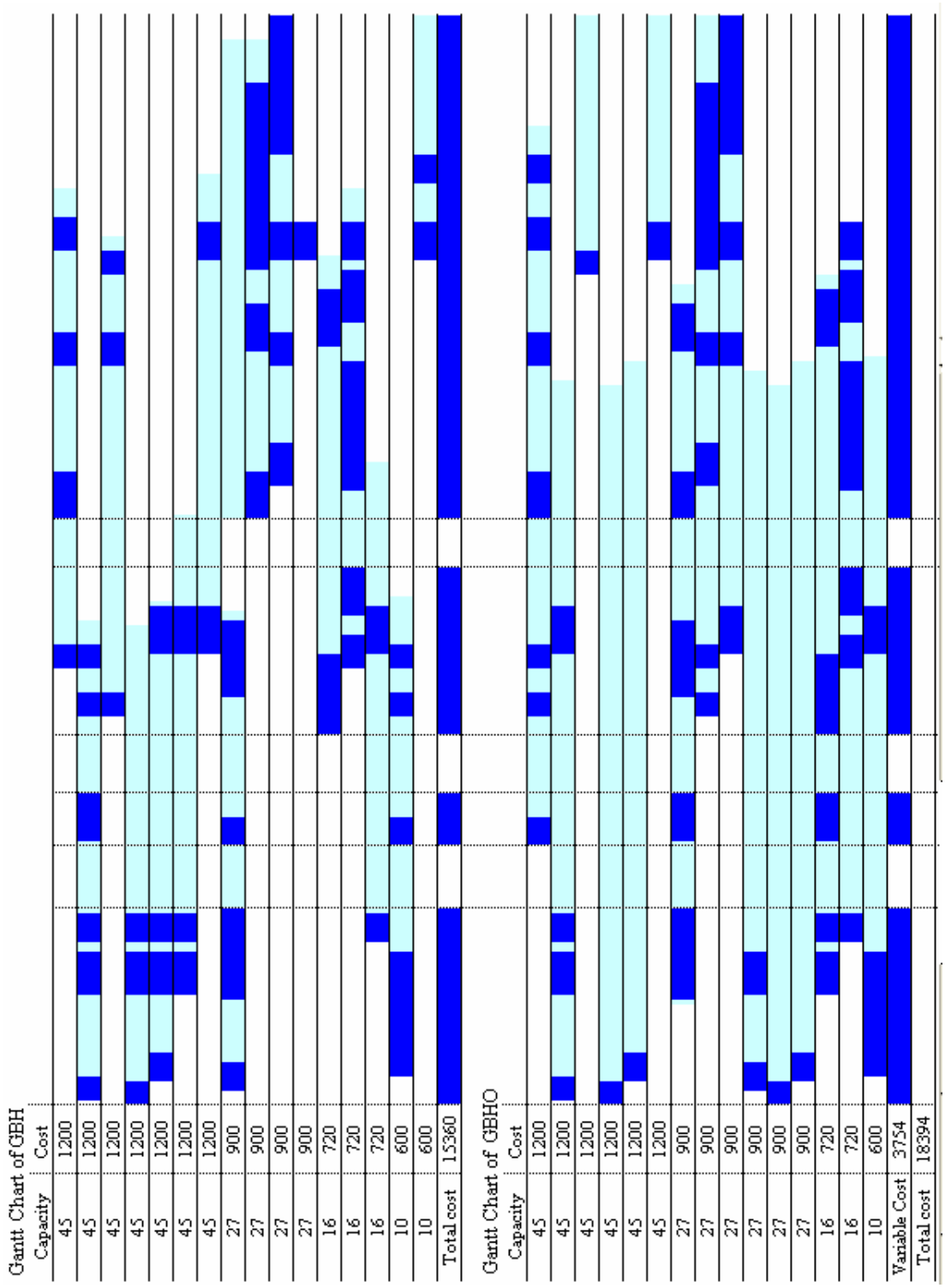
Appendix C.1: Gantt Chart of Ready Time and Processing Times of All Trips



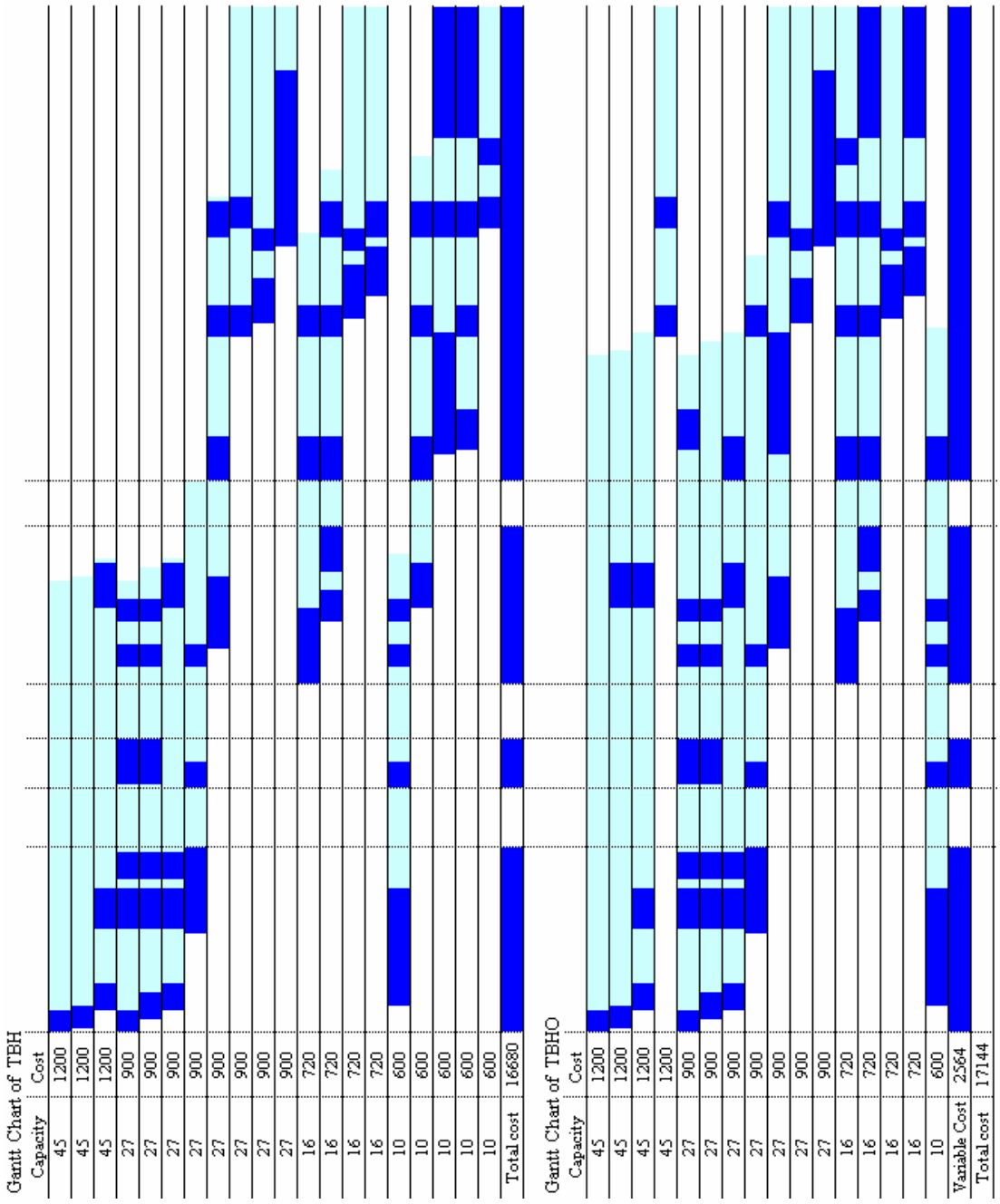
Appendix C.2: Gantt Chart of Optimum Scheduling



Appendix C.3: Gantt Chart of Scheduling of GBH and GBHO Methods



Appendix C.4: Gantt Chart of Scheduling of TBH and TBHO Methods



Appendix C.5: Gantt Chart of Scheduling of VBH and VBHO Methods